Fusion multiphysique pour système de localisation indoor

Abdelhak Bougouffa

université
PARIS-SACLAY

Fusion Multiphysique pour Système de Localisation Indoor

Multiphysic Fusion for Indoor Localization System

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Robotique
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **SATIE** (Université Paris-Saclay, ENS Paris-Saclay, CNRS),
sous la direction de **Samir BOUAZIZ**, Professeur des universités,
et la co-direction d'**Emmanuel SEIGNEZ**, Maître de conférences HDR

Thèse soutenue à Paris-Saclay, le 20 septembre 2023, par

Abdelhak BOUGOUFFA

Composition du jury

Membres du jury avec voix délibérative

Pierre BLAZEVIC *Président*
Professeur, Université Versailles-Saint-Quentin - Université Paris-Saclay
Kurosh MADANI *Rapporteur & Examinateur*
Professeur, Université Paris-Est Créteil
Antoine MANZANERA *Rapporteur & Examinateur*
Maître de conférences HDR, ENSTA Paris - Institut Polytechnique de Paris
Fawzi NASHASHIBI *Examinateur*
Directeur de recherche, INRIA Paris-Rocquencourt

THÈSE DE DOCTORAT
NNT : 2023UPAST110

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Fusion Multipysique pour Système de Localisation Indoor
Mots clés : robotique, estimation d'état, localisation indoor, fusion de données, environnements industriels, ceiling-vision, LiDAR

Résumé : Ce travail s'inscrit dans le cadre d'une thèse Cifre réalisée en collaboration entre le laboratoire SATIE et l'entreprise ez-Wheel. L'entreprise développe des roues motorisées compactes et hautement intégrées pour faciliter la motorisation des plateformes mobiles destinées au transport des lourdes charges. Pendant la durée de cette thèse, une nouvelle génération de produits a été développée par l'entreprise, spécifiquement conçue pour des applications à la robotique mobile sûre dans des environnements industriels. Pour assurer une navigation sécurisée, les plateformes robotiques équipées de ces roues nécessitent un système de localisation adapté aux environnements industriels visés. Dans cette thèse, nous abordons ce problème de localisation en étudiant d'abord les approches de localisation disponibles dans la littérature scientifique. Puis, dans un premier temps, nous proposons une solution basée sur la fusion multi-LiDARs en supposant un environnement statique ou faiblement dynamique. Cependant, dans les environnements industriels visés, nous nous attendons à des environnements hautement dynamiques. Après avoir observé que dans les environnements cibles, le plafond représente un espace invariant et entièrement statique, nous avons proposé une nouvelle approche qui consiste à séparer l'espace de navigation de l'espace de localisation. Ainsi, pour détecter les obstacles et assurer la sûreté des déplacements, nous exploitons un LiDAR 2D sur le plan horizontal. Et pour garantir une localisation relative robuste, nous utilisons une caméra orientée vers le plafond (plan vertical). Ce découplage nous permet d'éliminer les problèmes liés à la détection et au filtrage des objets dynamiques, tout en améliorant la qualité de la localisation. Nous avons proposé un système de localisation par vision verticale à base de la méthode directe DSO, avec une validation expérimentale sur notre plateforme. Enfin, afin de valider notre approche dans des environnements réels, nous avons conçu et réalisé une expérimentation pour collecter un jeu de données multicapteurs centré sur la vision verticale. Ce travail est unique, car dans la littérature scientifique, il n'existe pas de jeux de données permettant d'évaluer des méthodes de localisation par caméra verticale. Les choix méthodologiques et technologiques adoptés dans ce travail ont été fortement influencés par le contexte industriel de la thèse. Le but étant de proposer un prototype de maturité technologique de niveau 6 (TRL6) pour son intégration ultérieure dans les produits de l'entreprise.

Title : Multiphysic Fusion for Indoor Localization System
Keywords : robotics, state estimation, indoor localization, data fusion, industrial environments, ceiling-vision, LiDAR

Abstract : This work is part of a Cifre thesis carried out in collaboration between the SATIE laboratory and the ez-Wheel company. The company specializes in developing compact, integrated motorized wheels that facilitates the motorization and the automation of mobile platforms aimed at transporting heavy loads. Throughout the course of this project, the company introduced a new generation of products tailored for safe mobile robotics applications in industrial environments. A critical requirement for developing an autonomous mobile robot based on these wheels is a localization system that is suitable for the targeted industrial environments. To address this requirement, we initially investigated the localization approaches proposed in the scientific literature. Our first proposed solution is based on multi-LiDARs fusion while assuming a static or weakly dynamic environment. However, the targeted industrial environments are expected to have highly dynamic surroundings. Therefore, after observing that in such environments, the ceiling remains invariant and mostly static, we introduced a new approach that separates the navigation space from the localization space. Thus, to detect obstacles and ensure safe movements, we utilize a 2D LiDAR on the horizontal plane. While we use a camera oriented towards the ceiling (vertical plane) to guarantee a robust relative localization. This decoupling effectively eliminates the issues related to the detection and the filtering of dynamic objects, thereby improving the localization quality without extra processing cost. We have proposed a ceiling-vision localization system based on the Direct Sparse Odometry (DSO), and validated its performance and accuracy on our experimental platform. Finally, to further validate our approach in real-world scenarios, we designed and conducted an experiment to collect a multisensor dataset focused on ceiling-vision. The novelty of this work resides in the fact that there are no available datasets in the scientific literature that allow the evaluation of ceiling-camera localization methods. The methodological and technological choices of this work were strongly influenced by the industrial context of the thesis, aiming to present a TRL 6 prototype for subsequent integration into the company's products.

Fusion Multiphysique pour Système de Localisation Indoor

Multiphysic Fusion for Indoor Localization System

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Robotique
Graduate School : Sciences de l'ingénierie et des systèmes
Référent : Faculté des sciences d'Orsay
Unité de recherche : SATIE (Université Paris-Saclay, ENS Paris-Saclay, CNRS)
Numéro national de thèse (NNT) : 2023UPAST110

Thèse soutenue à Paris-Saclay, le 20 septembre 2023, par

Abdelhak BOUGOUFFA

En présence de

Samir BOUAZIZ Professeur, Université Paris-Saclay	*Directeur de thèse*
Emmanuel SEIGNEZ Maître de conférences HDR, Université Paris-Saclay	*Codirecteur de thèse*
Florian GARDES Directeur technique, ez-Wheel	*Superviseur en entreprise*
Pierre BLAZEVIC Professeur, Université Versailles-Saint-Quentin - Université Paris-Saclay	*Président du jury*
Kurosh MADANI Professeur, Université Paris-Est Créteil	*Rapporteur & Examinateur*
Antoine MANZANERA Maître de conférences HDR, ENSTA Paris - Institut Polytechnique de Paris	*Rapporteur & Examinateur*
Fawzi NASHASHIBI Directeur de recherche, INRIA Paris-Rocquencourt	*Examinateur*
Grégory MERMET Ingénieur - responsable du développement logiciel, ez-Wheel	*Invité*

﴿ ٱلَّذِى عَلَّمَ بِٱلْقَلَمِ ۝٤ عَلَّمَ ٱلْإِنسَـٰنَ مَا لَمْ يَعْلَمْ ۝٥ ﴾

« qui a enseigné par la plume, a enseigné à l'être humain ce qu'il ne savait pas. »

À la mémoire de mon grand frère bien-aimé, Abdelali (Dif)*, qui a quitté ce monde subitement pendant que je préparais cette thèse. Il était bien plus qu'un frère pour moi, il était mon deuxième père, mon pilier de force, mon guide et mon ami le plus cher.*

Quand notre père, que la Miséricorde de Dieu soit sur lui, nous a quittés, je n'avais que 9 mois. Abdelali *n'a pas tardé à prendre sur lui la responsabilité de subvenir aux besoins de la famille. Il a sacrifié sa propre éducation pour que je puisse avoir une chance de réussir. Il a abandonné ses propres rêves pour que les miens se concrétisent. J'ai vécu toute ma vie sans père, mais ce n'est qu'après la perte d'*Abdelali *que je me suis senti orphelin.*

Je suis fier de dire que tout ce que j'ai accompli, je le dois en grande partie à toi. Sans ton soutien, ta générosité, ta bienveillance, tes sacrifices et ton amour inconditionnel, je n'aurais jamais pu atteindre les sommets que j'ai atteints.

Ainsi, c'est avec une profonde gratitude, une grande tristesse et un immense chagrin que je dédie ce modeste travail à ta mémoire. Puisse Dieu, Le Tout-Puissant, t'accorder Son infinie Miséricorde et t'accueillir dans Son éternel Paradis.

Ton petit frère

Remerciements

Je souhaite exprimer ma sincère gratitude envers monsieur Antoine Manzanera et monsieur Kurosh Madani pour avoir accepté d'être les rapporteurs de ma thèse. Vos remarques et vos commentaires extrêmement pertinents ont grandement contribué à améliorer la qualité de ce travail. Mes sincères remerciements vont également au président du jury, monsieur Pierre Blazevic ainsi qu'à l'examinateur, monsieur Fawzi Nashashibi, qui ont donné de leur temps et de leur effort en acceptant d'évaluer ce modeste travail.

Je remercie chaleureusement mon directeur de thèse Samir Bouaziz, sans qui, ce travail n'aurait jamais vu le jour, pour ses conseils éclairés, son soutien inébranlable et pour toutes les discussions stimulantes que nous avons eues. Je remercie par ailleurs mon codirecteur de thèse Emmanuel Seignez pour son suivi régulier, ses précieux conseils, ses remarques pertinentes, son soutien constant et sa disponibilité, malgré ses nombreux engagements et responsabilités.

Je tiens aussi à exprimer mes sincères remerciements envers Florian Gardes et Antoine Juan pour leur confiance et pour l'accueil chaleureux qu'ils m'ont réservé au sein de leur entreprise. Merci d'avoir rendu ce travail possible, votre présence et votre engagement tout au long de cette période de thèse ont été d'une valeur inestimable. Je tiens par ailleurs à remercier tous mes collègues d'ez-Wheel pour leur collaboration, leur soutien et leur convivialité tout au long de cette aventure, avec une pensée spéciale à Grégory Mermet.

J'aimerais exprimer ma gratitude envers tous mes collègues chercheurs et doctorants du groupe MOSS du laboratoire SATIE. Je vous remercie pour tous les moments conviviaux et les discussions stimulantes à la machine à café. Mes remerciements ne sauraient être suffisants envers mon cher Stéphane Espié, chef du groupe MOSS, un chercheur exceptionnel avec qui j'ai eu le privilège de collaborer avant et pendant cette thèse. Merci pour ta confiance et pour tout ce que j'ai pu apprendre de toi, tu as été une véritable source d'inspiration et un soutien inestimable.

Je souhaite par ailleurs exprimer mes remerciements et mes pensées les plus sincères envers mes anciens enseignants à l'université de Boumerdes–Algérie : Hafidha Boumeridja, M'hamed Hamadouche et Samira Mechid, ainsi que mes anciennes collègues au CDTA : Sara Bouraine et Ouahiba Azouaoui. Ces personnes occupent une place spéciale dans mon cœur, ayant eu un impact immense sur mes choix de carrière. Ils m'ont fait confiance, m'ont encouragé, et ont été une source inestimable de soutien et d'inspiration. J'ai également une pensée particulière à Khaled Kara, un ingénieur-électronicien hors pair avec qui j'ai eu

le plaisir de collaborer. Je lui suis reconnaissant pour m'avoir appris les bonnes pratiques de l'électronique, allant de la conception de circuits imprimés multicouches jusqu'aux anciennes techniques de wrapping.

Cette réalisation découle d'un long parcours de vie qui a été rendu possible grâce à une famille exceptionnelle qui m'a toujours soutenue dans la poursuite de mes passions. C'est pourquoi je tiens à exprimer ma profonde gratitude et mes sentiments les plus sincères envers ma mère qui a toujours œuvré inlassablement pour ma réussite, avec son amour inconditionnel, son soutien indéfectible, ses sacrifices et ses précieux conseils. Merci pour tout ce que tu as fait et pour ta présence constante dans ma vie. Je suis pareillement reconnaissant envers mes sœurs Mounira et Amina ainsi que mes frères Abdelhalim et Mustapha, qui ont toujours été présents à mes côtés. Je tiens aussi à rendre hommage à mon regretté frère Abdelali, que la Miséricorde de Dieu soit sur lui, qui nous a quittés pendant que je préparais ma thèse.

Je tiens à exprimer ma profonde gratitude et mon amour incommensurable envers ma femme, Imane ♡, qui m'a accompagné et soutenu tout au long de cette aventure, et même bien avant. Je tiens par ailleurs à remercier mon beau-père Abdellah ainsi que mes belles-sœurs Mira, Wiza et Hayat et mes beaux-frères Amirouche, Samad, Azzou et Ghanou. Merci pour votre présence et votre soutien.

Je remercie aussi mes oncles maternels Mohamed, Mahmoud, et Abderrezzak et ma tante Fatma-Zohra, qui ont toujours été une précieuse source de soutien tout au long de ma vie. J'ai une pensée particulière à ma regrettée tante, et ma deuxième mère, Hadjira, qui nous quittée subitement alors que je préparais cette thèse, que Dieu t'accueille dans Son Éternel Paradis. Je tiens aussi à remercier mes oncles paternels Abdelkader, El-Eid, Yacine et Naamane, ainsi que tous mes proches, qu'ils soient grands ou petits, avec une pensée spéciale pour mon cousin Ismail, avec qui j'ai grandi.

Enfin, je tiens à remercier tous mes chers amis, avec qui j'ai partagé un long chemin de vie, et qu'ont été toujours présents même pendant les moments les plus difficiles. Merci Mohamed Yacine El Haddad pour tes conseils et ton aide. Mes remerciements vont également à Adel El Haddad, Anis Hocine, Badereddine Rebai, Hachemi Djemai, Hamza Sadaoui, Ilyes Kadri, Mohamed Amine Dridi, Nadir Nafa, Othmane Touat, Saïd Sofiane, Saifeddine Boularaoui, Sidali Nahnah, Zineddine Guennoun, et à toutes les personnes qui ont contribué, de près ou de loin, à cette petite réussite.

Abdelhak

Résumé

Ce travail s'inscrit dans le cadre d'une thèse Cifre réalisée en collaboration entre le laboratoire SATIE et l'entreprise ez-Wheel. L'entreprise développe des roues motorisées compactes et hautement intégrées pour faciliter la motorisation des plateformes mobiles destinées au transport des lourdes charges. Pendant la durée de cette thèse, une nouvelle génération de produits a été développée par l'entreprise, spécifiquement conçue pour des applications à la robotique mobile sûre dans des environnements industriels. Pour assurer une navigation sécurisée, les plateformes robotiques équipées de ces roues nécessitent un système de localisation adapté aux environnements industriels visés. Dans cette thèse, nous abordons ce problème de localisation en étudiant d'abord les approches de localisation disponibles dans la littérature scientifique. Puis, dans un premier temps, nous proposons une solution basée sur la fusion multi-LiDARs en supposant un environnement statique ou faiblement dynamique. Cependant, dans les environnements industriels visés, nous nous attendons à des environnements hautement dynamiques. Après avoir observé que dans les environnements cibles, le plafond représente un espace invariant et entièrement statique, nous avons proposé une nouvelle approche qui consiste à séparer l'espace de navigation de l'espace de localisation. Ainsi, pour détecter les obstacles et assurer la sûreté des déplacements, nous exploitons un LiDAR 2D sur le plan horizontal. Et pour garantir une localisation relative robuste, nous utilisons une caméra orientée vers le plafond (plan vertical). Ce découplage nous permet d'éliminer les problèmes liés à la détection et au filtrage des objets dynamiques, tout en améliorant la qualité de la localisation. Nous avons proposé un système de localisation par vision verticale à base de la méthode directe DSO, avec une validation expérimentale sur notre plateforme. Enfin, afin de valider notre approche dans des environnements réels, nous avons conçu et réalisé une expérimentation pour collecter un jeu de données multicapteurs centré sur la vision verticale. Ce travail est unique, car dans la littérature scientifique, il n'existe pas de jeux de données permettant d'évaluer des méthodes de localisation par caméra verticale. Les choix méthodologiques et technologiques adoptés dans ce travail ont été fortement influencés par le contexte industriel de la thèse. Le but étant de proposer un prototype de maturité technologique de niveau 6 (TRL6) pour son intégration ultérieure dans les produits de l'entreprise.

Mots clés : robotique, estimation d'état, localisation indoor, fusion de données, environnements industriels, ceiling-vision, LiDAR

Abstract

This work is part of a Cifre thesis carried out in collaboration between the SATIE laboratory and the ez-Wheel company. The company specializes in developing compact, integrated motorized wheels that facilitates the motorization and the automation of mobile platforms aimed at transporting heavy loads. Throughout the course of this project, the company introduced a new generation of products tailored for safe mobile robotics applications in industrial environments. A critical requirement for developing an autonomous mobile robot based on these wheels is a localization system that is suitable for the targeted industrial environments. To address this requirement, we initially investigated the localization approaches proposed in the scientific literature. Our first proposed solution is based on multi-LiDARs fusion while assuming a static or weakly dynamic environment. However, the targeted industrial environments are expected to have highly dynamic surroundings. Therefore, after observing that in such environments, the ceiling remains invariant and mostly static, we introduced a new approach that separates the navigation space from the localization space. Thus, to detect obstacles and ensure safe movements, we utilize a 2D LiDAR on the horizontal plane. While we use a camera oriented towards the ceiling (vertical plane) to guarantee a robust relative localization. This decoupling effectively eliminates the issues related to the detection and the filtering of dynamic objects, thereby improving the localization quality without extra processing cost. We have proposed a ceiling-vision localization system based on the Direct Sparse Odometry (DSO), and validated its performance and accuracy on our experimental platform. Finally, to further validate our approach in real-world scenarios, we designed and conducted an experiment to collect a multisensor dataset focused on ceiling-vision. The novelty of this work resides in the fact that there are no available datasets in the scientific literature that allow the evaluation of ceiling-camera localization methods. The methodological and technological choices of this work were strongly influenced by the industrial context of the thesis, aiming to present a TRL 6 prototype for subsequent integration into the company's products.

Keywords : robotics, state estimation, indoor localization, data fusion, industrial environments, ceiling-vision, LiDAR

Liste d'acronymes

AAA	Algorithm-Architecture Adequacy — *Adéquation Algorithme-Architecture*
AGV	Automated Guided Vehicle — *Véhicule guidé automatisé*
AMR	Autonomous Mobile Robot — *Robot mobile autonome*
AoA	Angle of Arrival — *Angle d'arrivée*
BA	Bundle Adjustment — *Ajustement de faisceaux*
BBoW	Bag of Binary/Visual Words — *Sac de mots binaires/visuels*
BLDC	Brushless DC (electric motor)
BMS	Battery Management System — *Système de gestion de batterie*
BRIEF	Binary Robust Independent Elementary Features (feature descriptor)
CAN	Controller Area Network (communication bus)
CEM	Compatibilité électromagnétique
CIR	Centre Instantané de Rotation
CMOS	Complementary Metal Oxide Semiconductor
CSD	Ceiling Space Density
CV	Ceiling-Vision — *Vision au plafond / Vision verticale*
CW-LiDAR	Continuous-wave based LiDAR — *LiDAR à base d'ondes continues*
DFD	Data-Feature-Decision — *Donnée-Attribut-Décision*
DSO	Direct Sparse Odometry — *Odométrie directe éparse*
DVS	Dynamic Vision Sensor — *Capteur de vision dynamique*
DoF	Degree of Freedom — *Degrée de liberté*
E/S	Entrée/Sortie
EKF	Extended Kalman Filter — *Filtre de Kalman étendu*
ER	Erreur relative
FAST	Features from Accelerated Segment Test (feature descriptor & extractor)
FPS	Frame per Second
FoV	Field of View — *Champs de vision*
GPS	Global Positioning System — *Système mondial de positionnement*
GT	Ground Truth — *Réalité terrain / Vérité terrain*
HDR	High Dynamic Range — *Haute plage dynamique*
HITL	Human-in-the-loop — *Humain dans la boucle*
IBL	Image-based Localization — *Localisation basée sur l'image*
ICL	Iterative Closest Line — *Itératif de la ligne la plus proche*
ICP	Iterative Closest Point — *Itératif du point le plus proche*
IDC	Iterative Dual Matching
IMRP	Iterative Matching Range Point
IMU	Inertial Measurement Unit — *Unité de mesures inertielles*
IQR	Interquartile range — *Intervalle interquartile*
JDL	Joint Directors of Laboratories
KF	Kalman Filter — *Filtre de Kalman*
LiDAR	Light Detection and Ranging — *Détection et estimation de distance par laser*

LoS Line of Sight — *Ligne de mire*
MC Monte Carlo
MCL Monte Carlo Localization
MHT Multiple-Hypothesis Tracking — *Suivi multihypothèses*
MOSS Méthodes et Outils pour les Signaux et Systèmes, *groupe de recherche du SATIE*
MVS Multi-View Stereo — *Stéréo multivues*
MoCap Motion Capture — *Capture de mouvement*
NDT Normal Distribution Transform — *Transformée de distribution normale*
NFC Near-field Communication
OF Optical Flow — *Flux optique*
OG Occupancy Grid — *Grille d'occupation*
ORB Oriented FAST and Rotated BRIEF (feature descriptor & extractor)
OSSD Output Signal Switching Device
PB-LiDAR Pulse-based LiDAR — *LiDAR à base d'impulsions*
PDF Probability Density Function — *Fonction de densité de probabilité*
PF Particle Filter — *Filtre particulaire* ou *à particules*
PM Perfect Match
PSO Particle Swarm Optimization — *Optimisation par essaims de particules*
PTAM Parallel Tracking and Mapping
PnP Perspective-n-Point
RANSAC Random sample consensus
RFID Radio-frequency Identification
RGB Red, Green, Blue — *Rouge, vert, bleu*
RGB-D RGB with Depth — *RGB avec profondeur*
ROS Robot Operating System
RPROP Resilient Back-Propagation — *Rétropropagation résiliente*
RSSI Received Signal Strength Indication — *Indication de la puissance du signal reçu*
RTK Real-time kinematic
SATIE Sciences et Applications des Technologies de l'Information et de l'Énergie, *laboratoire*
SBC Safe Brake Control
SDI Safe Direction Indication
SIFT Scale-Invariant Feature Transform (feature descriptor & extractor)
SIL Safety Integrity Level
SLAM Simultaneous Localization and Mapping — *Localisation et cartographie simultanées*
SLS Safe Limited Speed
SMC Sequential Monte Carlo — *Monte Carlo séquentiel*
SPI Serial Peripheral Interface
STO Safe Torque Off
SURF Speeded-Up Robust Features (feature descriptor)
SVD Singular Value Decomposition — *Décomposition en valeurs singulières*
SWD Safety Wheel Drive
SfM Structure from Motion — *Structure à partir du mouvement*
SoM System on Module — *Système sur module*
TDoA Time Difference of Arrival — *Décalage de temps d'arrivée*
TRL Technology Readiness Level — *Niveau de maturité technologique*
ToA Time of Arrival — *Temps d'arrivée*
ToF Time of Flight — *Temps de vol*
UAV Unmanned Areal Vehicules — *Drone*
UKF Unscented Kalman Filter — *Filtre de Kalman sans parfum*
USB Universal Serial Bus
UWB Ultra-Wide Band
VIO Visual-Inertial Odometry — *Odométrie visuelle et inertielle*
VLC Visible Light Communication — *Communication par lumière visible*

VO	Visual Odometry — *Odométrie visuelle*
VPR	Visual Place Recognition — *Reconnaissance visuelle des lieux*
vSLAM	Visual SLAM — *SLAM visuel / à base de vision*

CHAPITRE 1

Introduction générale

1.1 Une brève histoire de la robotique

La conception de machines automates a longuement fasciné l'homme. Bien que des légendes anciennes citent des machines de ce genre, les premières traces écrites que nous pouvons trouver remontent à l'Alexandrie grecque notamment les travaux de Ctésibios (285-222 av. J.-C.) et ceux d'Héron d'Alexandrie (10-70 av. J.-C.). Le scientifique chinois Su Song (1020-1101) a également documenté des automates, et plus spécifiquement son horloge hydraulique décrite dans son livre *« Xinyi Xiangfayao »* traduit à *« L'essentiel d'une nouvelle méthode de mécanisation de la rotation d'une sphère armillaire et d'un globe céleste »*.

Les premières traces de machines automates programmables sont apparues dans les travaux d'Ismail ibn al-Razaz al-Jazari (1136-1206) [1], décrit par certains comme *« le père de la robotique »* [2, 3], suite à son *« Livre de la connaissance des dispositifs mécaniques ingénieux »*, où il documente la conception de plusieurs machines automates hydrauliques. Parmi les inventions décrites dans le livre, nous trouvons *le bateau d'al-Jazari*, intégrant des robots humanoïdes programmables qui jouent des pièces de musique (figure 1.1).

Plusieurs autres inventions de ce type sont apparues par la suite, parmi les plus connues, nous pouvons citer les travaux de Leonardo da Vinci (1452-1519), notamment son chariot autopropulseur. En 1912, Leonardo Torres Quevedo (1852-1936) inventa *El Ajedrecista* (signifiant *« le joueur d'échecs »* en espagnol), l'automate électromécanique qui permet de jouer aux échecs (voir la figure 1.2), ce dernier peut être qualifié de la première machine autonome réelle [5]. Contrairement au *turc mécanique*, faux automate fonctionnant grâce à un humain, *El Ajedrecista* comporte une véritable automatisation intégrée, permettant de jouer aux échecs sans intervention humaine.

(a) Illustration du bateau prise du livre [4] **(b)** Reconstitution lors d'une exposition en Turquie

Figure 1.1 – Le bateau d'AL-JAZARI. Les humanoïdes, pilotés par un système hydraulique, peuvent être programmés mécaniquement pour jouer des pièces de musique.

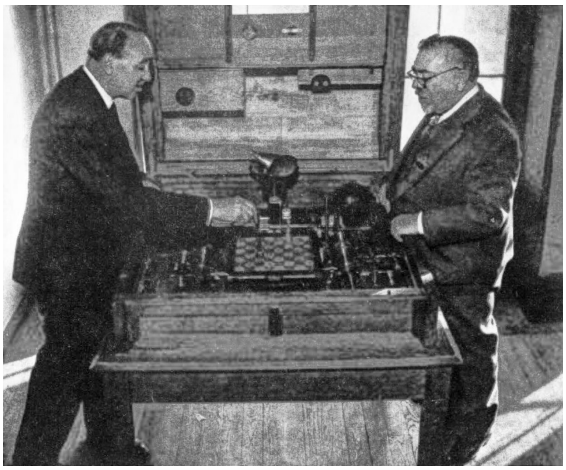


Figure 1.2 – Le fils de TORRES (à gauche), montrant l'automate *El Ajedrecista* au mathématicien Norbert WIENER lors du *Congrès international de cybernétique et de systémique de Paris* de 1951 (photo en domaine public, Wikimedia Commons).

Le terme *« robot »* apparaît pour la première fois en 1920 [6], dans une pièce de théâtre de science-fiction de l'écrivain tchécoslovaque Karel ČAPEK (1890-1938), intitulée *« Rossumovi Univerzální Roboti »*, traduite à *« Rossum's Universal Robots »* [7] et plus connue sous l'acronyme R.U.R. (voir la figure 1.3).

La fréquence d'apparition du mot « robot » dans la littérature vient confirmer cette information. L'usage du terme commence à partir des années 1920, comme le démontre le graphe illustré dans la figure 1.4.

Le mot « robot » a été originalement proposé à Karel par son frère Josef [7], ce dernier trouve inspiration dans le mot tchèque « robota », qui signifie « travail » ou « servage ». Il propose donc le mot « robot » à Karel qui l'utilise dans R.U.R. pour désigner ses personnages, présentés comme des machines humanoïdes réduites seulement à leurs forces de travail.

Figure 1.3 – Photographie signée « Henri MANUEL, Paris », prise lors de la mise en scène de R.U.R. par Fedor KOMISSARJEVSKY sous la direction de Jacques HÉBERTOT, le 26 mars 1924 à la *Comédie des Champs-Élysées à Paris*. À droite : l'affiche décorative sur le mur, extraite de la photo, affichant sous forme publicitaire le texte : *« Voulez-vous vendre à bas prix ? Commandez des robots »*.

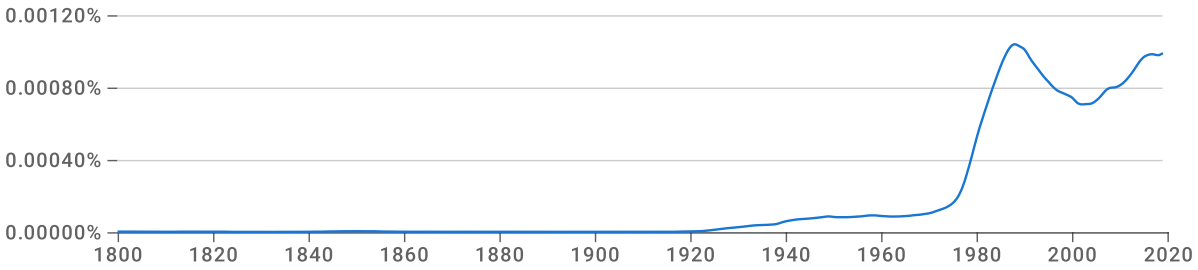


Figure 1.4 – La fréquence d'apparition du terme « robot » dans la littérature en Anglais. Graphe tracé à partir des données de *Google Books Ngram Viewer* de 1800 jusqu'à 2020 avec un lissage sur 5 ans.

Le terme « robotique », désignant l'art de construire des robots, vient lui aussi de la science-fiction. Il est apparu pour la première fois en 1941 dans la nouvelle *« Liar ! »* de l'écrivain et biochimiste russo-américain Isaac ASIMOV (1920-1992), cette nouvelle a ensuite été incluse dans le célèbre livre d'ASIMOV *« I, Robot »*. Ainsi, la robotique a été largement vulgarisée dans les années 1950-1970 grâce à la littérature de science-fiction.

Les premiers systèmes robotiques modernes commencent à apparaître au début des années 1940. La table 1.1 adaptée des travaux de NIKU [8], récapitule l'histoire moderne de la robotique.

Les systèmes robotisés peuvent être divisés à quatre catégories majeures [9], à savoir :

1. Les robots stationnaires de type bras/manipulateur.
2. Les robots mobiles terrestres (à roues, à pattes, etc).
3. Les robots volants (drones).
4. Les robots sous-marins.

Table 1.1 – Récapitulatif de l'histoire moderne de la robotique (adapté de [8]).

Date	**Événements majeurs**
1920	L'auteur Karel Čapek écrit la pièce R.U.R. qui donne naissance au terme *« robot »*.
1941	Première apparition du mot *« robotique »* dans la nouvelle *« Liar ! »* d'Isaac Asimov.
1946	George Devol développe le contrôleur magnétique. J. Presper Eckert et John Mauchly construisent l'ordinateur *ENIAC* à l'université de Pennsylvanie.
1952	La première machine contrôlée numériquement a été construite à MIT.
1954	Devol développe le premier robot programmable moderne.
1961	Brevet attribué aux États-Unis à Devol pour le *« Programmed Article Transfer »* [10], une base pour le premier robot industriel *« Unimate »*.
1962	Création de la société Unimation, l'apparition des premiers robots industriels, et General Motors installe ses premiers robots d'Unimation.
1967	Unimation présente le robot *Mark II*.
1968	Le robot *SHAKEY* est construit au Stanford Research Institute (SRI), reconnue par le programme *IEEE Milestone* comme le premier robot mobile intelligent.
1970	*Lunokhod-1* devient le premier robot qui se pose sur la Lune dans le cadre du programme Lunokhod de l'Union soviétique.
1973	Cincinnati Milacron présente le robot *T3*, qui devient très populaire en industrie.
1978	Le premier robot *PUMA* a été livré à GM par Unimation.
1982	GM et FANUC du Japan signent un accord pour fabriquer les robots *GMFanuc*.
1983	La robotique devient très populaire dans le monde industriel et académique. Plusieurs universités ont commencé à intégrer la robotique dans leurs programmes.
1997	*Sojourner* arrive sur Mars, envoyé par la NASA, il est le premier robot qui atteint la planète rouge.
2000	Le premier robot humanoïde *ASIMO* a été présenté par Honda.
2001	La FDA approuve l'utilisation du robot chirurgicale *« da Vinci »* aux États-Unis.
2008	Universal Robots crée le premier robot collaboratif (cobot) destiné au marché, suivi par Rethink en 2011.
À partir de 2010	Développement de plusieurs robots, véhicules autonomes, drones, *etc.* En 2021, le drone *Ingenuity* a volé pour la première fois sur Mars, dans le cadre de la mission Mars 2020 de la NASA.

Les débuts de la robotique moderne étaient notamment avec les *bras/manipulateurs (Robotic Arms)*. Par la suite, un intérêt aux robots mobiles s'est développé rapidement.

Pour qu'un robot mobile puisse se déplacer dans son environnement, il nécessite des mécanismes de *locomotion* adaptés à ce dernier. Il existe une grande variété de méthodes possibles pour se déplacer, qui dépendent du type d'environnements visé. Ceci fait du choix des moyens de locomotion un aspect crucial lors de la conception d'un robot mobile. En laboratoire, il existe des robots de recherche capables de *marcher*, *sauter*, *courir*, *glisser*, *patiner*, *nager*, *voler* et, bien sûr, *rouler* [11].

En robotique mobile autonome, le but est de concevoir des robots capables de réaliser des tâches de mobilité d'une manière autonome et sans intervention humaine. Un tel robot doit être doté de **capteurs** lui permettent d'observer son environnement ; d'algorithmes de **perception** qui permettent d'analyser les données des capteurs et d'extraire de l'information utile ; d'algorithmes de **prise de décision** qui permettent de sélectionner les actions adéquates à la situation perçue ; et d'**actionneurs** qui permettent au robot de réaliser ces actions et d'interagir avec son environnement.

1.2 Contexte

Ce travail a été réalisé dans le cadre du dispositif des *Conventions industrielles de formation par la recherche (Cifre)*, financé conjointement par l'*Association nationale de la recherche et de la technologie (ANRT)* et l'entreprise *ez-Wheel*. Cette thèse a été préparée à l'université Paris-Saclay au sein du groupe *Méthodes et outils pour les signaux et systèmes (MOSS)* du laboratoire *Sciences et applications des technologies de l'information et de l'énergie (SATIE)*.

La société ez-Wheel[1] (prononcée *easy wheel*) située à *La Couronne 16400* est une entreprise innovante française fondée en 2009 par trois jeunes ingénieurs charentais. Ez-Wheel développe principalement des roues motorisées compactes, hautement intégrées, modulaires et autonomes en énergie. La roue intègre toutes les composantes nécessaires pour un système motorisé, incluant *la roue*, *le moteur*, *le train d'engrenages*, *les batteries*, *l'électronique* et *le système embarqué* qui assure la régulation et la communication avec d'autres périphériques. La figure 1.5 illustre la roue ezW300I de la première génération, avec une vue sur les composantes internes.

Figure 1.5 – La roue ezW300I d'ez-Wheel.

1. ez-Wheel, The Electric Wheel. 135 Route de Bordeaux, 16400 La Couronne, France, ez-wheel.com.

Étant puissantes, compactes et intégrées, ces roues permettent de motoriser n'importe quelle plateforme de transport de lourdes charges. Les utilisateurs peuvent motoriser leurs infrastructures logistiques existantes sans besoin de changer les anciennes plateformes manuelles. Les roues d'ez-Wheel sont déployées dans plusieurs usines allant de l'industrie automobile, à l'industrie agroalimentaire et pharmaceutique, en passant par les entrepôts et les sites de production, *etc.* La figure 1.6 montre des exemples de plateformes motorisées à l'aide des solutions d'ez-Wheel.

Figure 1.6 – Exemples de plateformes motorisées à l'aide des roues autonomes d'ez-Wheel.

L'excellence et l'originalité des produits d'ez-Wheel ont été reconnues dans le monde du business et de l'industrie à travers l'attribution de plusieurs prix et trophées. Parmi ces prix remportés par ez-Wheel, nous citons :

- Lauréat du Concours National d'aide à la création d'entreprises de technologies innovantes - catégorie « émergence ». *Ministère de l'Enseignement Supérieur et de la recherche* ; juin 2009.
- Prix « Grand Public - Transport » Mechatronics Awards. *European Mechatronics Meeting* ; mai 2010.
- Lauréat du Concours National d'aide à la création d'entreprises de technologies innovantes - catégorie « création développement ». *Ministère de l'Enseignement Supérieur et de la recherche* ; juin 2010.
- Lauréat du Concours Européen de l'Entreprise Innovante - catégorie « business - projet en développement ». *Jeune Chambre Économique* ; décembre 2010.
- Nominé aux « Postal Technology International Awards ». *Salon Post-Expo - Stuttgart* ; septembre 2011.
- Lauréat du trophée « PME Bougeons-nous ». *RMC & BFMTV* ; octobre 2011.
- Lauréat du trophée « Embarqué critique ». *4ème Assises de l'embarqué à Paris* ; novembre 2011.
- Lauréat du trophée « Innovation » des éco-industries. *Poitou-Charentes* ; octobre 2014.
- Nominé aux « Handling Awards ». *Salon Motek - Stuttgart* ; octobre 2014.
- Lauréat du « Handling Awards » - catégorie « Automation ». *Salon Motek - Stuttgart* ; octobre 2020.
- Lauréat du « Trophée de la Supply Chain » - catégorie entreprise innovante. *Nuit de la Supply Chain, Paris* ; décembre 2022.

Historiquement, les produits d'ez-Wheel étaient destinés à la réduction de pénibilité dans le transport des lourdes charges. À ce jour, une majorité des plateformes équipées par des roues d'ez-Wheel sont pilotées manuellement, les roues assurent la traction tandis que l'opérateur s'occupe du pilotage et du guidage.

Dans un contexte d'industrie 4.0, l'entreprise développe une nouvelle génération de produits, destinée à des applications en robotique critique, avec des exigences industrielles de sûreté. Étant conçue pour des applications industrielles, cette nouvelle gamme de produits est certifiée conforme à un ensemble de normes orientées sécurité, dont les normes :

- NF EN 61800-5-2 : Entraînements électriques de puissance à vitesse variable — Partie 5-2 : exigences de sécurité ;
- NF EN 61508 : Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité ;
- ISO 3691-4 : Chariots de manutention — Exigences de sécurité et vérification — Partie 4 : Chariots sans conducteur et leurs systèmes.

Cette nouvelle gamme de produits prend en charge également des protocoles de communication sûre tels que le *CANopen Safety*, ce qui permet ces produits, en plus des avantages physiques de compacité et de modularité, d'être facilement interfaçables avec d'autres capteurs compatibles, tels que des LiDARs de sécurité.

Pour pouvoir automatiser des plateformes motorisées en utilisant ces produits, il est nécessaire de développer un système de localisation indoor (en intérieur). Cela impose l'utilisation de plusieurs technologies et le choix de méthodes adaptées permettant d'assurer une précision acceptable en fusionnant les données provenant de plusieurs capteurs multimodaux.

1.3 Problématique de la thèse

Dans le contexte précédemment décrit et dans le but de résoudre le problème de la localisation, nous avons formulé la problématique suivante dans cette thèse :

> *Comment concevoir une solution de localisation robotique en environnements indoor industriels qui soit non intrusive, modulaire, robuste, et peu coûteuse ?*

En effet, la localisation d'un robot industriel mobile dans son environnement d'évolution (*fig.* 1.7) est une étape primordiale pour la conception d'un robot mobile autonome qui réaliserait des tâches utiles. Néanmoins, ce type d'environnements relève plusieurs défis et nécessite donc des systèmes de localisation adéquats qui satisfassent les contraintes exigées en sûreté et en robustesse.

En environnements indoor, les systèmes de localisation doivent être conçus en prenant compte de toutes les contraintes de l'environnement cible. Dans notre cas, les environnements ciblés sont principalement des *sites industriels*, des *grands entrepôts* ou des *usines*. Ces environnements ont la particularité d'être grands avec une présence permanente d'objets *dynamiques* (mouvants), tel que des ouvriers, des robots, des machines, *etc.* Mais aussi des objets *statiques (immobiles) multitemporels*, *c.-à-d.*, des éléments qui restent statiques pendant les déplacements du robot, mais qui peuvent disparaître sur des échelles de temps plus importants. Par exemple, des cartons déposés sur l'environnement peuvent être considérés

comme immobiles durant les déplacements du robot. En revanche, une fois qu'ils sont retirés, il sera nécessaire de mettre à jour la carte de l'environnement qui les considérait comme des points de repère.

Figure 1.7 – Exemple d'un robot évoluant dans un environnement industriel dynamique.

L'objectif de cette thèse est de proposer une solution qui réponde aux contraintes spécifiques définies par l'entreprise partenaire. Ces contraintes comprennent :

- *La non-intrusion*, en évitant toute dépendance d'une infrastructure dédiée ou de modification de l'environnement.
- *La modularité*, afin de pouvoir intégrer la solution sur différents types de plateformes mobiles proposées par l'entreprise.
- *La robustesse*, en assurant une localisation précise et fiable même dans des conditions adverses.
- *Le faible coût*, pour permettre une adoption économiquement viable de la solution.

L'aspect modulaire est très important dans notre contexte, en effet, les travaux présentés dans cette thèse adressent principalement la famille des robots mobiles autonomes (AMRs), mais avec des considérations et des contraintes supplémentaires.

Contrairement aux AMRs, qui sont conçus et dimensionnés pour fonctionner de manière autonome dès le départ, ce travail est destiné à des plateformes qui peuvent être préexistantes (manuelles ou semi-automatiques) et sur lesquelles nous ajoutons des solutions de motorisation d'ez-Wheel. La vaste diversité des plateformes envisageables dans ce contexte signifie que les solutions de localisation que nous proposerons ne doivent pas dépendre des caractéristiques mécaniques et dynamiques de la plateforme, étant donné que celles-ci varient d'une plateforme à l'autre.

La problématique centrale de la thèse consiste ainsi à identifier les méthodes et les techniques les plus appropriées pour réaliser une localisation précise et robuste des robots dans des environnements industriels. Nous avons donc suivi une démarche scientifique que nous trouvons adaptée à ce type de travaux (*fig.* 1.8).

Figure 1.8 – La démarche scientifique suivie pour répondre à la problématique dans notre contexte.

Étant réalisé en contexte Cifre, ce travail vise à proposer un système de localisation adapté aux environnements industriels d'une maturité technologique de niveau 6 (TRL [2] 6), correspondant à un prototype validé en environnement opérationnel réel.

1.4 Organisation du manuscrit

Le manuscrit est structuré en six chapitres. Après l'introduction générale présentée dans ce premier chapitre, nous abordons dans le deuxième chapitre un état de l'art sur la localisation indoor. Nous définissons le problème de la localisation et nous classifions les méthodes de localisation en catégories méthodologiques et technologiques. Dans ce chapitre, nous explorons la littérature scientifique pour identifier les capteurs, les technologies et les méthodes de fusion multicapteur utilisés dans ce type de systèmes.

Le troisième chapitre se concentre sur l'utilisation du LiDAR pour la localisation indoor. Nous introduisons d'abord les différentes approches utilisées pour la localisation basée sur LiDAR, puis nous présentons et validons la plateforme expérimentale *« SmartTrolley »*. Et enfin, nous exposons notre contribution à la localisation relative basée sur la fusion multi-LiDARs.

Le quatrième chapitre traite de la localisation par vision verticale. Nous commençons par une introduction à la vision par ordinateur et aux outils mathématiques utilisés. Ensuite, nous définissons la notion d'odométrie visuelle et nous montrons la pertinence de l'utilisation du plafond comme espace d'observation pour la localisation, en nous appuyant sur des exemples de méthodes de localisation par vision verticale tirés de la littérature scientifique. Enfin, nous présentons notre contribution à la localisation par caméra verticale, dans laquelle nous exploitons l'approche directe de l'estimation visuelle de mouvements à travers la méthode *DSO (Direct Sparse Odometry)*.

Dans le cinquième chapitre, nous proposons un jeu de données multicapteur comprenant des données provenant d'une caméra verticale. Nous réalisons une expérimentation et nous concevons et collectons un jeu de données composé de plusieurs séquences de trajectoires complexes. Ce jeu de données est réalisé dans un environnement industriel réel et comprend des données en provenance de plusieurs capteurs. Ainsi, ce jeu de données nous offre l'opportunité d'évaluer à la fois notre méthode d'odométrie visuelle verticale, les méthodes classiques à base de caméra frontale, et les méthodes de localisation qui reposent sur la fusion multicapteur. Dans ce chapitre, nous présentons un état de l'art des jeux de données existants et nous expliquons les motivations derrière notre choix de construire un énième jeu

2. TRL pour *Technology Readiness Levels* : un système de mesure utilisé pour évaluer le niveau de maturité d'une technologie particulière. Le niveau TRL 1 étant le moins mature et le TRL 9 le plus mature.

de données. Nous décrivons ensuite l'expérimentation, la plateforme, les capteurs utilisés et le mode d'acquisition. Nous terminons ce chapitre par une analyse quantitative et qualitative des données acquises.

Nous clôturons ce travail par un sixième chapitre consacré à la conclusion générale, où nous dressons un bilan de nos travaux et exposons nos perspectives futures. Enfin, nous fournissons une série d'annexes sur les travaux en cours et sur les détails techniques de ce travail.

CHAPITRE 2

État de l'art sur la localisation indoor

2.1 Introduction

En robotique mobile autonome, nous pouvons définir le cycle global de fonctionnement d'un robot en trois étapes : *« percevoir, décider et agir »* (figure 2.1) [11]. Cette architecture permet de subdiviser le système robotique en sous-systèmes chargés d'accomplir des tâches spécifiques. Ainsi, *la perception* englobe les tâches d'acquisition, de filtrage et d'extraction d'informations pertinentes à partir des données brutes des capteurs. Ces informations sont ensuite utilisées pour *la localisation et la cartographie*, ainsi que pour la détection d'obstacles et la *planification* de la trajectoire du robot. Une fois que la trajectoire globale est planifiée, elle est transmise au *contrôleur de mouvements*, qui est chargé d'exécuter cette trajectoire le plus fidèlement possible tout en prenant en compte les éventuels obstacles qui se dresseraient sur son chemin.

Il est possible de classifier les robots mobiles destinés aux environnements industriels en deux catégories principales :

- Véhicules guidés automatisés *(AGV - Automated Guided Vehicles)* ;
- Robots mobiles autonomes *(AMR - Autonomous Mobile Robots)*.

D'une part, les véhicules guidés automatisés (figure 2.2a) sont généralement dépendants d'une infrastructure dédiée à la navigation. Ils sont par conséquent guidés par cette infrastructure et contraints de suivre les chemins et les trajectoires préinstallés sur l'environnement. Dans la pratique, les infrastructures des AGVs peuvent être mises en œuvre grâce à des techniques de *guidage actif* [12], en utilisant par exemple, des fils placés sous la surface du sol ; ces fils émettent des signaux électromagnétiques que le robot détecte et suit, d'une manière similaire à un robot suiveur de ligne. Alternativement, les infrastructures peuvent être basées sur des techniques de *guidage passif* [13, 14], à l'aide de bandes magnétiques installées au sol, d'étiquettes RFID, de marquage visuel au sol, *etc.*

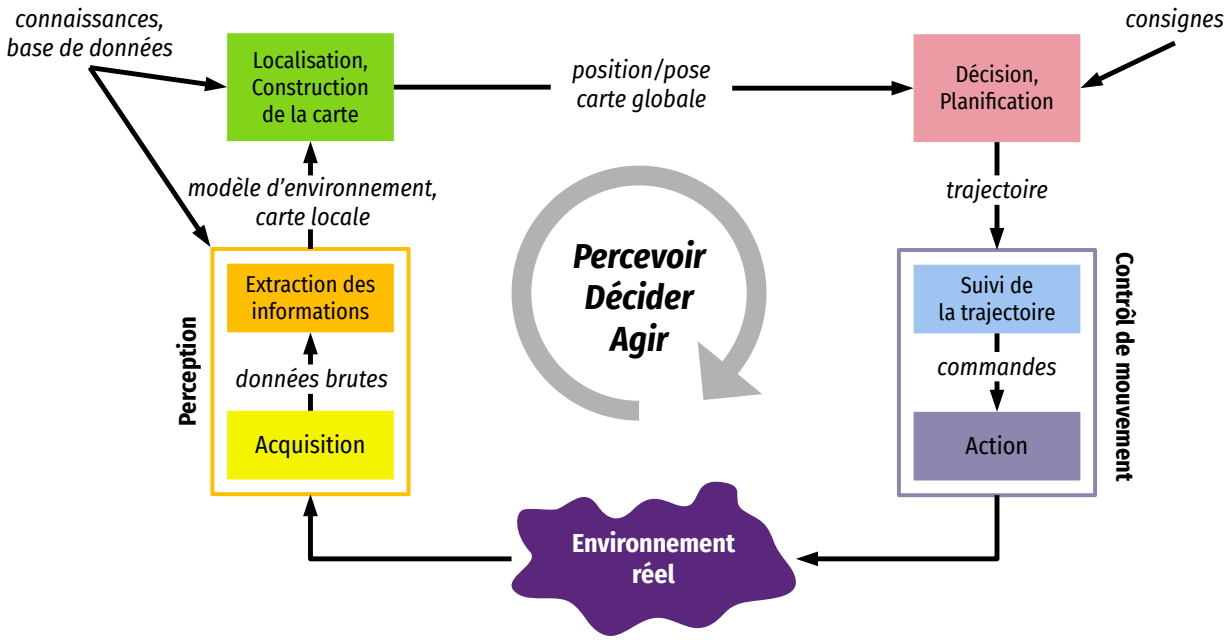
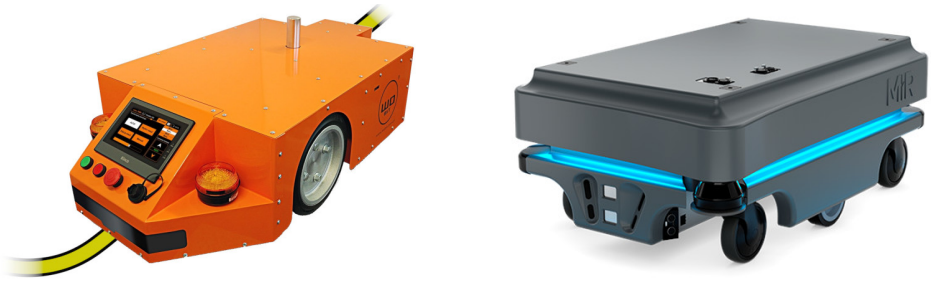


Figure 2.1 – Le cycle percevoir-décider-agir (adaptée de [11]).



(a) L'AGV Mobot AGV EcoRunner **(b)** L'AMR MIR200

Figure 2.2 – Exemples des principaux types des robots mobiles indusriels. Les *véhicules guidés automatisés (AGV - Automated Guided Vehicles)* et les *robots mobiles autonomes (AMR - Autonomous Mobile Robots)*.

D'autre part, les robots mobiles autonomes (figure 2.2b) nécessitent peu ou pas de modifications de l'environnement. L'AMR utilise ses capteurs pour percevoir l'environnement. Ensuite, il utilise cette perception pour modéliser et comprendre la scène, ce qui lui permet d'estimer son *égo-mouvement* (se localiser), de détecter les obstacles statiques ou mobiles qui l'entourent, et de naviguer dans son environnement. Les avancées technologiques des deux dernières décennies dans la production de capteurs et de calculateurs de plus en plus performants ont suscité un intérêt croissant pour le développement des AMRs. Leur flexibilité et leur facilité de déploiement en font une solution attrayante pour de nombreuses applications industrielles.

2.2 La localisation

La localisation en robotique est la tâche d'estimer la pose [1] d'un robot mobile dans son environnement. Le terme *« pose »* désigne en robotique la position (x, y) et orientation ϕ d'un robot dans l'espace, dans le cas bidimensionnel (2D). Dans son abstraction la plus fondamentale, la localisation est *un problème de changement de repère*, entre un référentiel mobile (O_R, X_R, Y_R) attaché au robot et un autre fixe (O_W, X_W, Y_W) attaché à notre espace, représenté par un plan 2D dans cet exemple ; la figure 2.3 illustre cela. Le but d'un système de localisation est donc de déterminer la transformation (x, y, ϕ) entre les deux référentiels.

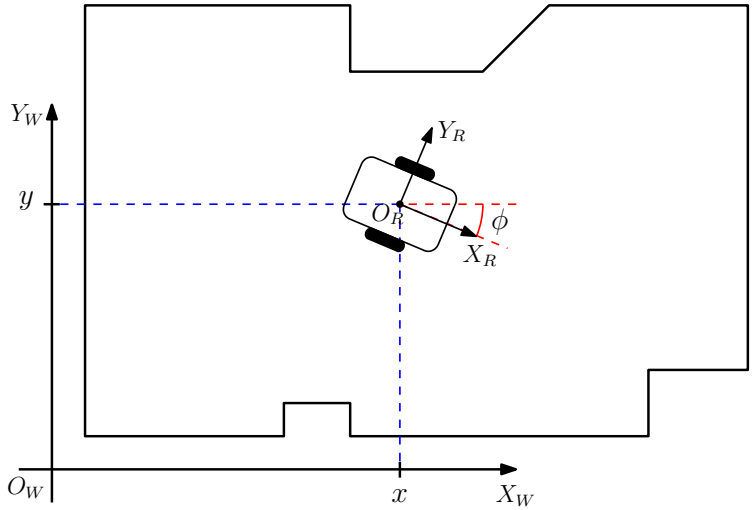


Figure 2.3 – Illustration du problème de la localisation en 2D. Le but est de déterminer la transformation (x, y, ϕ) entre le repère global (O_W, X_W, Y_W) et le repère mobile lié au robot (O_R, X_R, Y_R).

Traditionnellement, le problème de la localisation est divisé en deux catégories principales : la localisation indoor (en environnement intérieur) et la localisation outdoor (en environnement extérieur). La localisation outdoor est principalement étudiée dans le contexte des véhicules autonomes et des robots conçus pour opérer dans des environnements extérieurs (tels que les robots urbains, les robots agricoles, les drones, *etc.*).

Quant à la localisation indoor, elle est essentiellement abordée dans le contexte des robots mobiles industriels, des robots de service, des robots domestiques évoluant dans des espaces clos, *etc.* La localisation indoor a fait l'objet de nombreux travaux de recherche, et la majorité des solutions proposées dans la littérature scientifique utilisent les technologies existantes (telles que la vision, les scanners laser, le WiFi, l'Ultra-Wide Band (UWB), le Bluetooth, *etc.*) afin d'apporter des solutions au problème. Ces technologies sont ainsi exploitées pour développer des approches de localisation adaptées à des cas d'utilisation spécifiques.

1. Dans cet exemple, nous parlons d'une pose en plan bidimensionnel (2D), ce qui donne trois degrés de liberté (x, y, ϕ), deux pour la position et un pour l'orientation. Le principe reste le même dans le cas d'une localisation en espace tridimensionnel (3D), à la différence que la pose 3D a 6 degrés de liberté (DDL), trois pour la position et trois pour l'orientation.

Avant de pouvoir se localiser, il est essentiel de définir une carte de référence (avec un référentiel attaché à celle-ci) dans laquelle le robot cherchera à se positionner. La représentation de cette carte peut prendre diverses formes, plus ou moins abstraites, qui dépendent généralement des capteurs et des techniques utilisés. Par exemple, les positions des bornes WiFi dans un environnement peuvent constituer une carte pour un robot qui utilise les intensités des signaux WiFi (RSSI) pour estimer sa position.

Toutefois, en pratique, il peut s'avérer difficile de se disposer préalablement d'une carte de l'environnement. Dans de tels cas, la tendance dominante consiste à construire la carte et à effectuer la localisation en même temps, c'est ce qu'on appelle dans le jargon *« Localisation et cartographie simultanées »*, plus connue sous l'acronyme *SLAM* pour *Simultaneous Localization and Mapping* [15, 16].

2.3 Taxonomie du problème de la localisation

Le problème de la localisation peut être appréhendé de diverses manières en fonction du contexte d'utilisation et du point de vue adopté. Cela donne lieu à l'émergence d'une multitude de catégories possibles. Selon THRUN et al. [17], le problème de la localisation peut être divisé selon quatre critères principaux : le *type de localisation*, le *type d'environnement*, la *réactivité de l'approche* et l'*échelle d'application*. La figure 2.4 présente cette classification en détaillant les sous-catégories associées à chaque critère.

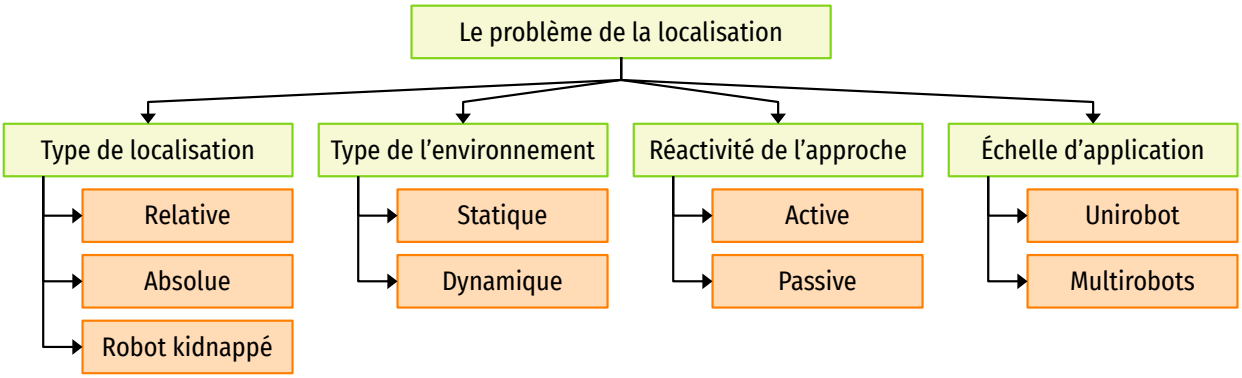


Figure 2.4 – Taxonomie du problème de la localisation robotique.

2.3.1 Classification par type de localisation

Classiquement, le problème de localisation est divisé en trois types qui répondent à différentes questions et contraintes, à savoir : la localisation *relative*, *absolue* et le *problème du robot kidnappé*.

2.3.1.1 Localisation relative

La *localisation relative*, également appelée *localisation locale* (voir l'illustration 2.5), consiste à estimer de manière progressive la pose du robot en répondant à la question suivante : *« connaissant ma pose précédente, où suis-je maintenant ? »* Dans ce cas, la configuration initiale (la pose initiale) est supposée connue, et le robot doit suivre, de manière récursive, l'évolution de sa pose au fil du temps.

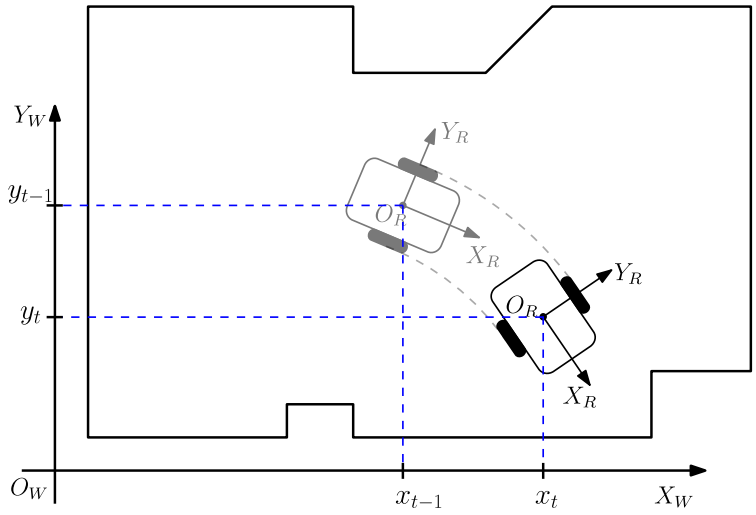


Figure 2.5 – Illustration du problème de la localisation relative. Le but est de déterminer la pose (x, y, ϕ) à partir de la pose précédente.

Ce problème a été abordé par plusieurs approches, l'une des plus utilisées, qui est devenue le standard de facto de l'estimation d'état en robotique est *l'approche probabiliste* [17]. Les méthodes probabilistes les plus connues comportent la famille des filtres de Kalman, avec d'un côté le filtre de Kalman *(KF - Kalman Filter)* classique destiné aux systèmes linéaires gaussiens. Et de l'autre côté, ses variantes, telles que le filtre de Kalman *« étendu » (EKF - Extented Kalman Filter)* et le *« sans parfum » (UKF - Unscented Kalman Filter)*, qui sont utilisés respectivement pour les systèmes gaussiens légèrement et fortement non-linéaires [18].

Une autre méthode probabiliste qui a connu un grand succès en robotique est le filtre particulaire *(Particle Filter)*, qui définie un cadre d'estimation d'état à base de la méthode de *Monte-Carlo* [19]. En dehors des approches probabilistes, d'autres formulations basées sur la théorie des fonctions de croyance (théorie de Dempster-Shafer), les erreurs bornées, et le calcul d'intervalles ont été proposées dans la littérature scientifique [20, 21].

2.3.1.2 Localisation absolue

La *localisation absolue*, dite aussi *localisation globale* (voir l'illustration 2.6), vise à répondre à une question plus générique que la précédente, à savoir : *« où suis-je ? »* Dans ce cas, le robot cherche à se localiser avec ou sans connaissance de sa pose initiale.

Dans la pratique, le problème de la localisation absolue peut être abordé de deux manières principales. La première consiste à utiliser un système qui est capable d'estimer en permanence la pose du robot dans le repère global, *c.-à-d.*, de manière absolue. La deuxième approche implique un système hybride qui combine des techniques de localisation relatives et absolues. Dans l'approche hybride, une phase initiale du système se concentre sur l'estimation de la pose initiale de manière absolue. Une fois cette pose initiale estimée, le système bascule vers une localisation relative par rapport à cette pose de départ.

Différentes approches ont été utilisées dans la littérature scientifique pour résoudre ce problème. Par exemple, la méthode basée sur les erreurs bornées et le calcul d'intervalles, présentée par HANEBECK et al. [20], permet d'estimer la pose initiale et l'utilise ensuite pour initialiser un algorithme de localisation relative. D'autres méthodes ont abordé ce problème en utilisant des approches probabilistes, basées notamment sur des chaines de MARKOV [22] et des méthodes à base de Monte-Carlo [23].

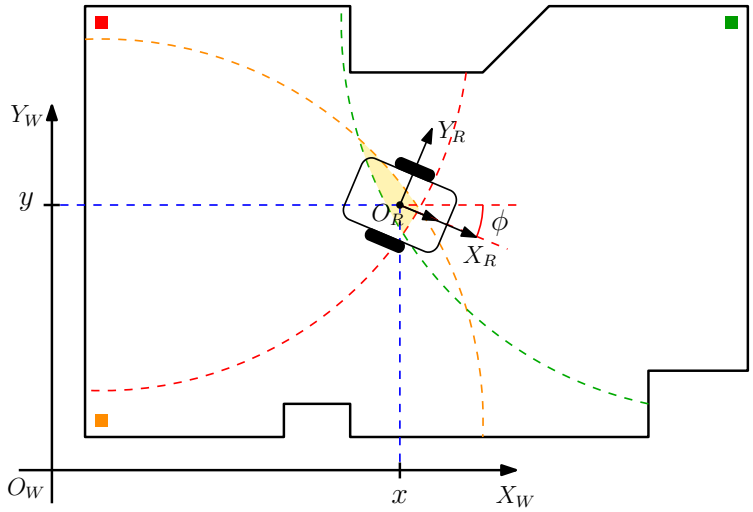


Figure 2.6 – Illustration du problème de la localisation absolue. Le but est de déterminer la pose (x, y, ϕ) directement dans le repère global. Par exemple, en mesurant les distances à des repères connus dans l'environnement, puis en réalisant une trilatération pour déterminer la position du robot par rapport à ces repères.

2.3.1.3 Le problème du robot kidnappé

Par souci d'exhaustivité, nous mentionnent également le problème du *robot kidnappé*, appelé parfois le problème de *relocalisation*. Ce dernier peut être considéré comme une variante plus complexe du problème de la localisation absolue [17].

Nous pouvons formaliser ce problème de la manière suivante : pendant son évolution, le robot est soudainement téléporté vers un autre endroit. Dans ce cas, le robot doit d'abord être capable de « savoir » qu'il a été déplacé puis, de se « relocaliser ». Dans ce type de problèmes, la question posée est : *« comment puis-je me relocaliser si on me téléporte ? »*.

Plusieurs travaux ont abordé ce problème, notamment des approches probabilistes basées sur le filtre à particules [24, 25], ainsi que d'autres approches basées sur l'analyse d'intervalles [26], entre autres.

2.3.2 Classification par type d'environnement

En localisation, le choix des systèmes, des méthodes, des capteurs et des technologies est fortement influencé par l'environnement dans lequel ils sont utilisés. Les contraintes exigées par l'environnement doivent être prises en compte lors de l'instrumentation des plateformes robotiques et aussi lors du développement des algorithmes de localisation. En fonction des objets présents dans l'environnement, nous pouvons distinguer deux types : les environnements *statiques* et les environnements *dynamiques*.

2.3.2.1 Environnements statiques

Lorsqu'on suppose un environnement statique, cela signifie que, dans l'espace d'évolution du robot, ce dernier *est le seul objet susceptible de bouger*, tandis que tous les autres objets présents dans l'environnement sont *fixes et immobiles*.

De nombreuses approches de localisation reposent sur des hypothèses qu'impliquerait un environnement statique. L'hypothèse de Markov, largement utilisée dans les méthodes probabilistes en est l'exemple [11, 17].

L'hypothèse de Markov stipule que l'état actuel du robot contient toutes les informations nécessaires pour prédire son état futur, indépendamment des états précédents. De même, lorsqu'on suppose un environnement statique, cela signifie que les objets de l'environnement ne changent pas au fil du temps. Ces deux hypothèses sont équivalentes d'un point de vue d'environnement, car dans un environnement statique, l'état actuel du robot suffit (en conditions idéales pour tout autre facteur) pour prédire son état futur, parce qu'il n'y a pas de changements à prendre en compte dans l'environnement (*c.-à-d.*, nous n'avons pas besoin de tout l'historique des mesures, de la carte, *etc.*).

L'hypothèse de Markov, même qu'elle n'est pas valable dans des conditions réelles, permet de simplifier grandement le raisonnement, le suivi et la planification. Ainsi, elle offre une approximation qui continue d'être extrêmement populaire en robotique mobile [11].

2.3.2.2 Environnements dynamiques

Dans certains cas, la présence d'objets mobiles dans l'espace d'évolution du robot nécessite de considérer que l'environnement n'est pas statique (figure 2.7). Dans de telles situations, le robot doit prendre en compte tous les objets en mouvement afin d'éviter de biaiser l'estimation de sa pose.

Dans la littérature scientifique, ce problème a principalement été abordé à travers deux types d'approches. D'une part, il existe des approches basées sur le filtrage [22, 27], où les données de perception sont filtrées pour réduire l'influence des objets mobiles sur l'estimation. Des techniques d'estimation robuste (de type RANSAC) peuvent être utilisées pour réaliser ce filtrage, dans ce cas, les mesures correspondantes aux objets mobiles sont considérées comme des données aberrantes *(outliers)* qui ne permettent pas de décrire l'égo-mouvement du robot.

Et d'autre part, il existe des approches basées sur l'identification et le suivi des objets mobiles [28, 29]. Dans les deux cas, les mesures associées aux objets mobiles sont exclues des données utilisées pour l'estimation de la pose, car ils ne font pas partie de la carte de l'environnement.

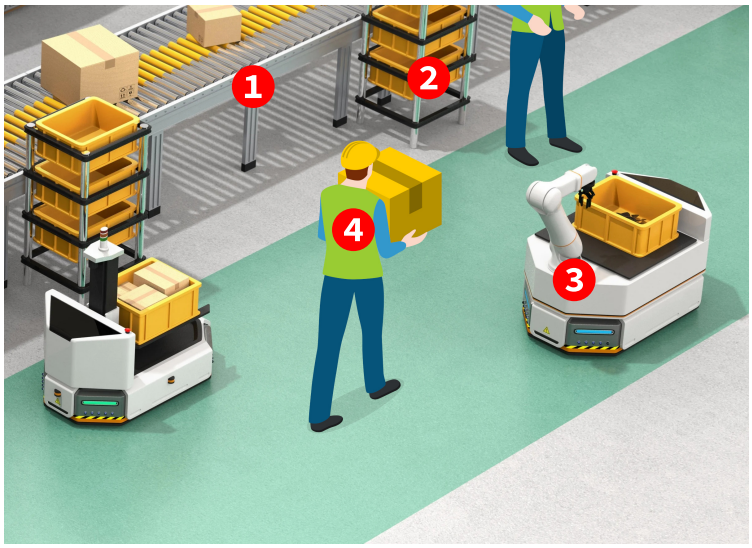


Figure 2.7 – Illustration d'un robot évoluant dans un environnement dynamique en présence : (1) d'objets statiques permanents, (2) d'objets statiques temporaires, (3) d'objets dynamiques prédictibles (autres robots), et (4) d'objets dynamiques imprédictibles (humains).

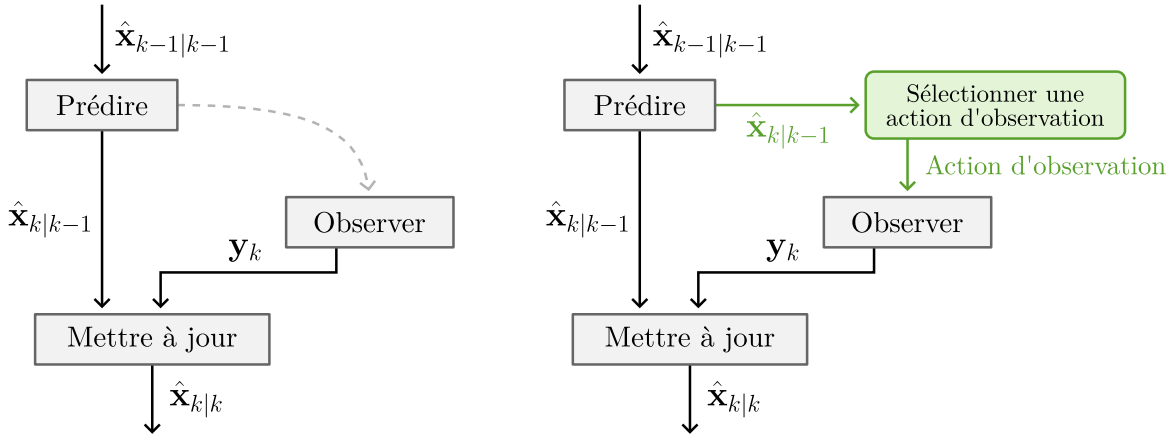
2.3.3 Classification selon la réactivité de l'approche

Un système de localisation *actif* est un système qui a la capacité de modifier la trajectoire du robot afin d'améliorer sa localisation [30, 31]. Ces systèmes guident le robot pour changer son point de vue d'observation pour chercher des points d'intérêt informatifs dans l'environnement ou de réduire l'incertitude liée à sa pose estimée. La figure 2.8b illustre un exemple conceptuel d'un système actif de localisation.

Dans la plupart des approches, les systèmes de localisation proposés sont plutôt *passifs*. Dans ce cas, le système de localisation estime la pose juste en observant l'environnement, et sans pouvoir modifier la trajectoire du robot. L'illustration 2.8a présente un exemple d'un système passif de localisation à base de filtrage.

2.3.4 Classification selon l'échelle d'application

Une autre classification des systèmes de localisation peut être basée sur leur échelle d'application. Certains systèmes [33] sont conçus pour localiser plusieurs robots de manière centrale ou collaborative, ce qui les qualifie de systèmes de localisation *multirobots*. Dans cette catégorie, les robots collaborent les uns avec les autres afin d'affiner les estimations de la pose de chacun.



(a) Localisation passive **(b)** Localisation active

Figure 2.8 – Illustration de la différence conceptuelle entre les systèmes passifs et actifs de localisation. À gauche, un système passif classique à base de filtrage. À droite, un système actif, capable d'actionner le robot pour améliorer l'observation (adaptée de [32]).

La majorité des approches de localisation décrites dans la littérature scientifique sont plutôt des systèmes *unirobots*, également appelés systèmes *égocentrés*. Dans cette catégorie, un robot cherche à estimer sa pose uniquement de son propre point de vue. Par conséquent, le robot n'est pas dépendant des autres robots présents dans l'environnement. Ces derniers ne contribuent pas directement aux données que le robot utilise pour estimer sa pose.

2.4 Technologies de localisation

Plusieurs méthodes ont été développées pour résoudre le problème de localisation en utilisant différentes technologies existantes telles que la vision, les scanners laser, le WiFi, l'Ultra-Wide Band (UWB), le Bluetooth, et bien d'autres. Le choix de la technologie à utiliser dépend fortement des cas d'usages ciblés et des objectifs fixés. Les technologies adaptées à un système de localisation dans un atelier de quelques dizaines de mètres carrés ne seront pas forcément appropriées pour une usine beaucoup plus vaste s'étendant sur plusieurs centaines de mètres carrés. On désigne cela par *l'effet variation d'échelle (scalability)* de l'environnement.

Lors de la conception d'une solution industrielle de localisation robotique, nous avons besoin de prévoir les éventuelles modifications que nous devons apporter à l'environnement, ainsi que les capteurs à utiliser, qu'ils soient *passifs* ou *actifs*. Cela nous permet d'anticiper des problèmes tels que :

- Les coûts liés à l'installation et à la maintenance ;
- La complexité du déploiement et de la mise en service ;
- Les problèmes d'interférence liés aux capteurs ;
- La non-adéquation du capteur à l'application cible ;
- Le niveau de résilience aux pannes nécessaire en fonction des applications ciblées, *etc.*

Ainsi, de ce point de vue, nous pouvons diviser les solutions en deux catégories : celles qui *nécessitent une infrastructure dédiée (infrastructure-based solutions)*, et celles qui *n'en nécessitent pas (infrastructure-less solutions)*. De plus, au sein de chaque catégorie, il est important de prendre en compte la nature du capteur utilisé, qu'il soit *passif* ou *actif*, afin d'identifier les problèmes potentiels associés à ce type de capteur et d'anticiper les limites d'une solution reposant sur celui-ci. Cette approche nous permettrait ensuite de choisir la démarche technologique et scientifique appropriée pour résoudre le problème de la localisation indoor en prenant en compte les contraintes des environnements industriels.

2.4.1 Solutions avec infrastructure

Cette catégorie regroupe les méthodes qui nécessitent la modification ou la collaboration de l'environnement avec le robot afin de localiser ce dernier. Ce type de solutions impose donc des coûts supplémentaires liés à la mise en service et la maintenance de l'environnement.

Ces solutions peuvent soit collaborer avec l'environnement activement ou passivement, d'où la sous-classification selon les types des capteurs, soit *actifs* ou *passifs*.

2.4.1.1 Capteurs passifs

Les capteurs passifs sont des capteurs qui ne reposent pas sur l'émission de radiation, d'onde ou de lumière pour fonctionner, par conséquent, ils ne font qu'observer l'environnement et les phénomènes physiques qui s'y passent. L'utilisation des capteurs passifs peut-être intéressante pour éliminer les potentiels problèmes d'interférence notamment dans le cas multirobot.

Suivi par vision assistée Plusieurs solutions pour la localisation indoor reposent sur l'installation de points d'intérêts visuels sur l'environnement, sous forme de codes bars [34, 35], ou d'une manière générale, de points visuellement reconnaissables. Les codes bars placés dans l'environnement sont identifiés et localisés à l'aide d'une caméra et leurs positions sont utilisés par la suite comme repères pour calculer la position du robot.

Par exemple, le système proposé par Nazemzadeh et al. [35] utilise un ensemble de QrCodes placés sur le sol pour assister l'algorithme de vision (figure 2.9). Ainsi, un robot équipé d'une caméra orientée vers le sol peut naviguer en utilisant son odométrie pour prédire sa pose et les emplacements des QrCodes la corriger. La fusion dans le système de Nazemzadeh et al. a été réalisée par un filtre H_∞ étendu, qui peut être considéré comme une forme spéciale plus générique du filtre de Kalman étendu.

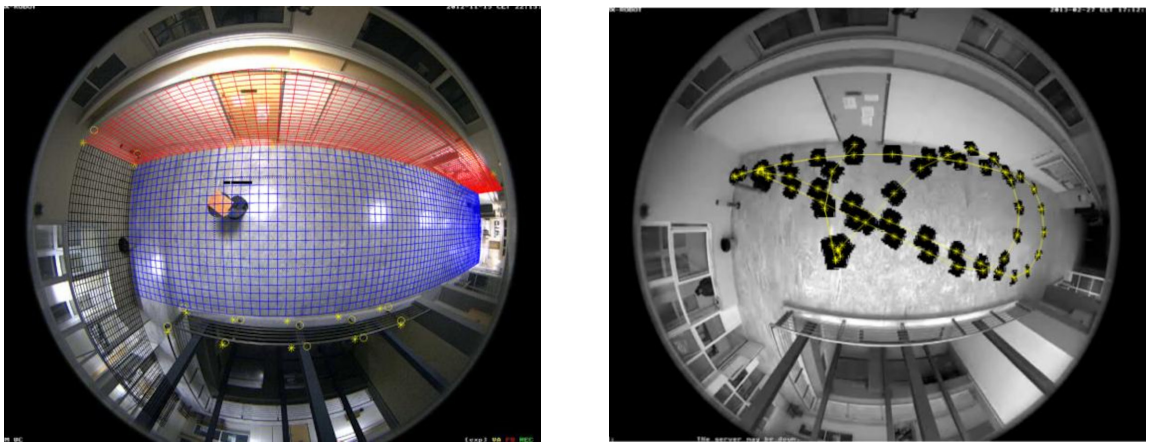
Suivi par vision centralisée Certaines solutions de localisation utilisent une caméra avec un objectif à grand angle, ou un réseau de caméras, fixées sur l'environnement pour observer la scène. Dans ce cas, des algorithmes de vision adéquats qui permettent la reconnaissance et le suivi des robots dans la scène sont utilisés [36, 37].

Le système proposé par Delibasis et al. [37] permet le suivi des déplacements d'un robot à l'aide d'une caméra stationnaire placée sur le plafond et équipée d'un objectif œil de poisson. Le haut du robot est coloré d'une couleur connue, puis, l'algorithme de localisation

Figure 2.9 – Un système de localisation visuelle à base de QrCodes placés au sol (tiré de [35]).

réalise une segmentation de l'image pour extraire l'empreinte du robot. La position du robot est ainsi calculée dans le repère sphérique de la caméra, et transformée ensuite au repère euclidien de l'environnement à l'aide d'une calibration préalable du modèle inverse de la caméra œil de poisson (figure 2.10).

Étant centralisées, ces approches nécessitent une communication entre le système central de localisation et les robots présents dans l'environnement. Cela permet au système central de fournir à chaque robot sa pose estimée, et éventuellement, les poses des autres robots.



(a) L'étalonnage du modèle fisheye **(b)** Une trajectoire estimée d'un robot

Figure 2.10 – Un système de localisation à base de vision centralisée (tiré de [37]).

2.4.1.2 Capteurs actifs

Dans cette catégorie, nous listons les capteurs actifs utilisés dans des solutions de localisation indoor, dans le contexte des systèmes à infrastructure. Un capteur est dit actif lorsqu'il *émet de l'énergie dans l'environnement* dans le but de transporter de l'information

ou de réaliser des mesures. Cette énergie peut être de différentes natures, par exemple, sous forme de rayonnements électromagnétiques (laser, radar, technologies radiofréquences, *etc.*) ou sous forme d'ondes mécaniques (ondes sonores, ultrason, *etc.*).

Technologies radiofréquences D'une manière générale, dans les solutions à base de technologies radiofréquences, des bornes sans-fils sont introduites dans des endroits fixes de l'environnement du robot. Ces bornes permettent par la suite de localiser, ou de contribuer à la localisation du robot.

D'une manière générale, les méthodes de localisation à base de radiofréquences mesurent l'une des trois suivantes caractéristiques du signal [38] :

- L'angle d'arrivée du signal *(AoA - Angle of Arrival)* ;
- Le temps de vol du signal (via le temps d'arrivée *(ToA - Time of Arrival)* ou le décalage de temps d'arrivée *(TDoA - Time Difference of Arrival))* ;
- L'indication de la puissance du signal reçu *(RSSI - Received Signal Strength Indication)*.

Pour les méthodes à base d'angle d'arrivée [39], la technique de *multiangulation*[2] est généralement utilisée. Les méthodes à base de temps de vol utilisent la technique de *multilatération* [40]. Et les méthodes à base de l'indication de la puissance du signal reçu (RSSI) utilisent des techniques de *proximité* [41], ou d'*empreinte énergétique (fingerprinting)* [42]. Il existe également dans la littérature scientifique des méthodes hybrides qui utilisent des combinaisons de ces techniques [43].

La plupart des méthodes de localisation à base de radiofréquences sont en réalité des *systèmes de positionnement*, *c.-à-d.*, ils permettent d'estimer la position (en 2D ou 3D), mais pas l'orientation. Pour des applications en robotique mobile, la connaissance de l'orientation est primordiale. Ainsi, des systèmes de localisation à base de RFs font généralement recours à d'autres capteurs pour estimer ou améliorer l'estimation de l'orientation.

Un exemple d'une telle méthode a été proposé par Dobrev et al. [44]. Dans un système qui utilise des matrices d'antennes placées sur le robot et sur l'environnement, Dobrev et al. ont réalisé une fusion des mesures en provenance de ces antennes avec les données d'un *inclinomètre* pour estimer l'orientation du robot dans l'espace 3D.

Un autre exemple peut être trouvé dans le système proposé par Choi et al. [45], qui ont utilisé des tags RFID placées sur l'environnement. En détectant ces tags, le robot parvient à se localiser grossièrement en absolu. Étant grossière et incertaine, cette première estimation est raffinée ensuite à l'aide de sonars utilisés pour construire une carte locale et la corréler à une carte globale connue préalablement, permettant ainsi de réduire les incertitudes et d'apporter une correction à l'estimation initiale de la pose.

Communication par lumière visible La technologie de la communication par lumière visible (VLC) a été également utilisée à des fins de localisation indoor. De nombreuses solutions de ce genre ont été proposés dans la littérature scientifique [46, 47]. Dans ce type d'approches, les LEDs présentes dans l'environnement pour l'éclairage sont pilotés par un système qui module la lumière en impulsions de hautes fréquences, ainsi, les impulsions ne

2. La *multiangulation* est le nom générique de la technique communément appelée *triangulation*, cette dernière représente le cas spécial où nous n'avons que trois repères. La même remarque est valable pour la *multilatération* qui constitue le cas générique de la *trilatération*.

seront pas perceptibles par l'œil humain. Ces modulations sont utilisées pour transmettre de l'information, que les robots reçoivent à l'aide de photodiodes ou de caméras, ce qui permet d'identifier chaque lampe permettant par la suite de se localiser par rapport à ces dernières.

Ultrasons Il existe des solutions centralisées de localisation qui utilisent des émetteurs ultrasoniques montés sur l'environnement et des récepteurs au niveau des robots [48].

Le système présenté par De Angelis et al. [49] mesure la position relative d'un nœud mobile par rapport à une grille portable de balises. Cette grille est réalisée à partir d'un ensemble d'émetteurs-récepteurs à ultrasons montés sur un panneau. La grille de balises peut se localiser avec précision en mesurant le *temps de vol (ToF - Time of Flight)* par rapport à un petit ensemble d'ancres placées en positions connues et situées d'une manière que la *ligne de mire (LoS - Line of Sight)* soit toujours garantie entre ces ancres et les balises. Dans le cas où le nœud mobile (monté sur le robot) explore une région qui occulte la ligne de mire avec la grille de balises, la grille de balises elle-même peut se déplacer vers un emplacement convenable. Ensuite, elle recalibre sa propre position en mesurant la position des ancres fixes, ainsi, les nouvelles mesures de la position du nœud mobile peuvent à nouveau être ramenées à la référence fixe.

Par ailleurs, il existe quelques solutions disponibles commercialement pour la localisation indoor à base d'un système ultrasonique centralisé. Un exemple de tel système est le *« Indoor GPS »* proposé par la société estonienne Marvelmind, qui réclame une précision de ±2cm dans les conditions idéales [50]. Ce système, représenté sur la figure 2.11, se compose de balises fixes positionnées dans l'environnement et des balises mobiles intégrées sur les robots. Ces balises émettent des ondes ultrasoniques pour mesurer les distances, et communiquent par radiofréquences sur les bandes de fréquences de 915MHz ou 868MHz pour coordonner et synchroniser les mesures entre balises.

Figure 2.11 – Composantes du système de localisation ultrasonique centralisée *Marvelmind Indoor GPS* (tirée de [50]).

Les solutions centralisées à base d'ultrason permettent une précision acceptable dans les conditions idéales (ligne de mire, pas d'interférence avec d'autres émetteurs ultrasoniques ou du bruit des machines, pas de variations en température de l'air, *etc.*). Néanmoins, ces solutions font généralement des systèmes de positionnement, ainsi, elles ne permettent pas de mesurer l'orientation du nœud mobile sans s'appuyer sur des mesures complémentaires.

En plus, dans certains cas, ces systèmes doivent gérer le partage d'une bande passante limitée entre tous les robots pour éviter les problèmes liés aux interférences. Cela impose donc des limitations supplémentaires en matière de l'extensibilité *(scalability)* de tels systèmes.

2.4.2 Solutions sans infrastructure

Dans cette catégorie, les solutions n'exigent aucune modification de l'environnement *(Infrastructure-less)*. Par conséquent, ces approches offrent des avantages en termes de facilité de déploiement et de réduction des coûts d'installation.

2.4.2.1 Capteurs passifs

Localisation à l'estime (Codeurs, IMU) Dans la localisation à l'estime *(Dead-reckoning)*, la pose est estimée incrémentalement en utilisant des *capteurs proprioceptifs*. La méthode la plus simple est d'utiliser les codeurs de roues, qui permettent de mesurer les vitesses de rotation des roues. En utilisant le modèle cinématique du robot, l'intégration de ces vitesses au fil du temps donne l'*odométrie*. Néanmoins, à cause des glissements des roues sur le sol, des déformations des roues, ainsi que d'autres facteurs externes, la trajectoire estimée par l'odométrie peut rapidement diverger de la vraie trajectoire [11].

La centrale inertielle *(IMU - Inertial Measurement Unit)* est un capteur embarquant un *accéléromètre* qui mesure les accélérations linéaires, un *gyromètre*[3] qui mesure les vitesses angulaires, et parfois, un *magnétomètre* qui mesure le champ magnétique terrestre permettant ainsi de déduire l'orientation par rapport au nord magnétique. Nous précisons que le magnétomètre habituellement intégré dans les IMUs est un capteur *extéroceptif*, car il mesure une quantité externe, en l'occurrence, le champ magnétique terrestre. L'IMU peut-être utilisée pour l'estimation des poses en intégrant les mesures dans un modèle dynamique du robot [51, 52].

Dans les deux cas, codeurs de roues et IMUs, le manque d'une correction externe (avec des *capteurs extéroceptifs*) fait que la localisation à l'estime diverge toujours de la réalité [11, 51].

Ce type de capteurs proprioceptifs a l'avantage de fournir des informations à hautes fréquences d'échantillonnage[4] qui permettront de réaliser des estimations rapides sur l'état du robot. Ainsi, ces capteurs sont communément utilisés comme source de données de prédiction dans les systèmes de localisation par fusion multicapteur [53, 54].

Caméra monoculaire ou stéréoscopique L'utilisation des caméras pour la localisation a toujours été un sujet d'actualité, mais dernièrement, il suscite de plus en plus d'intérêt. Ceci peut être expliqué d'un côté par le coût très abordable de ce type de capteurs. Et

3. En langue française, il y a une distinction entre le terme *gyromètre* et le terme *gyroscope*. Le *gyromètre* (*Rate Gyroscope* en anglais) est le capteur qui permet de mesurer la rotation instantanée *(vitesse angulaire)* de son boitier. Le *gyroscope* est le capteur qui mesure la rotation absolue *(position angulaire)* de son boitier dans l'espace.

4. D'une manière générale, les fréquences d'échantillonnage des codeurs et des IMUs varient dans la plage de 10Hz à 10KHz, selon le type et la qualité du capteur.

d'un autre côté par les avancées considérables en performance et en capacité de calcul des microprocesseurs modernes, ce qui permet d'implémenter des algorithmes de vision par ordinateur de plus en plus complexes.

Une caméra *monoculaire* observe une scène tridimensionnelle (3D) et produit des images bidimensionnelles (2D). L'information sur les profondeurs des points de l'image est ainsi perdue lors de la projection (3D $\rightarrow$ 2D).

Pour un algorithme de localisation monoculaire, le but est d'estimer les poses de la caméra à partir d'un flux d'images perçues. Ceci peut être réalisé en effectuant un suivi des points d'intérêt sur plusieurs images prises depuis des points de vue différents. Ensuite, la rétroprojection et le suivi sur plusieurs images des points extraits de l'image (en plan 2D) à des points (en espace 3D) sur la scène observée permet d'estimer les *paramètres extrinsèques* (*c.-à-d.*, la pose) de la caméra. En revanche, à cause de l'absence des informations sur les profondeurs, ces estimations issues de la rétroprojection ne sont valables qu'à une échelle près [5].

Les systèmes de vision *stéréoscopique* embarquent deux caméras monoculaires séparées d'une distance connue, appelée *l'entraxe (baseline)*. L'utilisation de deux caméras pour capturer des images synchronisées de la même scène permet d'identifier les mêmes points d'intérêt dans les deux images. Ceci rend possible l'observation directe de l'effet de la *parallaxe* sur ces points d'intérêt, permettant ainsi de calculer les profondeurs des points rétro-projetés lors de l'estimation des positions 3D des points dans la scène [55].

D'un point de vue conceptuel, les méthodes de localisation ou du SLAM visuelles peuvent être divisées en deux catégories, les méthodes *indirectes* et *directes* [56]. Dans les méthodes indirectes, dites aussi méthodes *basées sur des caractéristiques/attributs (Feature-based)*, les images sont pré-traitées pour extraire des points d'intérêt. Cela est fait à l'aide des algorithmes de traitement d'images, notamment les détecteurs et les descripteurs. Ensuite, une optimisation de l'*erreur géométrique* sur les positions des caractéristiques est réalisée pour estimer les mouvements de la caméra [57]. Pour les méthodes directes, les mouvements de la caméra sont estimés directement à partir des données brutes, ou en d'autres mots, les approches directes opèrent directement sur les intensités des pixels sur le flux d'images de la caméra. Dans ce cas, une optimisation de l'*erreur photométrique* est réalisée pour estimer les mouvements de la caméra [58].

Le problème de localisation visuelle a été traité dans la littérature scientifique de plusieurs manières. L'*odométrie visuelle* constitue une variante de ce type de systèmes, elle permet d'estimer les changements relatifs de la pose de la caméra d'une manière incrémentale similaire à celle de l'odométrie des roues. Le terme *odométrie visuelle (VO - Visual Odometry)* a été introduit la première fois par Nister et al. [59] ; depuis, plusieurs approches ont été proposées, que ce soit à base de formulations indirectes [60, 61], directes [62, 63], ou hybrides [64].

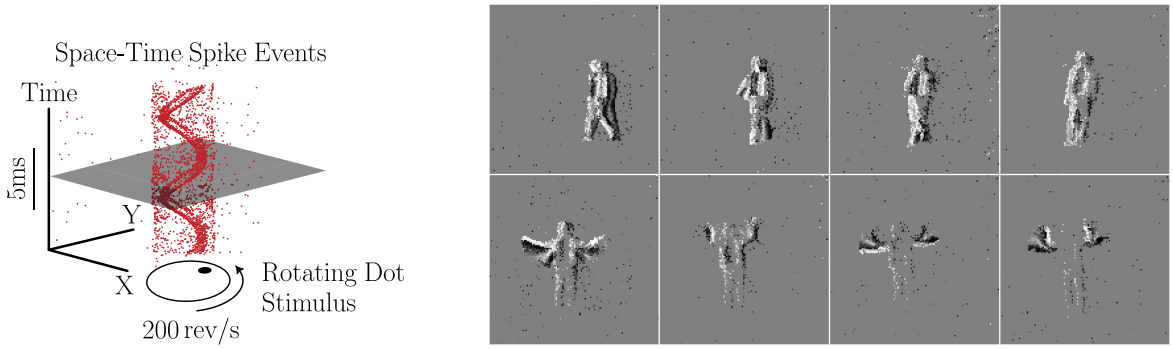
L'autre variante des systèmes de localisation par caméras est le SLAM visuel, ou *vSLAM* pour *Visual SLAM*. Le SLAM visuel peut être vu comme une combinaison de l'odométrie visuelle avec des mécanismes de gestion et de correction de la carte. Un des premiers systèmes vSLAM temps réel a été proposé par Davison [65], depuis, plusieurs autres vSLAMs ont été proposés, parmi les plus connus, nous citons MonoSLAM [66], ORB-SLAM [67, 68],

5. Nous aborderons ce problème en plus de détails dans le chapitre 4 et dans l'annexe A.

RTAB-Map [69] et KimeraSLAM [70] qui sont basés sur des approches indirectes, ainsi que d'autres basés sur des approches directes telles que PTAM [71], LSD-SLAM [72] et LDSO [64], pour ne citer que les plus connus.

Caméra d'événements Les caméras standards acquièrent des images complètes à une fréquence spécifiée. Chaque pixel de ces images représente *l'intensité* acquise pendant un certain temps d'exposition. En revanche, les *caméras d'événements (Event Cameras)*, appelées aussi *capteurs neuromorphiques (Neuromorphic Sensors)*, réagissent aux *changements de luminosité* de la scène de manière *asynchrone* et *indépendante* pour chaque pixel. Ainsi, la caméra d'événements permet d'acquérir un flux d'événements de débit variable, chacun représente un changement d'intensité de la luminosité en échelle logarithmique [73].

La figure 2.12 montre un exemple d'une acquisition à partir d'une caméra d'événements. La sous-figure 2.12a montre comment un stimulus à grande vitesse (le point qui tourne) génère un flux numérique asynchrone et épars d'événements, reflétant les changements rapides dans la luminosité de la scène. La sous-figure 2.12b montre des exemples d'images d'événements acquises lors de la captation d'une scène d'un homme en mouvement.



(a) Principe d'acquisition (adaptée de [74]) **(b)** Exemples d'images d'événements

Figure 2.12 – Exemples d'acquisitions à partir d'une caméra d'événements.

Les caméras d'événements sont des capteurs bio-inspirés qui imitent la rétine humaine. Il existe plusieurs implémentations de caméras d'événements sur silicium, mais le premier produit qui a été mise à disposition dans le commerce est connu sous le nom de *capteur de vision dynamique (DVS - Dynamic Vision Sensor)*.

La caméra d'événements permet d'avoir une *haute définition temporelle* avec seulement des mesures correspondantes aux zones en mouvement et avec des seuils de détection ajustables. En plus, les caméras d'événements ont une *haute plage dynamique (HDR - High Dynamic Range)*, et elles peuvent fournir des acquisitions avec une très basse latence, tout en fonctionnant à basses énergies [73].

Cette technologie a été exploitée dès son apparition pour répondre au problème de la localisation indoor à travers plusieurs applications. Censi et al. [75] ont été les premiers à proposer un système d'odométrie visuelle basé sur une caméra d'événements couplée avec une caméra classique. La caméra d'événements permet seulement de mesurer les mouvements, *c.-à-d.*, pas d'événements émis dans les scènes statiques, en plus, le capteur utilisé dans cette

approche était uniquement d'une résolution de 128×128 pixels. Ces deux limitations ont été surmontées par le couplage de ce dernier avec une caméra CMOS classique. En utilisant les images acquises de la caméra, les auteurs ont proposé une approche pour augmenter la résolution des événements en utilisant les images acquises de la caméra classique. Les événements sont ensuite utilisés pour estimer le déplacement relatif entre les images de basse fréquence. La méthode est formulée dans un cadre probabiliste similaire à la localisation de Markov, dans le sens où chaque événement est utilisé pour définir une fonction de vraisemblance pour pondérer l'estimation actuelle du mouvement relatif [75].

La vision d'événements est une technologie relativement récente qui constitue un sujet de recherche très intéressant, notamment pour les robots de hautes vitesses et les *drones (UAV - Unmanned Areal Vehicules)*. Ainsi, plusieurs méthodes de localisation à base de caméras d'événements ont été proposées dans la littérature scientifique durant les dernières années [73, 74]. Néanmoins, le coût actuel de ce type de caméras ne permet pas une adoption à grande échelle de cette technologie en dehors de la communauté scientifique ou de quelques applications de niche.

2.4.2.2 Capteurs actifs

LiDARs Le LiDAR, acronyme de *Light Detection and Ranging*, soit, *« détection et estimation de la distance par la lumière »* est l'un des capteurs les plus utilisés dans la robotique mobile. Il permet d'obtenir une carte locale d'obstacles sous forme de coordonnées polaires (angles et distances). Les LiDARs émettent de l'énergie, sous forme de radiations infrarouges comprises généralement entre 850nm et 940nm, et mesurent l'énergie réfléchie par les objets présents aux alentours. Par conséquent, des problèmes d'interférences peuvent se poser si d'autres sources infrarouges sont présentes dans l'environnement (*c.-à-d.*, autres LiDARs, la lumière du soleil, *etc.*).

Dans la littérature, un très grand nombre d'approches utilisent des LiDARs pour la localisation, la plupart des approches sont soit à base de filtres [76, 77], ou à base d'appariement de balayages LiDAR *(scan-matching)* [78-80].

Nous aborderons le sujet des LiDARs en plus de détails dans le troisième chapitre, consacré à la localisation par alignement de données multi-LiDARs.

Ultrasons Les capteurs ultrasons sont aussi très communs en robotique mobile, ils permettent d'estimer la distance d'un objet en mesurant le temps de vol d'une onde acoustique (dans le domaine ultrasonique) émise par le capteur et reflétée par l'objet. Plusieurs systèmes de localisation à base d'ultrason ont été proposés dans la littérature scientifique depuis les années 80s [81] et continuent de susciter l'intérêt des chercheurs [82, 83].

2.5 La fusion multicapteur

Dans la plupart des cas, et que ce soit dans des solutions intrusives ou non intrusives, avec des capteurs actifs ou passifs, l'utilisation d'un seul capteur pour estimer la pose du robot ne permet pas d'avoir la précision et la robustesse recherchées. Dans ces cas, nous pouvons

améliorer l'estimation en combinant des données en provenance de plusieurs capteurs afin d'arriver à une estimation plus précise, robuste et donc meilleure que l'information fournie par chaque capteur indépendamment.

2.5.1 Définition

Le *sous-panneau de fusion de données (Data Fusion Subpanel)* du *Joint Directors of Laboratories (JDL)* était l'un des premiers groupe de travail qui se sont intéressé à la question de la fusion de données (ou de l'information) ; ils ont donné en 1987 dans le *« Data Fusion Lexicon »* [84] une première définition de la fusion de données :

> Data fusion is « a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results » [84].

Plusieurs autres définitions, plus ou moins équivalentes, ont été proposées dans la littérature scientifique depuis la fin des années 80s. La majorité de ces définitions mettent l'accent sur des points qui caractérisent la fusion de données tels que les *méthodes* et les *techniques* utilisées ainsi que sur l'aspect *multisource*. Ces définitions partagent le même but commun de *l'aide à la prise de décision* [85].

Boström et al. [86] ont réalisé un excellent travail de synthèse en présentant plus d'une trentaine de définitions prises de la littérature scientifique. En conclusion, Boström et al. proposent une définition pour le *domaine de recherche en fusion de l'information* qui est plus générique et indépendante du domaine d'application :

> « Information fusion is the study of efficient methods for automatically or semi-automatically *transforming information* from different *sources* and different *points in time* into a *representation* that provides effective support for human or automated *decision making* » [86].

Cette définition souligne que la transformation de l'information peut être *automatique* ou *semi-automatique*. Ainsi, elle inclut le cas de la fusion multicapteur dans les systèmes autonomes et le cas des systèmes semi-autonomes avec l'*humain dans la boucle (HITL - Human-in-the-loop)*.

La définition de Boström et al. précise en outre que le domaine de la fusion de données ou de l'information concerne principalement *la transformation de l'information*. Ce terme générique couvre toutes les manières possibles de combinaison et d'agrégation pour l'inférence et la réduction des informations. Par ailleurs, en plus de la transformation d'informations provenant de différentes sources (capteurs, bases de données, simulations/modèles, humains, *etc.*), cette définition cite clairement le cas de la transformation d'informations obtenues à partir d'une source unique à différents points dans le temps [86], autrement dit le cas du filtrage.

2.5.2 Architecture des systèmes multicapteurs

L'architecture des systèmes multicapteurs peut comporter trois aspects : l'architecture *matérielle*, l'architecture *fonctionnelle* et l'architecture *système*. Le développement des systèmes à plusieurs capteurs a causé une augmentation de la complexité de gestion des données. Ceci exige la simplification des systèmes multicapteurs par leur décomposition fonctionnelle en plusieurs niveaux.

2.5.3 Le modèle JDL

Le modèle JDL proposé en 1987 par le *Joint Directors of Laboratories (JDL) Data Fusion Subpanel* est l'un des premiers modèles de fusion de données, et sans doute un des plus populaires.

Le JDL divise le système en *niveaux* de fusion avec une distinction entre les processus de fusion liés à l'affinement des estimations des paramètres d'intérêt liés aux « objets », « situations », « menaces » et « processus » (figure 2.13).

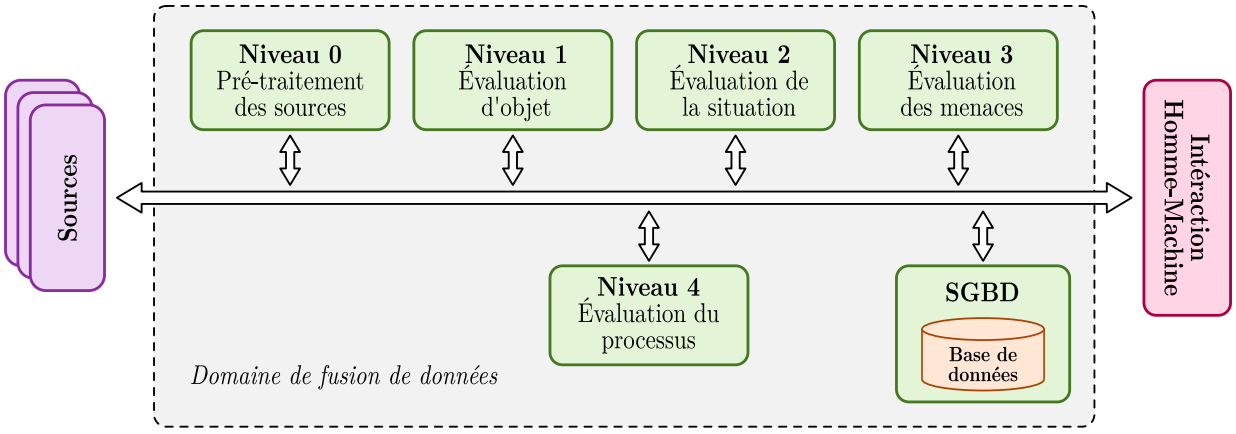


Figure 2.13 – Le modèle JDL classique.

La figure 2.13 est censée représenter soit un seul nœud de fusion, soit le traitement agrégé d'une suite de nœuds de fusion qui auraient chacun une structure similaire. À noter que ce modèle est strictement une aide à la discussion et non un schéma d'architecture ou de traitement [87].

2.5.4 Fusion en trois niveaux et modèle DFD

D'une manière générale, les systèmes de fusion multicapteur peuvent être divisés en trois catégories suivant le niveau des données traités. Dans la fusion multiniveau, nous distinguions trois niveaux :

- Niveau de données (bas niveau) ;
- Niveau d'attributs (niveau intermédiaire) ;
- Niveau de décisions (haut niveau).

La fusion au niveau de données (ou de mesures) combine des données brutes en provenance des capteurs pour produire une donnée en sortie qui serait plus informative. Pour la fusion au niveau d'attributs (ou de caractéristiques), les attributs [6] sont d'abord extraits à partir des données brutes avant de les fusionner pour produire un nouvel attribut qui serait plus informatif. La fusion au niveau de décisions prend des décisions en entrée pour les combiner et produire une meilleure décision en sortie.

Dans ce modèle de fusion multiniveau, les données d'entrée et de sortie du processus de fusion sont considérées de la même nature (même niveau). Néanmoins, il existe d'autres paradigmes de fusion où l'entrée et la sortie du processus de fusion appartiennent à des niveaux différents [88]. Un exemple d'un tel système peut être trouvé dans les algorithmes de la détection et l'extraction d'attributs, les données en entrée appartenant au niveau de données brutes (*ex.* image) et les résultats appartiennent au niveau d'attributs (*ex.* coins, lignes, descripteurs, objets annotés, *etc.*). Ces attributs peuvent être réinjectés dans un autre niveau de fusion pour produire des décisions en fonction des attributs d'entrée (*ex.* freinage du robot si un obstacle a été détecté).

Pour prendre en compte ces cas, Dasarathy [88] a proposé le modèle *Donnée-Attribut-Décision (DFD - Data-Feature-Decision)*, qui catégorise ces systèmes en fonction du niveau des données d'entrée et de sortie, et ainsi, donne cinq catégories de fusion, à savoir : données en entrée – donnée en sortie (DAI-DAO), données en entrée – attribut en sortie (DAI-FEO), attributs en entrée – attribut en sortie (FEI-FEO), attributs en entrée – décision en sortie (FEI-DEO), et décisions en entrée – décision en sortie (DEI-DEO). La figure 2.14 illustre ces cinq catégories.

2.5.5 La configuration des capteurs

Au niveau des capteurs et des données acquises de ceux-ci, les systèmes de fusion multicapteur, appelés parfois *« systèmes de fusion en réseaux de capteurs »*, peuvent être classés en fonction de la configuration des capteurs. Nous prenons cette classification de Durrant-Whyte [89], qui distingue trois types de configuration de capteurs : configuration *complémentaire*, *concurrente* ou *coopérative*, à noter que ces trois types ne sont pas mutuellement exclusifs.

La figure 2.15 schématise la classification en fonction de la configuration des capteurs. Les objets physiques A, B et C sont observés par les capteurs $\mathbf{S_{1,\ldots,5}}$. Les capteurs $\mathbf{S_1}$ et $\mathbf{S_2}$ observent le même objet, ils sont utilisés en configuration *complémentaire* pour améliorer la fiabilité de l'estimation. Les capteurs $\mathbf{S_2}$ et $\mathbf{S_3}$ observent deux objets différents et ils sont utilisés en configuration *concurrente* pour fournir une description plus complète de la scène. Les capteurs $\mathbf{S_4}$ et $\mathbf{S_5}$ observent le même objet et ils sont utilisés en configuration *coopérative* pour estimer une quantité non observable directement par chacun des deux capteurs séparément.

6. Les attributs dans ce contexte peuvent avoir différentes formes, mais d'une manière générale, ils sont des représentations plus ou moins abstraites des informations utiles extraites des données brutes. Par exemple, les coins extraits d'une image peuvent être considérés comme des attributs.

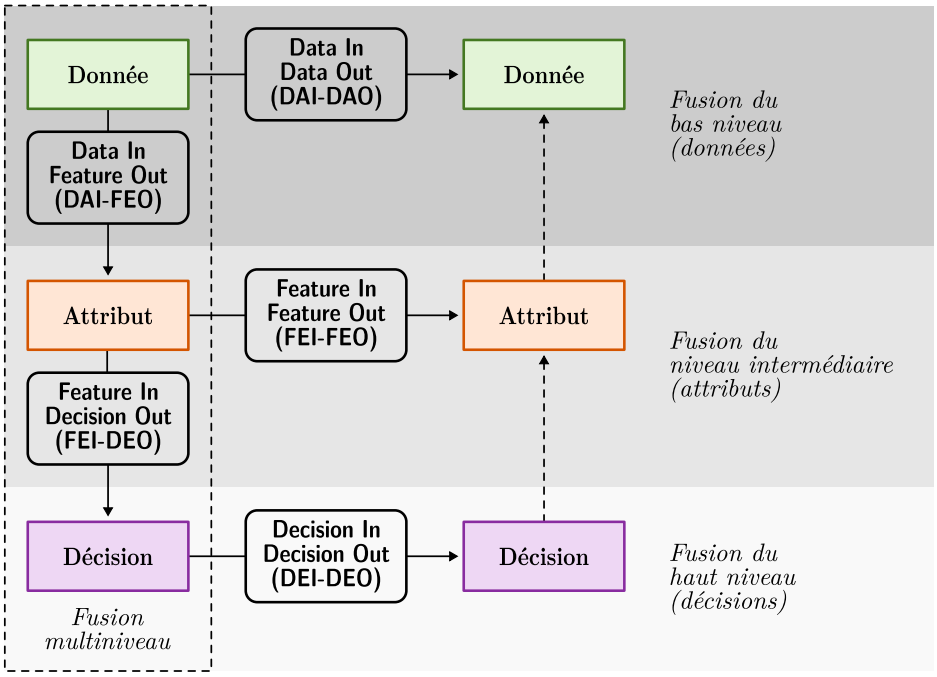


Figure 2.14 – Les catégories des systèmes de fusion dans le modèle DFD (par type de données d'entrée et de sortie).

2.5.5.1 Configuration complémentaire

Une configuration de capteurs est dite complémentaire lorsque les capteurs ne dépendent pas directement les uns des autres, mais peuvent être combinés afin de donner une information plus complète du phénomène observé. Cela résout le caractère incomplet des données du capteur [85].

2.5.5.2 Configuration concurrente

Les capteurs sont configurés en mode concurrent ou compétitif si chaque capteur fournit des mesures indépendantes de la même propriété. Ceci peut être divisé en deux configurations concurrentes possibles : la fusion de données issues de différents capteurs ou la fusion de mesures issues d'un même capteur, mais prises à des instants différents. La configuration concurrente de capteurs est également appelée configuration *compétitive* ou *redondante*. Ces configurations sont adaptées par nature à des applications aux *systèmes tolérants aux pannes (Fault-tolerant systems)*. Les configurations concurrentes peuvent par ailleurs apporter de la robustesse à un système, *c.-à-d.*, assurer un niveau de service dégradé en présence de pannes [90].

2.5.5.3 Configuration coopérative

Un réseau de capteurs coopératifs utilise les informations fournies par deux ou plusieurs capteurs indépendants pour dériver des informations qui ne seraient délivrées individuellement par aucun des capteurs.

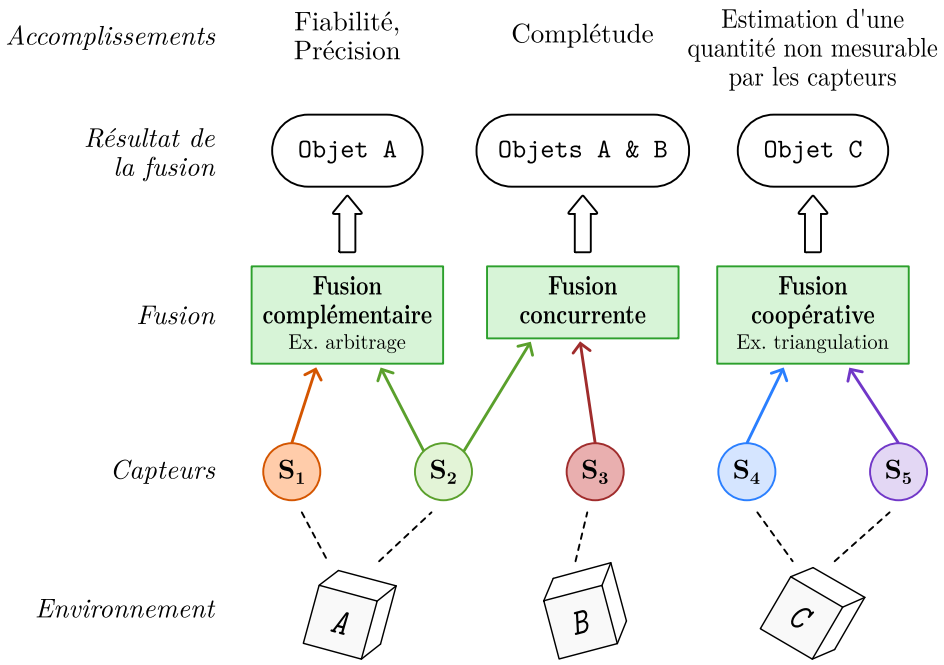


Figure 2.15 – Configurations des capteurs dans les systèmes de fusion de données (adaptée de [90]).

Un exemple d'une configuration collaborative de capteurs est la vision stéréoscopique. Dans cette dernière, en fusionnant les images provenant de deux caméras séparées d'une distance connue, il devient possible d'obtenir des informations sur les profondeurs des points sur l'image. Ceci est possible même si chacune des deux caméras ne peut pas fournir ces informations quand elle est utilisée individuellement.

La fusion multicapteur coopérative est la plus difficile à concevoir, car les données résultantes sont sensibles aux inexactitudes de tous les capteurs individuels participants. Ainsi, contrairement à la fusion compétitive, la fusion coopérative des capteurs diminue généralement la précision et la fiabilité [90].

2.5.6 La fusion multicapteur pour la localisation indoor

Dans le contexte de la localisation indoor, la fusion multicapteur combine les données en provenance des capteurs (caméras, LiDARs, ultrasons, IMUs, *etc.*) pour déterminer avec précision la position et l'orientation du robot dans l'environnement.

La fusion *multimodale* combine les données de plusieurs modalités de capteurs, par exemple, des données visuelles et des données en provenance d'un LiDAR. Ceci est réalisé pour but d'améliorer les performances globales de la localisation. Les systèmes de fusion multimodaux sont particulièrement utiles pour les environnements dans lesquels une seule modalité pourrait être momentanément ou partiellement inefficace, par exemple, la localisation visuelle pendant la nuit ou dans des environnements mal éclairés.

2.5.6.1 Filtre de Kalman

Nommé en l'honneur de l'ingénieur électricien et mathématicien hongrois-américain Rudolf Emil KALMAN. Le filtre de KALMAN est sans doute la méthode la plus connue de la fusion de données. Ce filtre définit un cadre mathématique pour estimer l'état d'un système sur la base d'une combinaison de mesures passées et d'une connaissance préalable de la dynamique du système (modèle physique).

Le filtre de KALMAN est composé de deux étapes principales, la *prédiction* et la *correction* (ou la *mise à jour*). Pour la phase de *prédiction*, le filtre utilise un modèle physique qui décrit la dynamique du système pour calculer une estimation actuelle des variables d'état, ainsi que leurs incertitudes. Une fois qu'une mesure (qui serait bruitée dans tous les cas) est observée, les estimations sont *mises à jour* à l'aide d'une moyenne pondérée, un poids plus important étant accordé aux estimations avec une plus grande certitude. La figure 2.16 illustre les étapes de ce filtre.

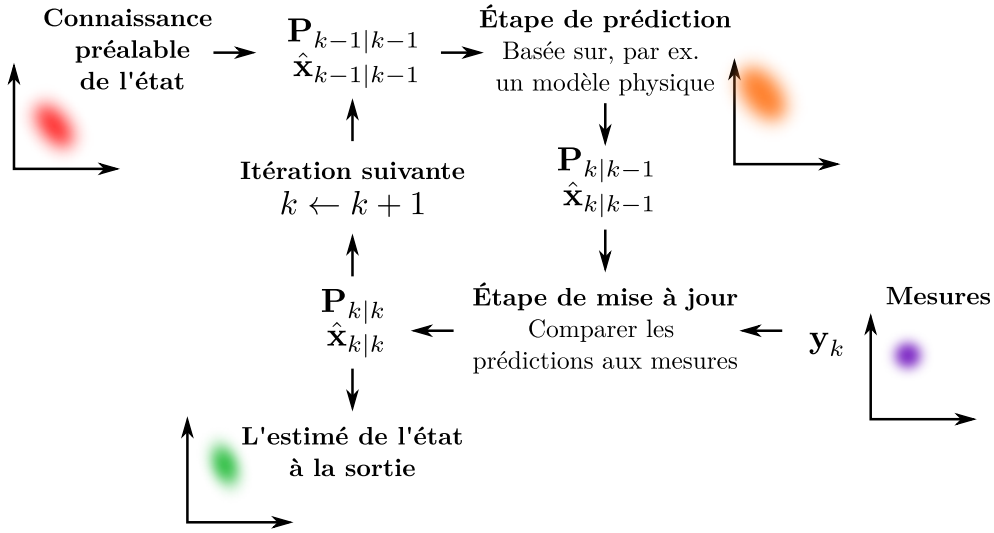


Figure 2.16 – Une illustration des étapes du filtre de KALMAN.

La pondération permet de donner plus d'importance aux valeurs considérées plus fiables, *c.-à-d.*, les valeurs avec une moindre incertitude estimée. Les poids de pondération sont calculés à partir de la matrice de covariance qui décrit l'incertitude estimée lors de la prédiction de l'état du système ainsi que celle qui décrit l'incertitude de la mesure. Le résultat de la moyenne pondérée est une nouvelle estimation d'état qui se situe entre l'état prédit et l'état mesuré, et qui a une meilleure incertitude estimée que l'un ou l'autre seul [91].

Le filtre de KALMAN est récursif, cela signifie qu'il peut être utilisé en temps réel, et que pour calculer le nouvel état, le filtre ne nécessite que la dernière meilleure estimation et sa matrice d'incertitude, plutôt que l'historique complet de l'état du système.

Le filtre de KALMAN repose sur un ensemble d'hypothèses, qui peuvent être résumé de la manière suivante [91] :

- Le modèle de prédiction décrit parfaitement le système réel ;
- Le bruit de process et de mesure sont blancs (bruit gaussien non corrélé centré sur le zéro) ;

- Les covariances du bruit sont connues et reflètent la vraie distribution des bruits ;
- Et les modèles de prédiction et de la mise à jour sont des systèmes linéaires.

Dans le cas où toutes ces hypothèses sont satisfaites, il est facilement prouvable que le filtre de Kalman est l'estimateur optimal. En revanche, dans la vie réelle, les systèmes sont rarement linéaires, ainsi, des extensions du filtre de Kalman pour les systèmes non-linéaires ont été proposées.

2.5.6.2 Filtre de Kalman étendu (EKF)

L'EKF est une extension du filtre de Kalman, qui peut être utilisée pour gérer les systèmes légèrement non-linéaires. Le filtre achève cela en linéarisant localement, autour de l'état actuel du système, les modèles non-linéaires de prédiction et/ou de mesure.

Le filtre de Kalman étendu est un choix populaire pour les systèmes robotiques, car il permet de fusionner des données en provenance de plusieurs capteurs tout en offrant un bon compromis entre la complexité de calcul et la précision.

L'EKF a été largement utilisé dans la littérature scientifique pour traiter le problème de la localisation. Par exemple, il a été employé dans de nombreux contextes tels que les systèmes d'odométrie/SLAM à base de vision ou des LiDARs, ainsi que dans l'intégration de données inertielles [92].

La simplicité d'implémentation et de déploiement du filtre de Kalman étendu, son efficacité de calcul et sa robustesse dans les systèmes légèrement non-linéaires en font de lui un bon outil pour la fusion des mesures de pose dans les systèmes de localisation multicapteurs. Ainsi, l'EKF serait abordé avec plus de détails dans le chapitre 3 de ce manuscrit.

2.5.6.3 Filtre de Kalman non parfumé (UKF)

L'UKF est une autre extension au filtre de Kalman. À la différence de l'EKF, le filtre de Kalman non parfumé (UKF) utilise les modèles du système non-linéaire de prédiction et/ou de mesure sans les linéariser localement.

Afin d'éviter la linéarisation locale des modèles, le filtre de Kalman non parfumé utilise un ensemble de *« points sigma »* de l'état (et/ou des mesures) pour approximer la fonction de densité de probabilité de l'état du système. Ceci lui permet de gérer les non-linéarités avec plus de précision tout en respectant les contraintes mathématiques du filtre de Kalman. L'UKF a été aussi utilisé dans de nombreux systèmes de localisation indoor [92].

2.5.6.4 Fusion par graphe de factorisation

Dans cette approche, la pose (ou l'état) du robot et la carte de l'environnement sont présentées sous forme de graphe de factorisation, où les nœuds représentent les poses du robot et les liens représentent les contraintes entre les poses.

Ainsi, de tels problèmes peuvent être appréhendés par le biais de l'optimisation, dans ces cas, l'approche de fusion par graphe de factorisation peut être utilisée. Dans de nombreux problèmes d'estimation d'état (SLAM, localisation, *etc.*), nous recherchons à estimer le *maximum a posteriori (MAP)*, *c.-à-d.*, nous essayons de maximiser la probabilité a posteriori de l'état compte tenu d'un ensemble de mesures.

2.5.6.5 Filtre à particules

Dans la famille des filtres de KALMAN, l'état est toujours représenté sous la forme d'une variable aléatoire de *distribution supposée gaussienne*, ce qui permet une représentation compacte des états (moments d'ordre 1 et 2). Néanmoins, dans certains cas, la distribution de l'état d'un système complexe ne peut pas être approximé d'une façon gaussienne.

Les méthodes basées sur la simulation de Monte Carlo (MC) expriment l'espace d'état sous forme d'échantillons pondérés (particules) et ne font pas d'hypothèses sur la distribution de probabilité sous-jacente. Cela permet de représenter des états ayant des distributions arbitraires.

Dans le filtre à particules, appelé aussi *Monte Carlo séquentiel (SMC - Sequential Monte Carlo)*, l'état du système est représenté sous la forme d'un ensemble de particules $x_{(t,i)}$, avec t le pas du temps et i l'indice de la particule. Chaque particule a un poids associé représentant la probabilité que la particule $x_{(t,i)}$ correspond à l'observation z_t, *c.-à-d.* :

$$w_{(t,i)} \propto p(z_t|x_{(t,i)}) \tag{2.1}$$

Ainsi, le filtre utilise les données du capteur pour mettre à jour le poids de chaque particule et déterminer l'état le plus probable du système.

L'essence du filtre à particule est l'étape d'*échantillonnage préférentiel (Importance sampling)*, où les particules sont rééchantillonnés en fonction de leur poids ; cette étape se rapproche de la distribution postérieure du filtre de BAYES. FastSLAM [19] est l'un des algorithmes les plus connus de localisation basée sur le filtre à particules.

2.5.6.6 Suivi multihypothèse (MHT)

Cette approche est utilisée pour suivre plusieurs objets dans une scène, en maintenant plusieurs hypothèses sur les états des objets et en les mettant à jour en fonction des données des capteurs.

Le *suivi multihypothèse (MHT - Multiple-Hypothesis Tracking)* est utile pour la localisation d'un robot lorsqu'il y a plusieurs objets dans l'environnement qui pourraient être confondus avec la position de ce dernier. Il est également utile dans le cas de la localisation absolue, où, au démarrage et en fonction de la partie de scène observée, nous pouvons avoir plusieurs états plausibles. Dans ce cas, le MHT permet de suivre tous ces états plausibles et de les mettre à jour pendant le mouvement du robot dans le but de converger vers l'état réel [93].

2.6 Conclusion

Dans ce chapitre, nous avons présenté le problème de la localisation en indoor ainsi que les considérations essentielles à prendre en compte lors de la conception de tels systèmes.

Nous avons effectué une revue de l'état de l'art sur la localisation en indoor en explorant la littérature scientifique afin d'identifier les solutions existantes. Dans ce contexte, nous avons présenté une classification des systèmes de localisation indoor selon plusieurs critères, en présentant les technologies et les capteurs utilisés, ainsi que les techniques permettant d'exploiter les données provenant de ces capteurs.

Suite à cette revue de la littérature scientifique, nous avons exclu plusieurs types de solutions qui ne s'alignent pas sur notre cas d'utilisation, ainsi que celles qui ne satisfont pas nos contraintes de base. Par conséquent, nous avons spécifiquement *écarté les solutions intrusives* qui exigent une infrastructure dédiée ou des modifications/extensions de l'environnement.

Nous avons conclu que le LiDAR, couramment utilisé en robotique pour la perception et la détection d'obstacles, pourrait être une option intéressante pour notre application.

En effet, le LiDAR délivre des *mesures de distance précises* pour plusieurs points d'un plan, et pour plusieurs plans dans le cas des LiDARs 3D. En choisissant un algorithme de traitement adéquat, les nuages de points issus des scans du LiDAR peuvent être alignés pendant les mouvements du robot. Cela permettrait de réaliser des cartographies précises, et ainsi, estimer les poses associées.

Par ailleurs, les robots industriels utilisés pour le déplacement de charges lourdes sont souvent équipés de LiDARs pour garantir la sécurité des travailleurs et préserver l'intégrité des produits transportés. Dans cette optique, il nous semblait pertinent de commencer par exploiter les données des LiDARs qui seront, de toute façon, présents sur le robot afin de les qualifier et d'identifier leurs limitations. Ainsi, nous nous concentrerons sur la localisation basée sur l'alignement *(scan-matching)* des données du LiDAR dans le prochain chapitre.

Parmi les diverses méthodes de fusion de données exposées dans ce chapitre, nous avons opté pour le filtre de Kalman étendu (EKF) que nous exploitons dans le prochain chapitre. Ce choix découle de la simplicité d'implémentation et de déploiement de ce filtre, de son efficacité de calcul ainsi que de sa robustesse dans les systèmes légèrement non-linéaires. En effet, dans notre contexte, l'EKF semble offrir un bon compromis entre performance et précision de l'estimation. Nous aborderons donc la fusion à base du filtre de Kalman étendu de manière plus détaillée dans le prochain chapitre.

CHAPITRE 3

Localisation par alignement des données multi-LiDARs

3.1 Introduction

L'usage des mesures de distances dans les systèmes de localisation a été largement démocratisé dans le monde de la robotique mobile. En effet, les capteurs ultrasoniques ont été parmi les premiers utilisés pour cette tâche, suivis par les LiDARs qui permettaient d'acquérir des mesures de hautes définitions qui sont beaucoup plus précises et fiables.

La mesure directe des distances dans une scène permet au robot de se disposer, par rapport à son point de vue au moment de l'acquisition, d'une connaissance locale et métrique sur l'environnement. Cette connaissance peut ensuite être intégrée incrémentalement dans la reconstruction d'une carte plus grande qui correspond à la représentation de l'environnement observé par le robot lors de ses déplacements.

Dans un premier temps, nous avons exploré la piste des approches de localisation indoor à base de LiDARs en s'appuyant sur la plateforme expérimentale réalisée à ez-Wheel, l'entreprise où s'effectue cette thèse. Il s'agit du robot *SmartTrolley* équipé de deux LiDARs placés à la diagonale (voir la figure 3.1) ainsi que deux codeurs incrémentaux qui permettent de mesurer les vitesses de rotation des deux roues motrices.

La figure 3.1 présente les deux vues, de dessous et de profil, du robot *SmartTrolley*. L'architecture mécanique du robot est assez simple, les roues motorisées sont montées au centre du robot pour permettre un entrainement différentiel *(Differential Drive)*, avec quatre roues folles qui permettent de supporter le robot et le maintenir à l'horizontal. Le placement des LiDARs est illustré sur la figure 3.1 qui montre une position verticale proche du sol pour assurer la détection des obstacles et un placement des deux LiDARs en diagonal, dans le but de pouvoir observer la totalité de l'espace entourant notre robot. La figure montre également un petit décalage entre les plans de mesures des deux LiDARs qui pourrait être accompagné d'une éventuelle inclinaison (différence d'orientation) de ces plans de mesures, l'un par rapport à l'autre.

La configuration des LiDARs en diagonale présente l'avantage de couvrir un champ de vision de 360°, ce qui pourrait être intéressant pour des applications de sûreté en site industriel. En revanche, elle peut poser quelques problèmes liés aux alignements des points perçus par les deux LiDARs ainsi que des soucis de synchronisation de l'acquisition des données à partir des deux capteurs.

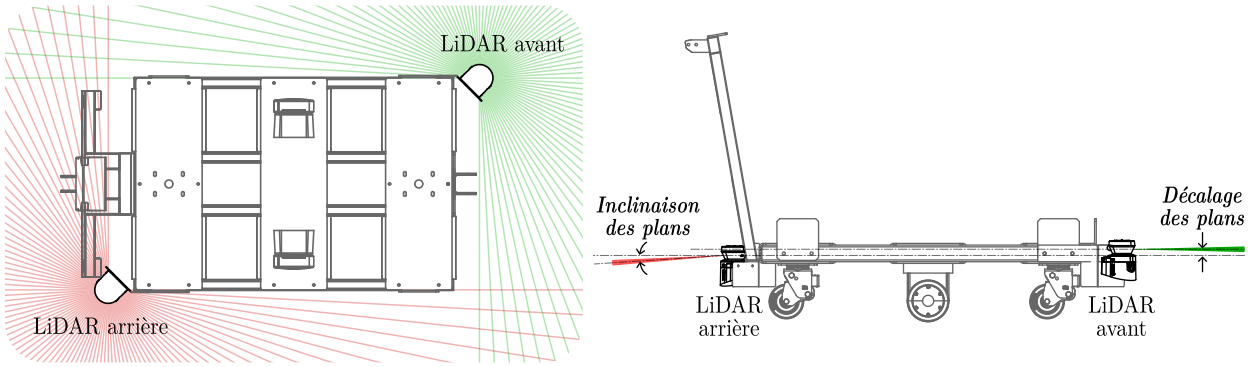


Figure 3.1 – Montage des deux LiDARs sur la plateforme expérimentale *SmartTrolley*. À gauche, une vue de dessus illustrant le champ de vue angulaire de chaque LiDAR. À droite, une vue de côté illustrant le potentiel décalage et/ou inclinaison entre les plans perçus par les deux LiDARs.

Dans ce chapitre, nous allons mettre en œuvre une implantation d'un système de localisation indoor sur le robot *SmartTrolley*, embarquant les capteurs avec une disposition supposée optimale pour les fonctions à accomplir. L'objectif de cette étape est de valider l'architecture globale, électronique et logicielle, qui permet à ce robot de se déplacer.

Ensuite, nous réaliserons des expérimentations afin de récolter un premier jeu des données des LiDARs et de l'odométrie des roues tout en gardant la configuration et le placement initiale des capteurs. Cela permettra de valider de manière incrémentale l'ensemble des couches de contrôle, de l'acquisition des données et de calcul de notre robot. Cela nous a permis également de mettre en place une première contribution d'un système de localisation incrémentale à base de la fusion des données en provenance des LiDARs et de l'odométrie des roues.

Cette approche de validation incrémentale nous semble bien adaptée à notre contexte industriel, car des améliorations successives découleront des critiques et des limitations que nous pourrons formuler et identifier à chaque étape.

3.2 La plateforme expérimentale SmartTrolley

Dans cette section, nous présentons le *SmartTrolley* [94], une plateforme mobile pour déplacer des lourdes charges dans des sites industriels, des grands entrepôts ou des usines. Nous utilisons cette plateforme expérimentale par la suite pour valider notre approche de fusion dans un algorithme de localisation incrémentale multi-LiDARs.

Nous avons utilisé un châssis à base d'un chariot standard à fortes charges de dimensions 1.2m × 0.8m. La propulsion de la plateforme est assurée par deux roues motorisées que nous avons montées sur l'axe central du chariot, séparées de $0.6m$, afin qu'elles fournissent un

entrainement différentiel. Pour stabiliser la plateforme, nous utilisons quatre roues folles (roulettes) montées une à chaque coin. Une vue de dessous annotée de la plateforme et une vue isométrique de celle-ci sont illustrées sur la figure 3.2.

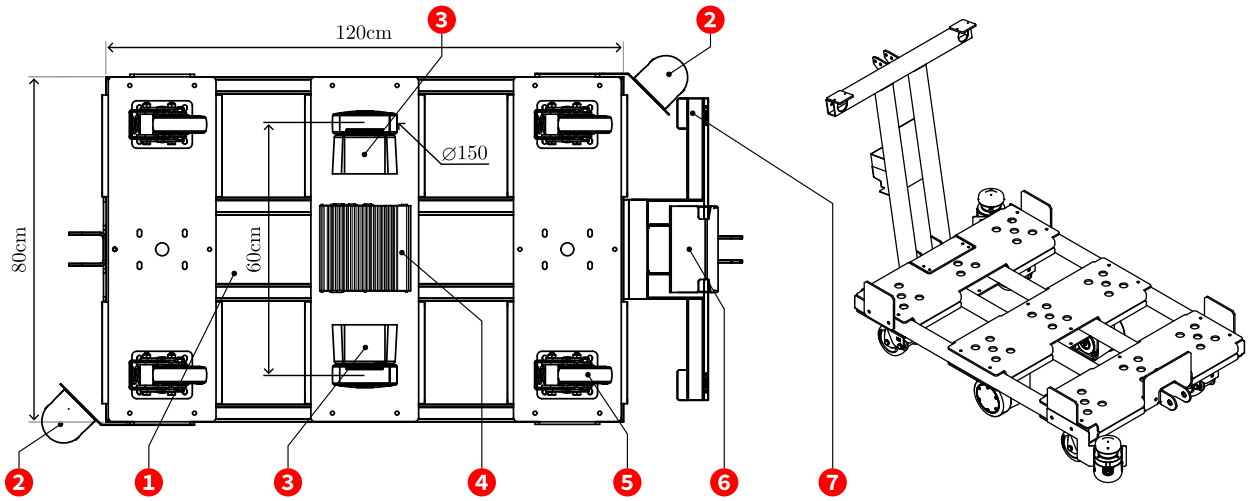


Figure 3.2 – Illustration des différents composants de la plateforme expérimentale *SmartTrolley*. (1) Le châssis métallique ; (2) Deux LiDARs de sécurité Sick S300 couvrant ensemble un champ de vue de 360° ; (3) Deux prototypes des roues SWD® 150 ; (4) Un ordinateur embarqué *Neousys Nuvo-7002LP* ; (5) Quatre roues folles ; (6) Bloc de batteries ; (7) Bouton d'arrêt d'urgence.

Dans un premier temps, nous avons équipé la plateforme de deux LiDARs de sécurité en plus des codeurs intégrés dans les roues motorisées. Nous l'avons aussi équipé d'un calculateur industriel embarqué qui prend en charge plusieurs protocoles de communication. Ceci permet de faciliter l'ajout d'autres capteurs afin de tester et de valider de diverses solutions de localisation, notamment celles à base de fusion multicapteur.

3.2.1 Propulsion

Nous avons utilisé deux prototypes de la roue de la nouvelle génération d'ez-Wheel, à savoir la *SWD® 150* (figure 3.3). Cette roue est en mesure de générer un couple capable de déplacer environ 1500kg de charges par roue. Le modèle *SWD® 150* intègre :

- Une roue de diamètre ⌀150mm
- Un *train d'engrenages épicycloïdal*[1] à deux étages entre le moteur et la roue offrant un facteur de réduction de 1 : 14 ;
- Un moteur Brushless à courant continu (BLDC) ;
- Un codeur incrémental à effet Hall de 420 ticks/rev ;
- Un bloc de batteries Lithium-ion de 100Wh ;
- Un système de gestion de batterie (*BMS - Battery Management System*) intégré ;
- Un circuit pour l'électronique de puissance ;

1. Le « train d'engrenages épicycloïdal », en anglais : « epicyclic gear train », est connu aussi comme le « train d'engrenages planétaires » ou « planetary gearset ».

- Un calculateur embarqué pour applications critiques de la famille *Texas Instruments Hercules™* qui assure les communications, la régulation de la vitesse et la gestion des entrées/sorties (E/S) sécurisées ;
- Un frein externe optionnel ;
- Un système sur module (SoM) optionnel de type *Variscite DART-MX6* de processeur *NXP/Freescale i.MX6 Quad-core Cortex-A9™*, pour les opérations de haut niveau ;
- Des interfaces de connexion pour les entrées/sorties (E/S) de sécurité (OSSD), l'USB, le bus CAN et l'Ethernet.

Le tableau 3.1 récapitule les caractéristiques principales de la roue ez-Wheel SWD® 150, et plus spécifiquement les performances maximales en vitesse et en force.

Table 3.1 – Les caractéristiques principales de la roue ez-Wheel SWD® 150

Paramètre	**Désignation**
Codeur	Effet Hall - 420 tick/rev SIL2/PLd
Diamètre de la roue [mm]	150
Type de pneu	PU 92 sh.A - Flat profile
Plage de vitesse (traction) [$km \cdot h^{-1}$]	0 à 3
Effort de poussée maximal [daN]	60 (suffisant pour déplacer 1500kg)
Performance nominale [daN]	22 (à 3km/h)
Vitesse linéaire maximale [$km \cdot h^{-1}$]	4
Poids vertical maximal [kg]	700 (jusqu'à 3km/h)
Capacité de la batterie intégrée [Wh]	100

La nouvelle technologie SWD® est destinée à des applications en robotique autonome, elle assure une intégrité de sécurité de niveau 2 et 3 *(SIL - Safety Integrity Level)*, et elle est conforme aux normes européennes relatives à la sûreté, notamment : l'EN 61508-5-2, l'EN 62061, l'EN 13849-1, l'EN 60204-1 et l'EN 61800-5-2.

Ainsi, la roue SWD® offre plusieurs fonctionnalités de sûreté que nous listons sur le tableau 3.2. Ces fonctionnalités sont exposées via un bus CANopen Safety®, et le cas échéant, via des E/S sécurisées *(OSSD - Output Signal Switching Device)*.

Table 3.2 – Les fonctions de sécurité supportées par la technologie ez-Wheel SWD®

Acronyme	**Fonction de sécurité**	**Description**	**Référence**
STO	Safe Torque Off	La déconnexion sûre du moteur	SIL3/PLe/Cat4
SBC	Safe Brake Control	Le freinage sûr	SIL2/PLd/Cat3
SLS	Safe Limited Speed	La limitation sûre de vitesse	SIL2/PLd/Cat3
SDI	Safe Direction Indication	L'indication de direction sûre	SIL2/PLd/Cat3

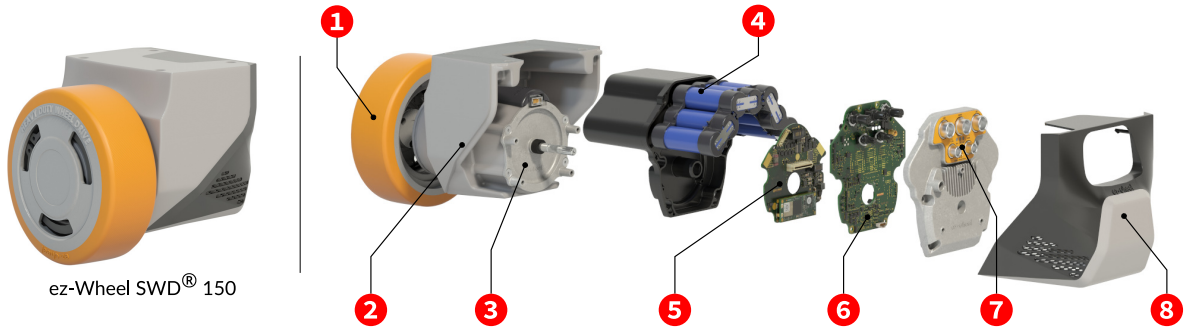


Figure 3.3 – La SWD® 150 d'ez-Wheel illustrée avec la roue assemblée à gauche et une vue éclatée sur les composants internes à droite. (1) Roue et train d'engrenages épicycloïdal, (2) Boîtier en acier dimensionné pour supporter des fortes charges, (3) Moteur BLDC, (4) Bloc de batteries, (5) Système de gestion de batterie (BMS) et optionnellement un SoM DART-MX6 avec un processeur NXP/Freescale i.MX6 Quad-core Cortex-A9™, (6) Codeur à effet Hall, électronique de contrôle sûr et de communication, (7) Prises de connectivité, (8) Boîtier en plastique.

3.2.2 Calculateur embarqué

Nous avons utilisé un calculateur industriel embarqué de type *Neousys Nuvo-7002LP* (figure 3.4a), intégrant un processeur *Intel Coffee Lake Core i5* de 8e génération et une mémoire *SDRAM DDR4 2666/2400* de 16 Go. Par défaut, cet ordinateur prend en charge de nombreux protocoles d'E/S, dont deux ports programmables RS-232/422/485 que nous avons utilisés pour connecter les deux LiDAR *S300* via RS-422. Pour connecter nos deux roues SWD®, nous avons ajouté une carte d'interfaçage au bus CAN à la configuration initiale. Nous avons utilisé *Ubuntu 18.04 LTS* comme système d'exploitation avec le middleware robotique *ROS Melodic Morenia*.

(a) Le calculateur Neosys Nuvo-7002LP **(b)** Le LiDAR de sécurité Sick S300 Standard

Figure 3.4 – Calculateur embarqué et LiDAR utilisés dans la plateforme expérimentale SmartTrolley.

3.2.3 Capteurs proprioceptifs

Les capteurs *proprioceptifs* ou *intéroceptifs* permettent de mesurer l'état interne du robot (tel que : la vitesse de rotation des roues, l'angle de braquage, l'état de charge des batteries, *etc.*), mais sans dépendre d'observations externes.

En robotique, les codeurs à roues sont les capteurs proprioceptifs les plus communs ; ils fournissent un compteur d'impulsions qui mesure l'angle de rotation de la roue de manière incrémentale ou absolue. La résolution d'un codeur est définie par le nombre d'impulsions *(ticks)* qu'il est capable de fournir par un tour de roue (unité *impulsions/tour* ou *ticks/rev*). Les codeurs de roues peuvent être divisés en deux types ; codeurs *incrémentaux* utilisés généralement pour la régulation de vitesse (*ex.*, régulation de la vitesse de propulsion) et codeurs *absolus* utilisés pour la régulation de position (*ex.*, régulation de l'angle de braquage dans un robot de type véhicule).

La roue SWD® contient un codeur incrémental (à effet Hall) intégré ; ce dernier est certifié SIL2/PLd et utilisé en interne pour la régulation sûre de la vitesse de rotation de la roue. Les données du codeur sont mises à disposition par l'électronique de la roue via le bus CAN. Ainsi, nous avons exploité ces données, en provenance des deux roues, pour calculer l'odométrie du robot (voir la section 3.5.1.2 de ce chapitre).

3.2.4 Capteurs extéroceptifs

Les capteurs *extéroceptifs* permettent de mesurer un état externe au robot, *c.-à-d.*, réaliser des mesures sur l'état de l'environnement. Les capteurs de distance (LiDARs, ultrasons, *etc.*), de la température, de la pression atmosphérique et les caméras en sont des exemples.

Pour percevoir l'environnement, nous utilisons deux LiDARs de type *SICK S300 Standard* (figure 3.4b) montés en diagonale pour couvrir un champ de vision de 360° (figure 3.2). Le tableau 3.3 présente les caractéristiques techniques essentielles du SICK S300 Standard, pour des informations plus détaillées, vous pouvez vous référer à la notice d'instructions de ce LiDAR.

Table 3.3 – Les caractéristiques du LiDAR SICK S300 Standard utilisé.

Paramètre	**Spécification**
Champ de protection [m]	2-3
Champ d'avertissement [m]	8
Portée de mesure [m]	30
Erreur systématique [mm]	±20
Champ angulaire [deg]	270
Résolution angulaire [deg]	0.5
Temps de balayage [ms]	30
Tolérance de détection [mm]	100
Champ de détection	Du noir-réflecteur de 1.8% au rétro-réflecteur
Type d'optique	PLD *(Pulsed Laser Diode)*
Classe du laser	Class 1 (IEC60825-1)
Longueur d'onde [nm]	895-915, typique 905

Le S300 est un capteur populaire pour les applications industrielles ; il utilise le principe du temps de vol du laser pour mesurer les distances. Ce modèle est destiné à des applications de surveillance d'espaces sensibles en environnements d'intérieurs, par exemple, il est souvent

utilisé pour la détection d'intrusion dans les zones réservées à des machines à risque. Le S300 est conforme à de nombreuses directives et normes centrées sur la sécurité, telles que l'EN ISO 12100, l'ISO 11161, l'EN ISO 10218-1 et l'ANSI/RIA R15.06. Le *S300 Standard* possède une plage de mesure maximale de 30m, un champ de vision de 270° avec une résolution angulaire de 0.5° et offre des fonctionnalités configurables centrées sur la sécurité avec un champ maximal de protection de 3m.

L'architecture globale de la plateforme SmartTrolley est illustrée dans la figure 3.5.

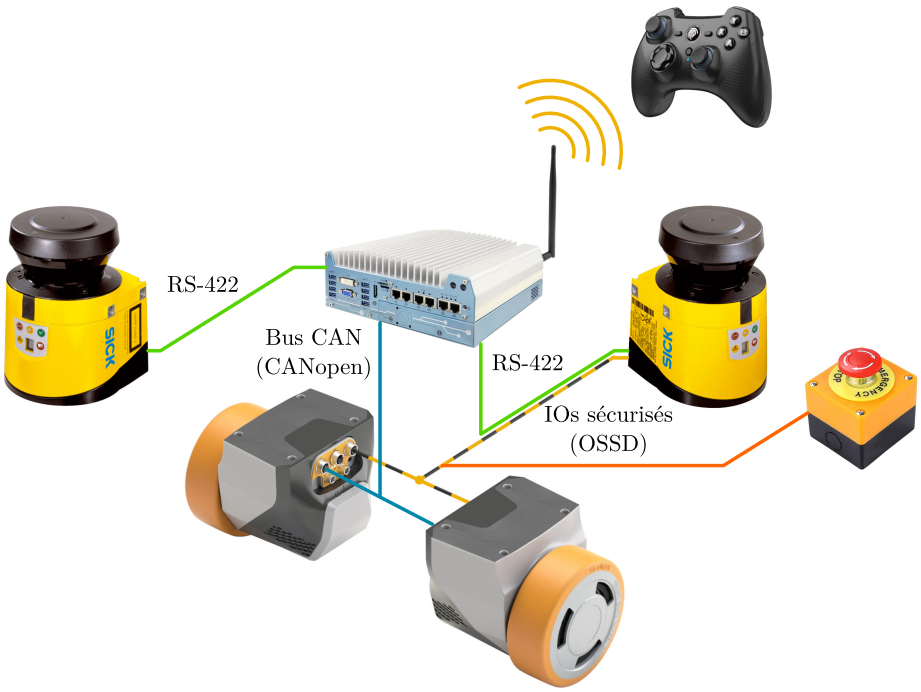


Figure 3.5 – Architecture de la plateforme expérimentale *SmartTrolley*. Les roues sont connectées via des E/S sécurisés au bouton d'arrêt d'urgence et au LiDAR avant. Les roues sont également reliées à un bus CAN. L'ordinateur embarqué utilise ce bus pour commander les roues, obtenir leur état et les données des codeurs. Deux LiDARs sont reliés au calculateur via des ports RS-422 pour transférer les données des scans en nuages de points.

Pour améliorer la sécurité de notre système, le S300 peut être configuré avec le logiciel *SICK Configuration & Diagnostics Software (CDS)* pour définir des zones critiques autour du robot. En condition opérationnelle, si un objet pénètre une zone critique, le LiDAR envoie un signal sur ses *sorties de sécurité*. Ces sorties peuvent être reliées aux freins ou au calculateur embarqué pour gérer la situation critique de sécurité. Nous avons exploité cette fonctionnalité en configurant les roues SWD® pour prendre en compte ces informations, ce qui nous a permis d'améliorer la sécurité dans les environnements dynamiques au niveau le plus bas.

Ainsi, trois zones sont définies, une zone d'avertissement qui donne une indication au système qu'un objet est à proximité du robot. Une deuxième zone de limitation de vitesse (via le signal SLS) qui impose d'une manière sûre, au niveau du calculateur intégré à la roue, une vitesse maximale à ne pas dépasser. Et une troisième zone d'interdiction de direction

(via le signal SDI), qui impose une interdiction d'avancer si l'obstacle se trouve devant le robot ou de reculer si l'obstacle se trouve derrière lui. La figure 3.6 illustre ces différentes zones.

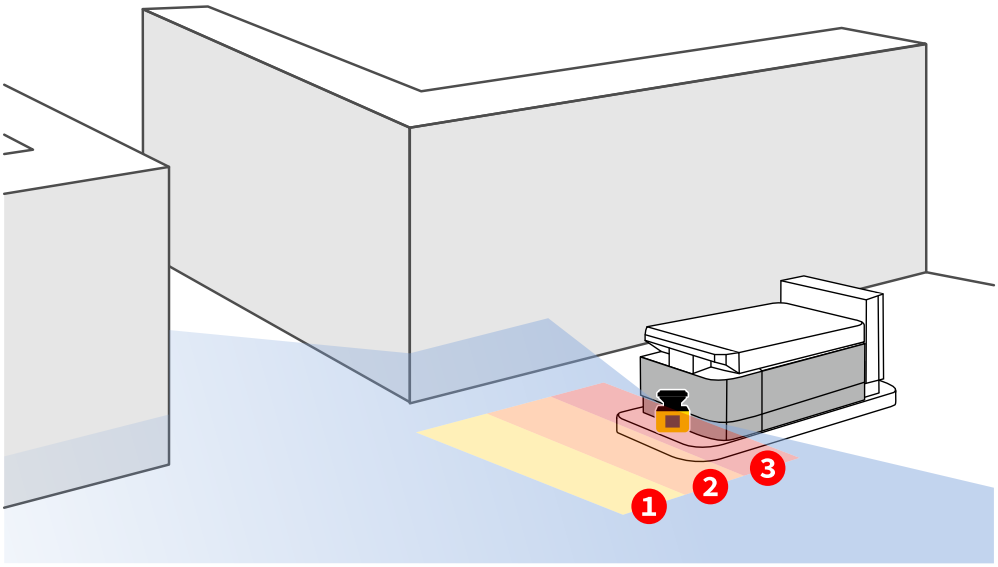


Figure 3.6 – Les zones de sécurité définies dans notre système. (1) la zone d'avertissement, (2) la zone de limitation de vitesse, (3) la zone d'interdiction de direction.

3.3 Les LiDARs

Le LiDAR, acronyme de *Light Detection and Ranging*, désigne la technique de *« détection et estimation de la distance par la lumière »*. Il est un instrument actif qui permet de mesurer des distances à base d'une source d'ondes électromagnétiques dans le domaine de la lumière visible ou l'infrarouge. Il s'agit d'un capteur actif, *c.-à-d.*, il se compose de deux parties : un *émetteur* et un *récepteur*. L'émetteur (généralement, une source de lumière cohérente, *c.-à-.d*, laser, émettent dans le domaine de l'infrarouge), émet un faisceau laser avec une ouverture angulaire très étroite. Et le *récepteur* permet de détecter ce faisceau réfléchi par un obstacle dans l'espace de mesure.

Trois principales technologies de LiDARs sont disponibles : les LiDARs à *ondes continues (CW - Continuous-wave)*[2], les LiDARs *à base d'impulsions (PB - Pulse-based)* et les LiDARs *sur semi-conducteur (Solid-State LiDARs)*.

Les CW-LiDARs estiment la distance en mesurant la différence de phase $\Delta\lambda$ entre l'onde émise et l'onde réfléchie par la surface de l'objet. Tandis que les PB-LiDARs estiment la distance en mesurant directement le temps de vol Δt nécessaire à une impulsion lumineuse pour faire un aller-retour de l'émetteur au récepteur.

Les LiDARs utilisés pour la perception en robotique sont généralement dotés d'une partie mécanique consistant en un moteur rotatif qui fait tourner l'émetteur/récepteur, ou fait tourner un miroir placé devant un émetteur/récepteur fixe. Cela permet d'acquérir des

2. Connus aussi comme *LiDARs à base de déphasage (Phase-shift LiDARs)*.

données sur plusieurs points d'un plan horizontal (LiDARs 2D) ou, avec un mécanisme supplémentaire, d'acquérir des données sur plusieurs plans (nappes) grâce à un mouvement à la verticale (LiDARs 3D). Cependant, les LiDARs *sur semi-conducteur (Solid-State LiDARs)* constituent une exception à cette règle, car ils n'ont pas de pièce mobile dans leur structure.

La figure 3.7 illustre les principes de fonctionnement des principales technologies des LiDARs. À gauche, une représentation d'un LiDAR à ondes continues (CW-LiDAR) équipé d'un miroir tournant, comme c'est le cas du *SICK LMS 100*. Au milieu, un LiDAR à impulsions (PB-LiDAR) avec des paires d'émetteur/récepteur en rotation, tel qu'il est conçu le *Velodyne Alpha Puck LiDAR*, qui utilise un réseau de 128 paires. À droite, une représentation d'un LiDAR sur semi-conducteur, utilisant un réseau optique phasé pour dévier les faisceaux laser, comme c'est le cas du *Quanergy S3-2* [95].

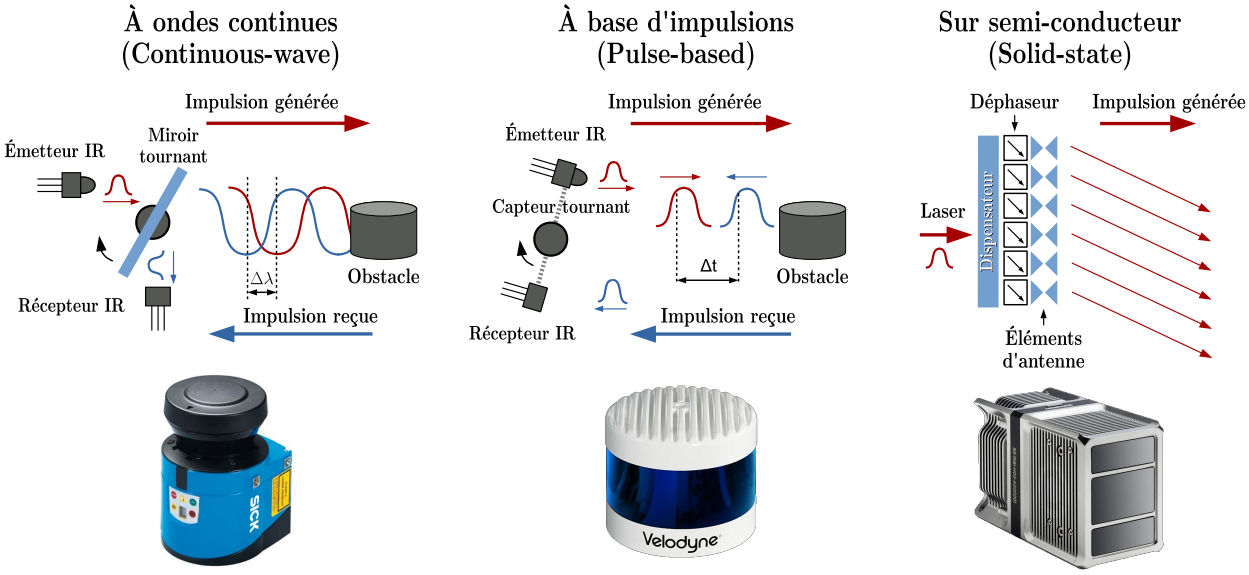


Figure 3.7 – Principes de fonctionnement (rangée du haut) de trois types de LiDAR (rangée du bas). De gauche à droite : un LiDAR à ondes continues (CW-LiDAR), un LiDAR à impulsions (PB-LiDAR) et un LiDAR sur semi-conducteur (adaptée de [95]).

3.4 Appariement des données du laser *(scan-matching)*

L'appariement (appelé aussi *alignement*, *enregistrement* ou *scan-matching*) de balayages (ou *scans*) laser est une stratégie de localisation et/ou de cartographie de haut niveau. Elle se base sur l'hypothèse que *« deux balayages laser d'un environnement qui sont spatio-temporellement colocalisés sont similaires et peuvent être alignés »* [96].

Les méthodes d'alignement des balayages laser recherchent des transformations qui alignent deux balayages effectués par un robot dans le même environnement, mais à des emplacements et des points de vue différents. Le premier balayage, effectué au premier emplacement, est appelé *scan de référence* et le deuxième, effectué au deuxième emplacement, est appelé *scan actuel* ou *nouveau scan*. La transformation qui aligne les scans est définie par des paramètres qui décrivent la position et la rotation relatives entre les deux scans [96].

Le LiDAR délivre des vecteurs de N mesures de distances $S = \{\rho_0, \rho_1, \ldots, \rho_N\}$ correspondant aux points d'intersection d'un faisceau laser avec des objets dans l'environnement du robot (*fig.* 3.8). Les N mesures correspondent à des angles d'acquisition $A = \{\alpha_0, \alpha_1, \ldots, \alpha_N\}$ calculables en fonction de la résolution angulaire du LiDAR. Les paires $u_i = (\rho_i, \alpha_i)$ constituent les coordonnés polaires des points mesurés. Ainsi, un balayage de distances décrit une tranche 2D de l'environnement.

Afin de simplifier la formulation du problème, nous supposons que le LiDAR est monté au centre du robot, *c.-à-d.*, la transformation rigide (position et orientation relatives) entre le repère lié au robot et celui lié au LiDAR est supposée nulle. Dans la pratique, cela peut être corrigé à l'aide d'une transformation de corps rigide qui décrit la position et/ou l'orientation du LiDAR par rapport au centre du robot.

Les mesures u_i peuvent être converties de la représentation polaire à la représentation équivalente en coordonnées cartésiennes p_i tel que :

$$p_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \rho_i \cos(\alpha_i) \\ \rho_i \sin(\alpha_i) \end{bmatrix} \quad \text{avec} \quad i \in [0, N[\tag{3.1}$$

Supposons un robot qui observe l'environnement depuis deux poses différentes aux instants $k-1$ et k en produisant deux scans du LiDAR S_{k-1} et S_k centrés sur leurs origines respectives $\mathcal{O}_{k-1}$ et $\mathcal{O}_k$ (correspondent aux origines du LiDAR aux instants $k-1$ et k), voir la figure 3.8. Le but de l'appariement est de réaliser la mise en correspondance des deux scans, en cherchant les paramètres de la translation et la rotation qui permettent de *superposer les parties communes des nuages de points des deux scans*. Le résultat de cette procédure donne la transformation relative $[\Delta x \ \Delta y \ \Delta\phi]^\mathsf{T}$ qui représente le changement de la pose (position et orientation) du robot entre $k-1$ et k.

La sous-figure 3.8a montre une représentation des données acquises vues par le robot (dans son repère attaché) entre deux poses. La sous-figure 3.8b illustre le résultat de l'appariement de ces données qui met en évidence un mouvement du robot sous la forme d'une translation $[\Delta x \ \Delta y]^\mathsf{T}$ et d'une rotation $\Delta\phi$ entre les deux poses.

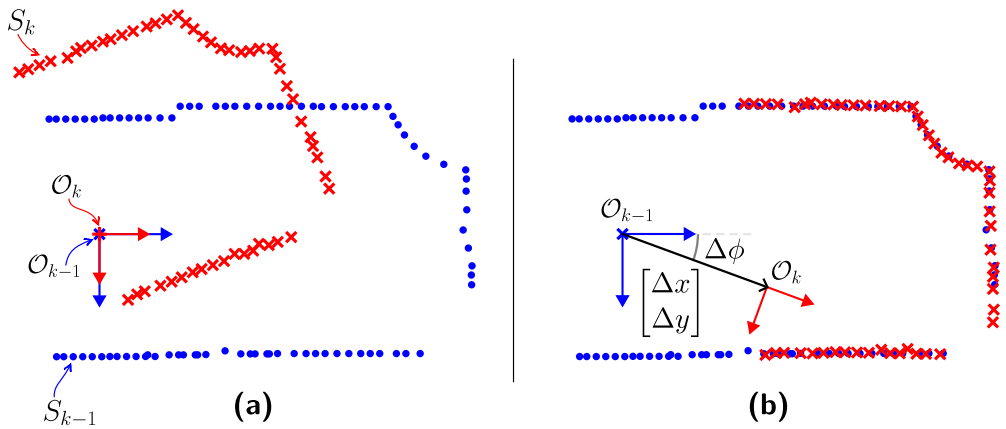


Figure 3.8 – Illustration de l'appariement *(scan-matching)* de deux scans laser. Le scan de référence représenté en bleu (•) et le nouveau scan en rouge (×). En (a), les deux scans sont représentés du point de vue égocentré du robot. En (b), les deux scans sont alignés, résultant en une transformation relative (translation $[\Delta x \ \Delta y]^\mathsf{T}$ et rotation $\Delta\phi$) entre les deux scans.

Le problème d'appariement peut donc être formulé sous forme d'une recherche de la transformation de corps rigide $\mathcal{T}$ entre les référentiels liés à l'origine du robot aux instants $t-1$ et t, notés $\mathbf{x}_{t-1}$ et $\mathbf{x}_t$, respectivement, tel que :

$$\mathcal{T}(\mathbf{x}_{t-1}, \Delta\mathbf{x}_t) : \begin{bmatrix} x_t \\ y_t \\ \phi_t \end{bmatrix} = \begin{bmatrix} \cos\Delta\phi_t & -\sin\Delta\phi_t & 0 \\ \sin\Delta\phi_t & \cos\Delta\phi_t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \phi_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\phi_t \end{bmatrix} \tag{3.2}$$

$$\text{avec} \quad \begin{cases} \mathbf{x}_{t-1} & = [x_{t-1}\ y_{t-1}\ \phi_{t-1}]^\mathsf{T} \\ \Delta\mathbf{x}_t & = [\Delta x_t\ \Delta y_t\ \Delta\phi_t]^\mathsf{T} \\ \mathbf{t}_t & = [\Delta x_t\ \Delta y_t]^\mathsf{T} \\ \mathbf{R}_t & = \begin{bmatrix} \cos\Delta\phi_t & -\sin\Delta\phi_t & 0 \\ \sin\Delta\phi_t & \cos\Delta\phi_t & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{cases}$$

Il existe de nombreuses techniques d'alignement de données du LiDAR. Dans les sections suivantes, nous présentons quelques techniques qui sont couramment utilisées dans le contexte de la localisation.

3.4.1 Alignement ICP – Iterative Closest Point

L'ICP, ou l'algorithme *itératif du point le plus proche (Iterative Closest Point)* [97] est le standard de facto de l'alignement des balayages laser. L'ICP se base sur la recherche des paires de points les plus proches dans deux balayages successifs issus d'un même LiDAR à deux instants différents. Pour chaque point du scan à aligner, l'algorithme cherche à trouver le point le plus proche dans le scan de référence, puis cherche les paramètres de la transformation qui minimisent la distance euclidienne entre ces deux points. L'ICP comprend les étapes suivantes :

1. Prétraitement ;
2. Appariement *(matching)* ;
3. Réjection ;
4. Évaluation de la fonction de coût ;
5. Minimisation de la fonction de coût (passez à l'étape 1 tant que le critère de terminaison n'a pas été satisfait).

À l'étape du prétraitement, les points qui ne sont pas utiles pour faire correspondre les deux nuages de points (par exemple, les valeurs aberrantes) sont filtrés pour réduire le volume des données à traiter. L'étape d'appariement permet de trouver les paires des points (p, q) les plus proches dans les scans appariés, S_t et S_{t-1} respectivement :

$$d(p, S_{t-1}) = \min_{q \in S_{t-1}} \|p - q\| \tag{3.3}$$

La phase de réjection élimine les paires de points séparées de grandes distances. La fonction de coût, Λ, qui évalue la combinaison de rotation et de translation, utilise une métrique de distance moyenne quadratique point-à-point qui alignera au mieux chaque point

du scan actuel sur sa correspondance trouvée à l'étape précédente. La fonction de coût Λ s'applique sur l'ensemble des points du balayage de référence S_{t-1} et ceux du balayage actuel S_t. Elle est définie comme :

$$\Lambda(S_{t-1}, S_t, \mathcal{T}_t) \triangleq \sum_i w_i \|\mathbf{R}_t \cdot q_i - p_i + \mathbf{t}_t\|^2 \tag{3.4}$$

$$\text{avec} \quad \begin{cases} q_i \in S_{t-1} \\ p_i \in S_t \\ w_i = \begin{cases} 1 & \text{si} \quad q_i = \arg\min\limits_q d(p_i, S_{t-1}) \\ 0 & \text{sinon} \end{cases} \end{cases}$$

Où $\mathcal{T}_i$ est la transformation définie par l'équation (3.2), $S_t = \{p_i\}$ est le scan à aligner, $S_{t-1} = \{q_i\}$ est le scan de référence, et w_i est un facteur de sélection qui égale à 1 si le point q_i est le point qui minimise l'équation (3.3), *c.-à-d.*, le point le plus proche du point p_i.

La fonction objective (3.4) est par la suite minimisée itérativement jusqu'à la satisfaction du critère de convergence. Classiquement, cette minimisation peut être achevée d'une manière directe, d'abord par le calcul des centres de masses des deux nuages de points, qui donne une approximation de la translation, puis la recherche de la rotation optimale par décomposition en valeurs singulières *(Singular Value Decomposition - SVD)*. Les paramètres de la transformation $\mathcal{T}$ dans l'ICP peuvent être estimés par d'autres méthodes d'optimisation, par exemple, en appliquant la méthode de Newton-Raphson à la fonction objective de l'équation (3.4).

La méthode ICP permet ainsi d'apparier deux nuages de points, bidimensionnels ou tridimensionnels. En revanche, les points observés par un LiDAR n'existent pas en réalité, ils sont plutôt des échantillons d'une surface d'un objet plus grand (mur, pilier, meuble, *etc.*), par conséquent, l'appariement point-à-point peut être trompeur. Pour remédier à ça, plusieurs variantes de l'ICP ont été proposées dans la littérature scientifique, tel que la *ICP point à plan (Point to plane ICP)*, dans laquelle, l'erreur est minimisée au long de la normale de la surface η_i, *c.-à-d.*, la projection de l'erreur $(\mathbf{R}_t \cdot q_i - p_i + \mathbf{t}_t)$ sur la normale à la surface η_i. Ceci modifiera la fonction de coût (3.4) pour qu'elle devienne :

$$\Lambda(S_{t-1}, S_t, \mathcal{T}_t) \triangleq \sum_i w_i \|(\mathbf{R}_t \cdot q_i - p_i + \mathbf{t}_t) \cdot \eta_i\|^2 \tag{3.5}$$

3.4.2 Alignement ICL – Iterative Closest Line

L'ICL ou l'algorithme *itératif de la ligne la plus proche (Iterative Closest Line)* [98] est une méthode qui reprend la même structure de l'ICP, mais qui opère sur des lignes au lieu de points.

L'ICL commence par l'extraction des lignes à partir des nuages de points (scans). Une ligne peut être définie par deux points, nous pouvons donc extraire C_n^2 combinaisons possibles de lignes à partir d'un nuage de points de taille n. Par conséquent, il est nécessaire d'utiliser des algorithmes qui réduisent la complexité de ce problème en utilisant des stratégies d'extraction plus sélectives (RANSAC, transformée de Hough, *etc.*).

Au cœur de l'ICL se trouve l'appariement de lignes provenant des deux nuages de points d'un LiDAR. La condition initiale de couplage des lignes est la distance qui les sépare lors de la première itération de ce processus.

Lorsque l'alignement ICL commence à converger, une condition de direction peut être introduite pour raffiner l'estimation. Cette condition se traduit par l'imposition d'une contrainte sur les lignes couplées qui doivent être « parallèles » dans un certain seuil.

À noter que le nombre des paires de lignes couplées est égal au nombre minimum des lignes extraites dans le nuage de points du scan actuel et dans le nuage de points de référence (un modèle représenté par le scan précédent ou par la carte). L'estimation de la transformation rigide entre le scan actuel et le nuage de points de référence peut être lancée une fois que les lignes extraites des deux nuages de points soient couplées.

Pour l'étape de l'estimation de la transformation rigide, deux formes de l'ICL ont été présentées par Alshawa [98], à savoir, une forme basée sur l'ICP et une forme alternative.

En reposant sur l'alignement des caractéristiques linéaires, la méthode ICL répond au problème d'appariement de scans d'environnement bien structurés. Elle n'est donc pas adaptée à des environnements qui ne sont pas de forme géométrique polygonale (objets circulaires, arbres, plantes, *etc.*).

3.4.3 Alignement IDC – Iterative Dual Correspondence

IDC *(Iterative Dual Matching)* [99] est un algorithme itératif d'appariement de scans composé de deux étapes. La première étape est prise de l'algorithme d'ICP, elle consiste à trouver, pour chaque point du balayage de référence, un point correspondant dans le balayage courant. Les correspondances sont des points dans chacun des deux balayages qui représentent les mêmes points physiques dans l'environnement.

La deuxième étape de l'IDC est prise de l'algorithme IMRP *(Iterative Matching Range Point)*, où chaque point du nouveau balayage correspond au point du balayage de référence qui a la même distance à partir de l'origine.

L'ensemble des points sont ensuite utilisés dans un processus de mise en correspondance et l'algorithme s'arrête lorsque les erreurs des moindres carrés sont suffisamment petites. Dans l'algorithme IDC, cette erreur des moindres carrés est définie de la même manière que celle de l'ICP (équation 3.4).

3.4.4 Alignement PM – Perfect Match

L'algorithme de la *correspondance parfaite (PM - Perfect Match)* est un algorithme léger d'alignement qui a été proposé par Lauer et al. [100]. Dans cet algorithme, la pose du robot est calculée à l'aide des distances 2D à l'environnement autour du robot, qui peuvent être acquises à partir d'un LiDAR ou d'une caméra.

L'objectif principal de l'algorithme est de minimiser l'erreur d'appariement, *c.-à-d.*, l'erreur l'alignement entre les données acquises et la carte de l'environnement.

L'algorithme PM se compose de trois étapes principales :

1. Le calcul de l'erreur d'appariement et du gradient ;

2. La fonction d'optimisation basée sur la *rétropropagation résiliente (RPROP - Resilient Back-Propagation)* ;
3. L'estimation de la covariance à l'aide de la dérivée seconde.

Dans le PM, une carte préalable de l'environnement est convertie en une carte de grille d'occupation *(OG - Occupancy Grid)*. Puis, à partir de cette grille d'occupation, il est possible de calculer la carte des distances et la carte de gradient. Pour la carte des distances, chaque cellule donne la distance à l'obstacle le plus proche. La carte de gradient comporte deux composantes, selon les axes des x et des y. Ces trois cartes (carte des distances, et les cartes de gradients selon x et y) sont statiques et peuvent être pré-calculées afin d'accélérer l'algorithme du traitement.

Considérons les points p_i d'un balayage LiDAR S_t représentés dans le référentiel global. La fonction de coût de l'algorithme PM est donnée par l'équation :

$$\Lambda(S_{t-1}, S_t, \mathcal{T}_t) \triangleq \sum_i e(d(\mathbf{t}_t + \mathbf{R}_t p_i)) \tag{3.6}$$

Où la fonction $d(\cdot)$ donne la valeur de la carte des distances évaluée pour un certain point. La formule (3.6) est un M-estimateur, où, au lieu d'utiliser une norme euclidienne (ℓ_2) qui n'est pas robuste en présence de données aberrantes *(outliers)*, le PM utilise la fonction $e(\cdot)$ qui pénalise les écarts entre le point mesuré par le LiDAR ($p_{t,i}$) et la carte préalable de l'environnement d'une façon plus robuste :

$$e(x) \triangleq 1 - \frac{c^2}{c^2 + x^2} \tag{3.7}$$

Enfin, la fonction d'erreur (3.6) est minimisée en utilisant l'optimiseur *rétropropagation résiliente (RPROP)*. L'algorithme PM a été appliqué à l'origine à la localisation à base d'une caméra [100], puis à l'alignement des balayages LiDAR [101].

3.4.5 Alignement corrélatif

Cette famille d'algorithmes a été proposée initialement par Olson [102]. Le problème a été formulé dans un cadre probabiliste, avec un espace modélisé en *grille d'occupation (Occupancy Grid)* dans le but de trouver la transformation de corps rigide qui maximise la probabilité d'observer les données du balayage.

Comme son nom l'indique, cette technique est basée sur la *corrélation croisée (Cross-correlation)*. Cette mesure statistique évalue la similarité entre deux signaux, x et y, et à un décalage temporaire spécifique τ, tel que [103] :

$$\Gamma_{xy}(\tau) \triangleq \sum_i x_i(t) \cdot y_i(t - \tau) \tag{3.8}$$

Plutôt que de faire confiance à une recherche locale pour trouver un maxima global, l'appariement corrélatif propose d'effectuer une recherche sur tout l'espace des transformations rigides plausibles. Ceci est réalisé par d'abord le choix d'une version adaptée aux données du LiDAR de l'équation 3.8, puis, appliquer celle-ci pour l'évaluation de la corrélation croisée entre les données du LiDAR et une région plausible de la carte. Cette région est dérivée d'une

prédiction dont la source serait la commande, l'odométrie des roues ou l'odométrie visuelle. La corrélation croisée donne une mesure de similarité, par conséquent, elle est appropriée à l'utilisation comme fonction de coût dans un processus d'optimisation.

Pour améliorer les performances de l'algorithme, la corrélation croisée est réalisée sur plusieurs résolutions, en commençant par une estimation grossière, puis la raffiner en passant sur des résolutions plus fines.

D'autres approches de la même famille ont été proposées par la suite, notamment celle de Vath et al. [104], qui, en plus de l'alignement multirésolutions, propose d'améliorer la qualité et la robustesse en utilisant un maximum d'informations possible, notamment en redéfinissant la fonction de coût pour aligner les polygones marqués par les points du LiDAR et la position du robot.

3.4.6 Alignement NDT – Normal Distribution Transform

La *transformation des distributions normales (NDT - Normal Distribution Transform)* a été introduite par Biber et al. [78] comme méthode d'appariement des balayages LiDAR 2D. Elle a été ensuite étendue pour l'appariement des balayages 3D par Magnusson et al. [105].

Cette méthode d'appariement crée une représentation lisse et continue de l'environnement à partir des données discrètes du LiDAR. Pour achever cela, la NDT modélise l'environnement par un ensemble de *fonctions de densité de probabilité (PDF - Probability Density Functions)* locales. Cette représentation est construite à partir d'un ensemble de points de référence qui sont regroupés en un ensemble de cellules (ou voxels dans le cas 3D) de taille fixe.

Dans les approches précédentes, des correspondances explicites doivent être établies entre le balayage précédent S_{t-1} (ou la carte de référence) et le balayage actuel S_t. Ceci peut être problématique lorsque les mesures sont trop bruitées, présentant des valeurs aberrantes *(outliers)* ou dans le cas d'un environnement non structuré. La NDT diffère sur ce point, dans le fait qu'elle ne nécessite pas d'établir une correspondance directe entre les primitives. Les points du balayage S_t ne sont pas comparés à des points du balayage S_{t-1}, mais à une représentation lisse de l'environnement sous forme d'un ensemble de fonctions de densité de probabilité, calculées à partir de ce balayage de référence.

L'algorithme NDT présente de nombreux avantages, en plus de sa représentation lisse, continue et compacte de l'environnement, la NDT est adaptée à plusieurs types d'environnements, dans la mesure où elle ne fait pas d'hypothèse sur la forme et/ou la régularité de l'environnement. Contrairement à l'ICP et l'IDC qui supposent que les points mesurés sont des points physiques dans l'environnement, ou l'IDL qui suppose que l'environnement est composé seulement de lignes et de formes polygonales. La NDT adopte une représentation probabiliste générique très adaptée à l'appariement des données bruitées [78, 80, 105, 106].

Ainsi, nous avons opté pour un algorithme basé sur la NDT pour réaliser l'appariement des données des LiDARs. Nous allons donc présenter l'appariement NDT plus en détail dans la section suivante.

3.5 Système proposé pour la localisation relative multi-LiDARs

Ayant une plateforme munie de deux LiDARs et de deux roues équipées de codeurs, nous avons proposé et implémenté une approche de localisation relative multi-LiDARs. Nous avons choisi un algorithme d'appariement des données LiDAR basé sur la NDT et l'*optimisation par essaims de particules (PSO - Particle Swarm Optimization)* [80, 106], que nous avons faiblement couplé avec les codeurs des roues en utilisant le filtre de Kalman étendu.

3.5.1 Modélisation du système

3.5.1.1 Le modèle cinématique

Notre plateforme est tractée grâce à deux roues motrices, nous décrierons donc la cinématique du robot par le modèle d'*entraînement différentiel* (*differential-drive*). Dans ce cas, les seuls paramètres que nous pouvons contrôler directement sont les vitesses de rotation des roues gauche et droite (v_l et v_r respectivement).

Nous définissons le vecteur $\begin{bmatrix} x & y & \phi \end{bmatrix}^\intercal$ qui représente l'état du robot (la position et l'orientation dans l'espace), L qui désigne la distance entre les deux roues motrices, et R qui représente le rayon de la roue (figure 3.9). Le modèle cinématique du robot dans ce cas est donné par :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = R \begin{bmatrix} \frac{1}{2}\cos\phi & \frac{1}{2}\cos\phi \\ \frac{1}{2}\sin\phi & \frac{1}{2}\sin\phi \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} \frac{R}{2}(v_l + v_r)\cos\phi \\ \frac{R}{2}(v_l + v_r)\sin\phi \\ \frac{R}{L}(v_r - v_l) \end{bmatrix} \tag{3.9}$$

Pour contrôler le robot, nous avons besoin d'un moyen de commander le robot en vitesses linéaire et angulaire (v et ω, respectivement). Pour ce faire, nous pouvons substituer les deux roues motrices par une seule roue *orientable* imaginaire (pilotable en vitesse de rotation et en vitesse de braquage), située au centre du train de traction (figure 3.9). Cela donne un modèle simplifié (équation 3.10), appelé *modèle monocycle*, qui est équivalent au modèle différentiel à deux roues.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v\cos\phi \\ v\sin\phi \\ \omega \end{bmatrix} \tag{3.10}$$

À partir des deux équations (3.9) et (3.10) nous pouvons exprimer les paramètres de sortie v_l et v_r en fonction des paramètres de commande v et ω :

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \frac{1}{2R} \begin{bmatrix} 2v - \omega L \\ 2v + \omega L \end{bmatrix} \tag{3.11}$$

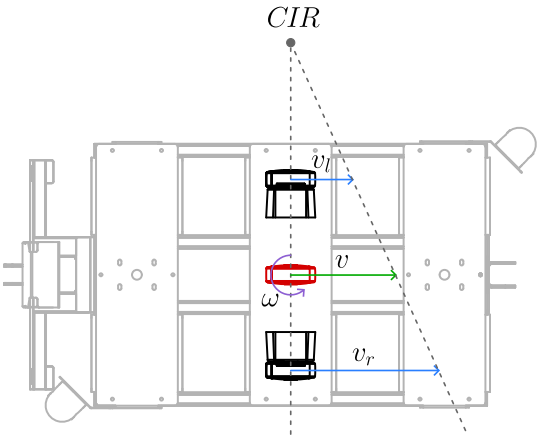


Figure 3.9 – L'équivalence entre le modèle d'*entraînement différentiel* et le modèle *monocycle* ; les deux vitesses v_l et v_r sont équivalentes à une vitesse linéaire v et une vitesse angulaire ω. Le CIR désigne le centre instantané de rotation.

Nous avons utilisé ce modèle (équation 3.11) pour implémenter le contrôleur du robot [107, 108] sous ROS [3] et ROS 2.

3.5.1.2 L'odométrie des roues

Le modèle cinématique présenté par l'équation (3.9) nous décrit l'évolution de la vitesse du robot au fil du temps, en fonction des vitesses mesurées des roues gauche et droite. En intégrant cette vitesse sur des petites unités de temps, nous pouvons déduire la pose du robot. Le résultat de cette intégration est appelée l'*odométrie* des roues, la figure 3.10 montre une illustration géométrique du modèle de l'odométrie.

La seule information que nous pouvons obtenir des codeurs des roues consiste en un compteur incrémental d'impulsions (*ticks*). Soit $\Delta\mathcal{Z}_t = (\mathcal{Z}_t - \mathcal{Z}_{t-1})$ la différence d'impulsions à une itération temporelle donnée t. Connaissant le rayon de la roue R et la résolution du codeur $\mathcal{R}$, la distance parcourue par le centre du robot, notée $\Delta\mathfrak{D}_t$, peut être calculée en utilisant les distances parcourues par les roues gauche et droite, ΔD_t^l et ΔD_t^r respectivement :

$$\Delta\mathfrak{D} = \frac{\Delta D_t^l + \Delta D_t^r}{2} \quad \text{avec} \quad \Delta D_t^{l|r} = \frac{2\pi R}{\mathcal{R}} \Delta\mathcal{Z}_t^{l|r} \tag{3.12}$$

Ainsi, le changement partiel de la pose $\Delta\mathbf{x}_t = [\Delta x_t \ \Delta y_t \ \Delta\phi_t]^\mathsf{T}$ peut être calculé à partir des distances ΔD_t^r et ΔD_t^l de l'équation (3.12) :

$$\Delta\mathbf{x}_t = \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\phi_t \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\phi_{t-1} & \frac{1}{2}\cos\phi_{t-1} \\ \frac{1}{2}\sin\phi_{t-1} & \frac{1}{2}\sin\phi_{t-1} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} \Delta D_t^l \\ \Delta D_t^r \end{bmatrix} = \begin{bmatrix} \Delta\mathfrak{D}_t \cos\phi_{t-1} \\ \Delta\mathfrak{D}_t \sin\phi_{t-1} \\ \dfrac{\Delta D_t^r - \Delta D_t^l}{L} \end{bmatrix} \tag{3.13}$$

3. ROS - Robot Operating System (`ros.org`), un méta-système d'exploitation qui est devenu le standard de facto des applications robotiques.

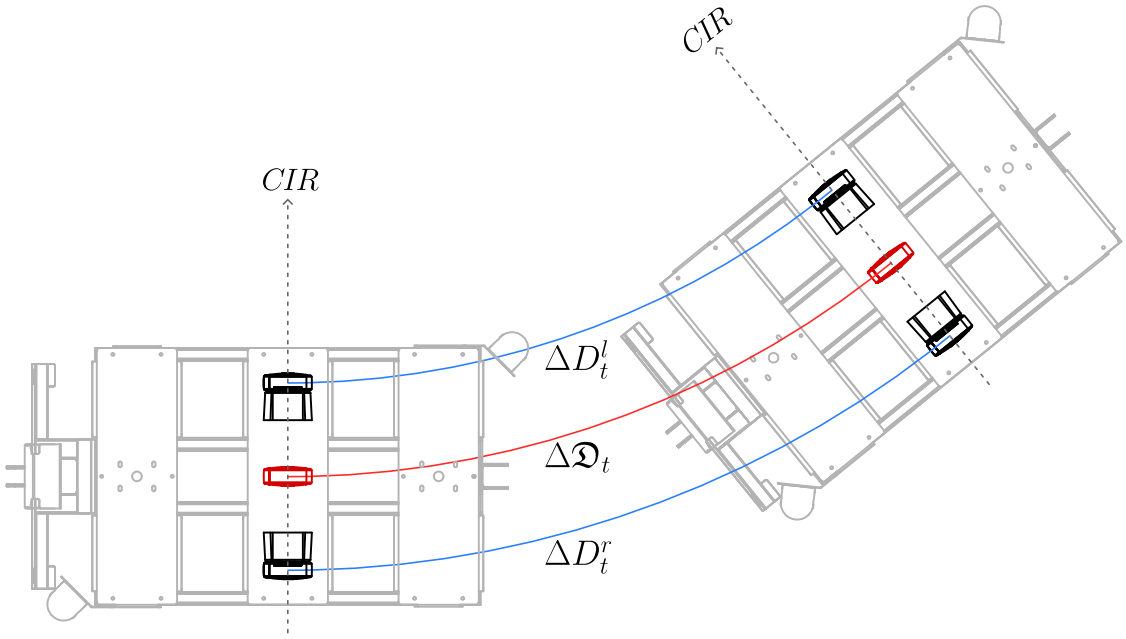


Figure 3.10 – Le modèle de l'odométrie, les roues gauche et droite traversent des distances ΔD_t^l et ΔD_t^r, respectivement. La roue centrale imaginaire traverse une distance moyenne $\Delta \mathfrak{D}_t$. Les deux axes qui passent par les centres des roues aux temps t et $t-1$ s'intersectent dans un point fixe nommé CIR (centre instantané de rotation).

À l'instant t, la pose du robot $\mathbf{x}_t = [x_t \; y_t \; \phi_t]^\mathsf{T}$ peut être déterminée en accumulant les changements partiels $\Delta \mathbf{x}_t$ jusqu'à t :

$$\begin{aligned} \mathbf{x}_t &= f(\mathbf{x}_{t-1}, \Delta D_t^l, \Delta D_t^r) \\ &= \mathbf{x}_{t-1} + \Delta \mathbf{x}_t \\ &= \begin{bmatrix} x_{t-1} + \Delta \mathfrak{D}_t \cos \phi_{t-1} \\ y_{t-1} + \Delta \mathfrak{D}_t \sin \phi_{t-1} \\ \phi_{t-1} + \dfrac{\Delta D_t^r - \Delta D_t^l}{L} \end{bmatrix} \end{aligned} \tag{3.14}$$

Nous utilisons le modèle décrit par l'équation (3.14) pour implémenter le calcul de l'odométrie dans notre paquet ROS destiné au contrôle du robot nommé `ezw_ros_controllers` [107, 108].

Dans un système à entraînement différentiel comme le nôtre, les roues motrices doivent être pilotées avec le même *profil de vitesse*, ce qui peut s'avérer difficile en raison des variations entre les formes des roues, les caractéristiques des moteurs [4] et les conditions du sol [11].

4. Bien que nous utilisions des moteurs identiques, le profil de vitesse d'un même moteur peut être différent selon s'il tourne dans le sens direct ou indirect.

En connaissant ces contraintes et en ajoutant les problèmes de glissement et de dérapage [5] des roues, l'état estimé uniquement à partir des données des codeurs des roues déviera rapidement de l'état réel du robot. Ainsi, l'odométrie constitue une mesure facile à calculer, utile pour des prédictions à court terme, mais peu fiable sur le long terme à cause de la dérive qu'elle cumule.

3.5.1.3 L'incertitude de l'odométrie

L'incertitude est la quantité qui caractérise la dispersion des valeurs attribuées à une mesure, elle permet ainsi de quantifier notre confiance en l'exactitude de la valeur mesurée. Cette quantité permet ensuite d'intégrer efficacement les données de l'odométrie dans un cadre de fusion de données.

Pour un vecteur d'état $\mathbf{x}_t$, représentant dans notre cas l'odométrie des roues, l'incertitude est modélisée par la variance de ce vecteur $Var[\mathbf{x}_t]$. Cette dernière peut être calculée de la manière suivante [110] :

$$\begin{aligned} Var[\mathbf{x}_t] &= Var[\mathbf{x}_{t-1} + \Delta\mathbf{x}_t] \\ &= Var[\mathbf{x}_{t-1}] + Var[\Delta\mathbf{x}_t] + 2Cov[\mathbf{x}_{t-1}, \Delta\mathbf{x}_t] \end{aligned} \tag{3.15}$$

Le terme $Var[\mathbf{x}_t]$ est récursif, il est donc nécessaire de l'initialiser en connaissant l'incertitude de la pose initiale, que nous pouvons supposer nulle $Var[\mathbf{x}_0] = \mathbf{0}$ pour une pose initiale parfaitement connue.

Le terme de la covariance croisée $Cov[\mathbf{x}_{t-1}, \Delta\mathbf{x}_t]$ décrit la contribution de la pose précédente $\mathbf{x}_{t-1}$ sur l'incertitude de l'incrément actuel $\Delta\mathbf{x}_t$ de l'état, ce terme peut être supposé nul si on accepte d'ignorer l'influence de l'erreur de ϕ_{t-1} sur le calcul de $\Delta\mathbf{x}_t$ (voir l'équation 3.13).

Le terme le plus important dans l'équation (3.15) est la variance de l'incrément actuel $Var[\Delta\mathbf{x}_t]$, qui peut être calculée par le théorème de König-Huygens :

$$Var[\Delta\mathbf{x}_t] = E[\Delta\mathbf{x}_t\Delta\mathbf{x}_t^\mathsf{T}] + E[\Delta\mathbf{x}_t]E[\Delta\mathbf{x}_t]^\mathsf{T} \tag{3.16}$$

$$\text{Avec} \quad E[\Delta\mathbf{x}_t\Delta\mathbf{x}_t^\mathsf{T}] = \begin{bmatrix} E[\Delta x_t^2] & E[\Delta x_t\Delta y_t] & E[\Delta x_t\Delta\phi_t] \\ E[\Delta y_t\Delta x_t] & E[\Delta y_t^2] & E[\Delta y_t\Delta\phi_t] \\ E[\Delta\phi_t\Delta x_t] & E[\Delta\phi_t\Delta y_t] & E[\Delta\phi_t^2] \end{bmatrix} \tag{3.17}$$

$$\text{Et} \quad E[\Delta\mathbf{x}_t]E[\Delta\mathbf{x}_t^\mathsf{T}] = \begin{bmatrix} E[\Delta x_t]^2 & E[\Delta x_t]E[\Delta y_t] & E[\Delta x_t]E[\Delta\phi_t] \\ E[\Delta y_t]E[\Delta x_t] & E[\Delta y_t]^2 & E[\Delta y_t]E[\Delta\phi_t] \\ E[\Delta\phi_t]E[\Delta x_t] & E[\Delta\phi_t]E[\Delta y_t] & E[\Delta\phi_t]^2 \end{bmatrix} \tag{3.18}$$

5. Le *« dérapage » (skidding)* est la perte de traction qui se produit lorsque le moteur applique beaucoup de puissance à la roue, par exemple, en suraccélération. Le *« glissement » (slipping)* est la perte de traction qui se produit lorsque le véhicule est en mouvement, mais qu'une roue est bloquée ou partiellement bloquée, par exemple, le patinage sur une surface à faible friction ou l'effet du freinage [109].

Les éléments de la matrice $E[\Delta\mathbf{x}_t\Delta\mathbf{x}_t^\mathsf{T}]$ sont donnés par [110] :

$$E[\Delta x_t^2] = \frac{k_6}{4}\left(\frac{1}{2} + \frac{1}{2}\cos(2\phi_{t-1})\exp\left(-2\sigma^2_{\phi_{t-1}}\right)\right) \tag{3.19}$$

$$E[\Delta x_t\Delta y_t] = E[\Delta y_t\Delta x_t] = \frac{k_6}{4}\left(\frac{1}{2}\sin(2\phi_{t-1})\exp\left(-2\sigma^2_{\phi_{t-1}}\right)\right) \tag{3.20}$$

$$E[\Delta x_t\Delta\phi_t] = E[\Delta\phi_t\Delta x_t] = k_7\cos(\phi_{t-1})\exp\left(-\frac{1}{2}\sigma^2_{\phi_{t-1}}\right) \tag{3.21}$$

$$E[\Delta y_t^2] = \frac{k_6}{4}\left(\frac{1}{2} - \frac{1}{2}\cos(2\phi_{t-1})\exp\left(-2\sigma^2_{\phi_{t-1}}\right)\right) \tag{3.22}$$

$$E[\Delta y_t\Delta\phi_t] = E[\Delta\phi_t\Delta y_t] = k_7\sin(\phi_{t-1})\exp\left(-\frac{1}{2}\sigma^2_{\phi_{t-1}}\right) \tag{3.23}$$

$$E[\Delta\phi_t\Delta\phi_t] = \frac{k_6}{L^2} \tag{3.24}$$

Avec :

$$k_1 = (\Delta D_t^r)^2 + (\Delta D_t^l)^2 \tag{3.25}$$

$$k_2 = (\Delta D_t^r)^2 - (\Delta D_t^l)^2 \tag{3.26}$$

$$k_3 = \Delta D_t^r\Delta D_t^l \tag{3.27}$$

$$k_4 = \sigma_r^2 + \sigma_l^2 \tag{3.28}$$

$$k_5 = \sigma_r^2 - \sigma_l^2 \tag{3.29}$$

$$k_6 = k_1 + 2k_3 + k_4 \tag{3.30}$$

$$k_7 = \frac{k_2 + k_5}{2L} \tag{3.31}$$

Ensuite, le deuxième terme de la matrice de la variance $Var[\Delta\mathbf{x}_t]$, donné par $E[\Delta\mathbf{x}_t]E[\Delta\mathbf{x}_t^\mathsf{T}]$ peut facilement être calculé à partir de :

$$E[\Delta\mathbf{x}_t] = \begin{bmatrix} \dfrac{\Delta D_t^l + \Delta D_t^r}{2}\cos(\phi_{t-1})\exp\left(-\frac{1}{2}\sigma^2_{\phi_{t-1}}\right) \\ \dfrac{\Delta D_t^l + \Delta D_t^r}{2}\sin(\phi_{t-1})\exp\left(-\frac{1}{2}\sigma^2_{\phi_{t-1}}\right) \\ \dfrac{\Delta D_t^r - \Delta D_t^l}{L} \end{bmatrix} \tag{3.32}$$

Les deux mesures directes dans l'odométrie sont $\Delta\mathcal{Z}_t^l$ et $\Delta\mathcal{Z}_t^r$, les lectures des codeurs gauche et droit, respectivement. Ces mesures peuvent comporter des erreurs relatives aux glissements, aux dérapages, aux déformations des pneus, *etc.* Pour chacun des codeurs, nous définissons les erreurs absolues des codeurs gauche et droit ϵ_l et ϵ_r, qui représentent l'écart entre la valeur réelle $\Delta\mathcal{Z}_t$ et la valeur mesurée $\Delta\hat{\mathcal{Z}}_t$.

$$\Delta\hat{\mathcal{Z}}_t = \Delta\mathcal{Z}_t + \epsilon \qquad \text{Avec } \epsilon \sim \mathcal{N}(0, \sigma^2_{\mathcal{Z}_t}) \tag{3.33}$$

Cette erreur peut ainsi être liée, à l'aide de la formule de l'équation (3.12), aux erreurs sur les distances parcourues par la roue gauche ΔD^l_t et la roue droite ΔD^r_t, de distributions normales $\mathcal{N}(0, \sigma^2_l)$ et $\mathcal{N}(0, \sigma^2_r)$, respectivement.

3.5.1.4 Les données du LiDAR

Le LiDAR fournit une liste de mesures de distances, qui contient dans notre cas 541 éléments ; chacun de ces éléments représente une distance mesurée à un angle allant de $-135°$ à $135°$, avec une résolution de $0.5°$.

Soit $S = \{\rho_0, \rho_1, \ldots, \rho_N\}$ la liste des distances mesurées par le LiDAR. Nous pouvons passer de cette liste de mesures à une liste de points bidimensionnels (2D) en coordonnées polaires, le point u_n correspondant au n-ième élément de la liste S tel que :

$$u_n = \begin{bmatrix} \rho_n \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \rho_n \\ -135° + n \times 0.5° \end{bmatrix} \qquad \text{avec} \qquad 0 \leq n < 541 \tag{3.34}$$

Pour notre algorithme, nous devons convertir les points des coordonnées polaires (u_n) en coordonnées cartésiennes (p_n) en utilisant l'équation (3.1). Ainsi, nous représentons chaque balayage du laser comme une collection de 541 points bidimensionnels.

3.5.2 La méthode d'alignement

Ayant une plateforme avec un odomètre et deux LiDARs, nous avons voulu exploiter ces données pour localiser le robot. Nous nous sommes basés sur une méthode d'alignement de scans laser nommé NDT-PSO [6] [80, 106]. Cette méthode est basée sur la *« Transformée de distribution normale » (NDT - Normal Distribution Transform)* [78] et sur la méthode d'optimisation bio-inspirée nommée *« Optimisation par essaims de particules » (PSO - Particle Swarm Optimization)* [111].

Notre choix découle des résultats de nos précédents travaux [80, 106]. En effet, comme nous l'avons exposé précédemment dans ce chapitre, la méthode NDT offre une représentation de l'environnement qui pourrait être intéressante pour l'élaboration d'un système de localisation générique sans recourir à des hypothèses spécifiques concernant la géométrie de l'environnement.

Pour calculer les transformations optimales, la méthode NDT originale [78] se base sur l'optimisation de Newton. En revanche, cela la laisse vulnérable aux problèmes des minimas locaux, notamment sur des trajectoires complexes avec beaucoup de changements d'orientation. Ceci a donné naissance à la méthode NDT-PSO que nous avons proposé auparavant [80, 106]. Cette dernière utilise l'algorithme PSO pour réaliser la tâche d'optimisation. Étant un algorithme stochastique, PSO permet de surmonter tant que possible les problèmes liés aux minimas locaux [80, 106].

6. Le code source de la méthode NDT-PSO est disponible publiquement sur GitHub sous licence GPL-2.0 `abougouffa/ndtpso_slam`.

Le système que nous proposons ici se compose d'un nœud de calcul pour l'odométrie, deux nœuds pour d'alignement NDT-PSO (un nœud par LiDAR) et un nœud qui effectue une fusion de données basée sur le *filtre de Kalman étendu (EKF)*. Dans l'EKF, nous utilisons l'odométrie comme donnée de prédiction, et les données des LiDARs traitées par NDT-PSO pour apporter des corrections et mettre à jour l'état.

3.5.2.1 Modélisation de l'environnement

En se basant sur la NDT-PSO, l'environnement est modélisé par la *transformation de distribution normale (NDT)* [78]. La première étape de cette représentation consiste à diviser l'environnement en un ensemble de cellules de dimensions connues (1m × 1m par exemple). Ensuite, le nuage de points bidimensionnels contenus dans chaque cellule « c » est représenté sous forme d'une distribution normale $\mathcal{N}_c(\mu_c, \Sigma_c)$. Pour ce faire, nous prenons comme échantillon l'ensemble des « n » points 2D $p_{i=1\ldots n} = [x_i\ y_i]^\mathsf{T}$ contenus dans une cellule « c », et nous calculons les paramètres de la distribution normale $\mathcal{N}_c(\mu_c, \Sigma_c)$ à partir de cet échantillon, à savoir, les moments d'ordre 1 et 2 :

$$\mu_c = \frac{1}{n}\sum_{i=1}^{n} p_i \quad \text{et} \quad \Sigma_c = \frac{1}{n}\sum_{i=1}^{n}(p_i - \mu_c)(p_i - \mu_c)^\mathsf{T} \tag{3.35}$$

Notons que pour des points 2D, la moyenne $\mu_c \in \mathbb{R}^2$ et la matrice de covariance $\Sigma_c \in \mathbb{R}^2 \times \mathbb{R}^2$.

L'ensemble des distributions normales de chaque cellule ($\mathcal{N}_c$) représente la carte NDT locale. Cette représentation ressemble à la représentation en grille d'occupation [112]. Cependant, en NDT, la probabilité n'est pas la même pour toute la cellule, à la place, la NDT associe une fonction de densité de probabilité (PDF) à chaque cellule, la figure 3.11 visualise cette représentation. La probabilité de mesurer un échantillon d'un point bidimensionnel (p) dans une cellule c est donnée par la PDF « Π_c » telle que :

$$\Pi_c(p) = \mathcal{C} \exp\left(-\frac{(p-\mu_c)^\mathsf{T}\Sigma_c^{-1}(p-\mu_c)}{2}\right) \tag{3.36}$$

Notez que, dans une distribution normale multivariée à k dimensions, la constante de normalisation $\mathcal{C}$ est égale à $((2\pi)^{-\frac{k}{2}})\cdot\det(\Sigma)^{-\frac{1}{2}})$. Néanmoins, dans le processus d'optimisation, nous n'avons pas besoin de normaliser les probabilités. Par conséquent, $\mathcal{C}$ est mis à 1 dans l'expression (3.36).

3.5.2.2 Alignement des scans et estimation de la pose

Soit $[\Delta x\ \Delta y]^\mathsf{T}$ la translation entre deux balayages laser et $\Delta\phi$ le changement d'orientation entre ces derniers. La transformation spatiale $\mathcal{T}(\mathbf{x}_{t-1}, \Delta\mathbf{x}_t)$ entre deux poses du robot aux temps $t-1$ à t est :

$$\mathcal{T} : \begin{bmatrix} x_t \\ y_t \\ \phi_t \end{bmatrix} = \begin{bmatrix} \cos\Delta\phi_t & -\sin\Delta\phi_t & 0 \\ \sin\Delta\phi_t & \cos\Delta\phi_t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \phi_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta x_t \\ \Delta y_t \\ \Delta\phi_t \end{bmatrix} \tag{3.37}$$

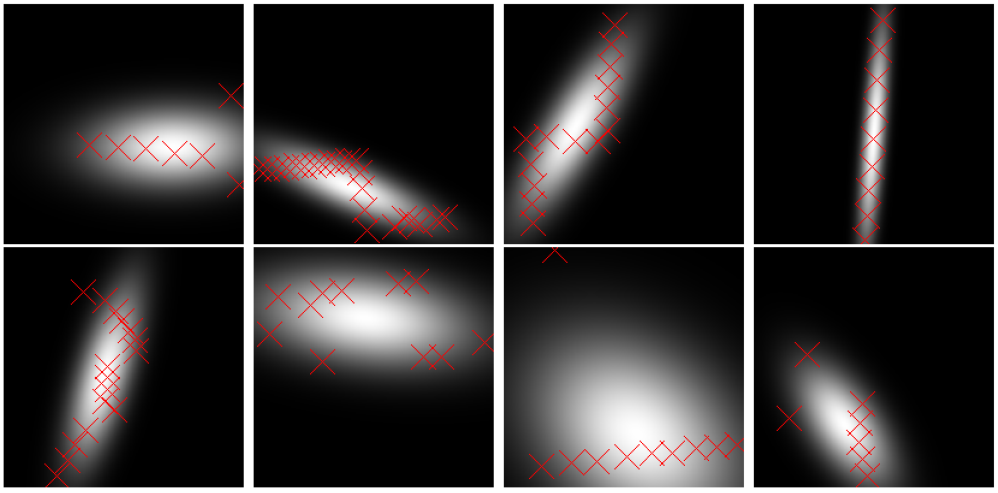


Figure 3.11 – Exemple de la représentation NDT de quelques cellules. Les points bidimensionnels mesurés sont représentés par des croix rouges. Ces points sont utilisés pour calculer la distribution normale. Les niveaux de gris représentent la PDF qui associe à chaque point de la cellule une probablilité de mesurer un point physique (tirée de [78]).

$$\text{Avec} \quad \begin{cases} \mathbf{x}_{t-1} & = [x_{t-1}\ y_{t-1}\ \phi_{t-1}]^\mathsf{T} \\ \Delta\mathbf{x}_t & = [\Delta x_t\ \Delta y_t\ \Delta\phi_t]^\mathsf{T} \end{cases}$$

Étant donné deux scans aux instants t et $t-1$, le but de l'*alignement des scans* est de trouver les paramètres de transformation Δx_t, Δy_t et $\Delta\phi_t$ permettent de superposer les parties communes des deux scans.

Pour un scan de N points bidimensionnels $\mathbf{p}_{i=1\ldots N}$, la fonction objective Λ qui évalue le paramètre $\Delta\mathbf{x} = [\Delta x\ \Delta y\ \Delta\phi]^\mathsf{T}$ est donnée par [78] :

$$\Lambda(\Delta\mathbf{x}) = -\sum_{i=1}^{N} \Pi_c(\mathcal{T}(\mathbf{p}_i, \Delta\mathbf{x})) \tag{3.38}$$

Dans la NDT originale [78], le $\Delta\mathbf{x}$ est optimisé en utilisant la fonction objective Λ avec la méthode de Newton. La méthode NDT-PSO utilise plutôt une optimisation basée sur PSO qui permet une meilleure convergence globale [80, 106].

L'optimisation PSO [111] commence par initialiser un essaim de M particules en positions $\mathbf{x}^i \in \mathbb{R}^k$ avec $i = \{1, 2, \cdots, M\}$ et k la dimension du vecteur $\mathbf{x}^i$. Les positions sont initialisées d'une manière uniformément aléatoire dans un espace de recherche choisi, ou dans tout l'espace de recherche. Dans notre cas, la position de la particule est représentée par un vecteur de tridimensionnel $\mathbf{x}^i = [\Delta x_i\ \Delta y_i\ \Delta\phi_i]^\mathsf{T}$ qui représente l'incrément de la pose entre deux balayages du LiDAR.

À chaque itération de temps t, nous calculons pour chaque particule une nouvelle vitesse $\mathbf{v}^i_{t+1}$ en fonction de sa vitesse précédente.

$$\mathbf{v}_{t+1}^{i} = w\mathbf{v}_{t}^{i} + c_1 \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_3 \end{bmatrix} (\mathcal{P}_{t}^{i} - \mathbf{x}_{t}^{i}) + c_2 \begin{bmatrix} r_4 & 0 & 0 \\ 0 & r_5 & 0 \\ 0 & 0 & r_6 \end{bmatrix} (\mathcal{G}_{t} - \mathbf{x}_{t}^{i}) \tag{3.39}$$

De même, nous calculons la mise à jour de la position (l'incrément d'état) $\mathbf{x}_{t+1}^{i}$ de la particule tel que :

$$\mathbf{x}_{t+1}^{i} = \mathbf{x}_{t}^{i} + \mathbf{v}_{t+1}^{i} \tag{3.40}$$

Avec w, c_1 et, c_2 sont les constantes inertielles, cognitives et sociales de la PSO, respectivement. Les éléments $(r_1, r_2, r_3, r_4, r_5, r_6)$ sont des nombres aléatoires positifs dans l'intervalle $[0, 1]$. Le $\mathcal{P}_{t}^{i}$ est le meilleur résultat personnel *(personal best)* de la particule « i » et le $\mathcal{G}_t$ est le meilleur résultat global *(global best)* de tout l'essaim.

La PSO permet de faire évoluer tous les éléments de l'essaim d'une façon collective. Ainsi, à chaque itération t, la nouvelle position (estimation) calculée pour chaque particule i prend en compte la mémoire à court terme (apport personnel) de la particule, *c.-à-d.*, sa vitesse précédente $\mathbf{v}_{t}^{i}$; la mémoire à long terme (apport cognitif) de la particule, *c.-à-d.*, son meilleur résultat personnel $\mathcal{P}_{t}^{i}$; et la mémoire de tout l'essaim (apport social), *c.-à-d.*, le meilleur résultat global $\mathcal{G}_t$. La somme des trois est pondérée par les facteurs aléatoires. La figure 3.12 illustre graphiquement ce concept.

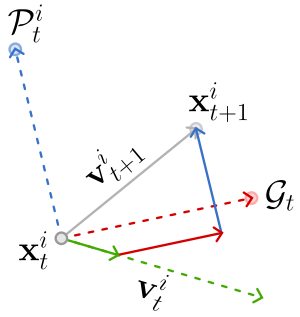


Figure 3.12 – Principe de la PSO, montrant l'apport personnel, cognitif et social à chaque itération.

En utilisant la fonction objective (équation 3.38), nous pouvons sélectionner les meilleurs résultats personnels $\mathcal{P}_{t+1}^{i}$ et globaux $\mathcal{G}_{t+1}$ comme suit :

$$\mathcal{P}_{t+1}^{i} = \begin{cases} \mathbf{x}_{t+1}^{i} & \text{si} \quad \Lambda(\mathbf{x}_{t+1}^{i}) < \Lambda(\mathcal{P}_{t}^{i}) \\ \mathcal{P}_{t}^{i} & \text{sinon} \end{cases} \tag{3.41}$$

$$\mathcal{G}_{t+1} = \begin{cases} \mathcal{P}_{t+1}^{i} & \text{si} \quad \Lambda(\mathcal{P}_{t+1}^{i}) < \Lambda(\mathcal{G}_{t}) \\ \mathcal{G}_{t} & \text{sinon} \end{cases} \tag{3.42}$$

Avec tous ces éléments, l'appariement à base de la NDT-PSO s'exécute de la manière suivante :

1. À un instant t, diviser le scan $t-1$ en cellules (de 1m $\times$ 1m), puis calculer la moyenne et la covariance pour chaque cellule comme présenté dans l'équation (3.35) ;

2. Convertir le balayage laser U_k^t en un nuage de points bidimensionnels p_k^t en utilisant les équations (3.34) et (3.1) ;

3. Initialiser aléatoirement l'essaim de particules dans un espace de recherche choisi qui peut être fixe, initialisé à partir du $\Delta\mathbf{x}_{t-1}$ ou initialisé à partir de l'odométrie ;

4. Pour chaque particule « i » :

   (a) Calculer la vitesse et la position de la particule en utilisant les équations (3.39) et (3.40) ;

   (b) Transformer les échantillons p_k du référentiel du scan t au référentiel du scan $t-1$ en utilisant $\mathcal{T}(p_k, \mathbf{x}^i)$ de l'équation (3.37) ;

   (c) Utiliser la fonction objective (3.38) pour déterminer les meilleures performances personnelles et globales ($\mathcal{P}_t^i$ et $\mathcal{G}_t$ respectivement) de la particule, comme présenté dans (3.41) et (3.42) ;

   (d) Répéter à partir de l'étape (4) jusqu'à la convergence ou pour un nombre fixe d'itérations.

5. Le meilleur résultat global $\mathcal{G}_t$ est sélectionné comme étant le changement de la pose du robot entre les instants $t-1$ et t.

À la fin de l'itération, la pose est mise à jour en utilisant le meilleur résultat global comme suit :

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathcal{G}_t \tag{3.43}$$

Notre implémentation de la NDT-PSO utilise une fenêtre temporelle pour chaque cellule, implémentée par un tampon circulaire de points à l'intérieur de la cellule. Avec un tampon circulaire de taille F, nous gardons les points issus des F derniers scans qui tombent dans la cellule.

Cette technique nous permet de limiter le nombre de points stockés par cellule, ce qui fait que l'utilisation de la mémoire ne dépend que de la taille de la carte. Cela permet également de réduire le temps de calcul lors du calcul des paramètres de la distribution normale, notamment lors du calcul de la matrice de la covariance.

La limitation du nombre de points est très importante lorsque l'algorithme s'exécute pendant une longue période. De plus, comme les points anciens sont tous le temps remplacés par des nouvelles observations, cette fenêtre temporelle permet d'avoir un filtre qui permet d'éliminer des objets mobiles de la carte sans un mécanisme de détection et de filtrage classique (*fig.* 3.13).

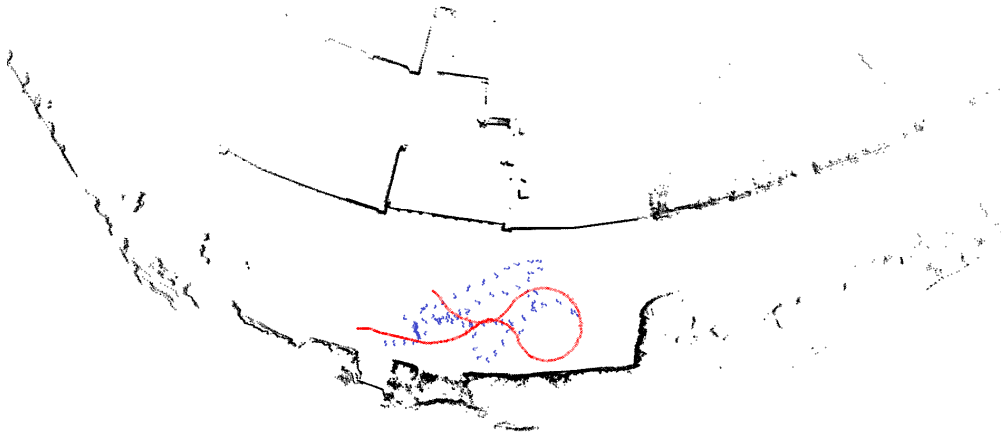


Figure 3.13 – Carte reconstruite par NDT-PSO en noir, la trajectoire en rouge, les mesures correspondantes aux objets dynamiques en bleu (tirée de [80]).

3.5.3 Fusion par filtre de Kalman étendu

Dans une majorité des cas d'utilisation, l'approche à suivre avec un système multi-LiDARs est de combiner les données issues des deux LiDARs pour construire un scan plus complet, puis effectuer l'alignement sur ce scan combiné (architecture illustrée sur la figure 3.14).

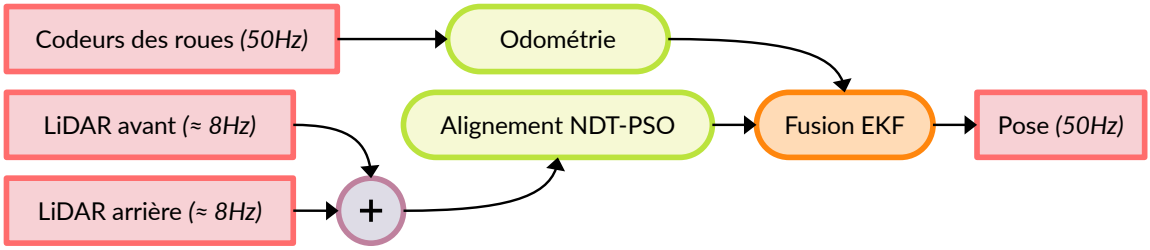


Figure 3.14 – Architecture à base de combinaison à priori des scans des deux LiDARs.

En raison des imperfections structurelles de la plateforme, l'installation des deux scanners laser dans le même plan horizontal peut s'avérer une tâche délicate. En effet, même la présence d'une petite différence d'orientation entre les plans des deux LiDARs peut entraîner des différences plus remarquables dans les distances mesurées (voir figure 3.1). Sans un calibrage physique très précis, les LiDARs avant et arrière peuvent voir des plans légèrement décalés et/ou inclinés l'un par rapport à l'autre. Ceci peut entraîner des incohérences lors de la mise en correspondance des scans du LiDAR de l'avant avec ceux obtenus à partir du LiDAR du derrière, ce qui entraîne des incertitudes plus importantes dans les scans appariés.

En plus, l'utilisation d'une stratégie de combinaison préalable des scans suppose que les LiDARs sont parfaitement synchronisés et que les mesures des deux LiDARs sont réalisées au même moment. Cependant, cette contrainte s'avère difficile à satisfaire avec des LiDARs du commerce, car ce type de synchronisation doit être assuré au niveau des deux capteurs.

L'alignement des données de chaque LiDAR séparément peut assurer une sorte de *cohérence locale* entre les différents scans du même LiDAR. Ensuite, les poses estimées en utilisant les données de chaque LiDAR peuvent être fusionnées pour obtenir une meilleure estimation globale de la pose.

De plus, étant donné que les deux LiDARs sont complètement indépendants, l'utilisation d'une architecture de fusion distribuée (figure 3.15) peut fournir des estimations de pose, même en cas d'échec d'un des LiDARs. Ceci permet d'assurer un niveau minimal de résilience aux pannes, ce qui constitue une contrainte extrêmement importante pour les systèmes de navigation sûre. D'ailleurs, cette architecture distribuée ne suppose aucune synchronisation entre les deux LiDARs, permettant ainsi de fonctionner indépendamment des fréquences d'échantillonnage des deux LiDARs.

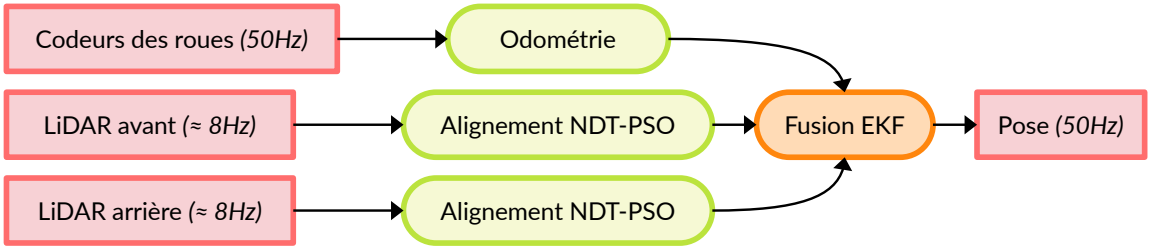


Figure 3.15 – L'architecture de fusion distribuée proposée pour notre *SmartTrolley* [94].

L'expression $\mathbf{x}_t = f(\mathbf{x}_{t-1})$ dans le modèle odométrique (3.14) étant non-linéaire ; nous avons choisi d'utiliser un filtre de KALMAN étendu (EKF) pour la fusion.

Il convient de noter que dans nos tests préliminaires, nous avons expérimenté à la fois l'EKF et le *filtre à particules (PF - Particle Filter)*. Nous avons opté pour l'EKF car il est beaucoup plus simple et rapide que le PF. D'ailleurs, nos tests sur le PF n'ont montré aucun gain notable de précision par rapport à l'EKF. En outre, vu que dans notre architecture (*fig.* 3.15), la fusion est effectuée sur le résultat de la NDT-PSO, qui elle-même est une approche à base d'une optimisation stochastique (PSO), le PF ne semble en effet pas adéquat à notre système.

3.5.3.1 Phase de prédiction

Pour l'étape de prédiction, nous utilisons l'odométrie issue du système proprioceptif des codeurs des roues. La prédiction de l'état $\hat{\mathbf{x}}_{k|k-1}$ et son incertitude associée $\mathbf{P}_{k|k-1}$ au temps k étant donné toutes les mesures jusqu'à $k-1$ sont donnés par :

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}) + q_{k-1} \tag{3.44}$$

$$\mathbf{P}_{k|k-1} = f'(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}f'(\hat{\mathbf{x}}_{k-1|k-1})^{\mathsf{T}} + \mathbf{Q}_{k-1} \tag{3.45}$$

Avec $q_{k-1} \sim \mathcal{N}(0, \mathbf{Q}_{k-1})$ est le bruit gaussien du processus ; f est l'expression de l'odométrie définie dans (3.14) et f' est son premier terme du développement de TAYLOR qui peut être exprimé comme suit :

$$f'(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} \approx \begin{bmatrix} 1 & 0 & -\Delta\mathfrak{D}\sin\phi \\ 0 & 1 & \Delta\mathfrak{D}\cos\phi \\ 0 & 0 & 1 \end{bmatrix} \tag{3.46}$$

3.5.3.2 Phase de correction

Les paramètres utilisés pour la NDT-PSO ont été choisi empiriquement, ainsi, la taille des cellules NDT a été fixée à 1m × 1m, ce qui offre un bon compromis entre les détails représentés dans chaque cellule et le nombre de cellules valides [80]. L'étape de l'optimisation est réalisée grâce à la PSO avec une population de 30 particules qui évoluent pendant 50 itérations. Cette combinaison de paramètres donne en effet un bon compromis entre la précision et le temps de calcul dans notre cas d'utilisation [80].

À chaque itération, nous exécutons NDT-PSO sur les données des LiDARs de l'avant et du derrière. Chacun donne une observation (pose) exprimée dans le même référentiel. L'étape finale consiste à fusionner ces deux observations en utilisant l'état prédit. À chaque fois que nous calculons une pose avec NDT-PSO, nous appliquons la mise à jour de l'état avec l'observation reçue en utilisant les formules suivantes :

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_{k|k-1})) \tag{3.47}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^{\mathsf{T}} \tag{3.48}$$

Avec $\mathbf{S}_k$ est la covariance prédite de la mesure $\mathbf{y}_k$, $\mathbf{K}_k$ est le gain de Kalman et $\mathbf{R}_k$ est la covariance du bruit gaussien de mesure.

$$\mathbf{S}_k = h'(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}h'(\hat{\mathbf{x}}_{k|k-1})^{\mathsf{T}} + \mathbf{R}_k \tag{3.49}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}h'(\hat{\mathbf{x}}_{k|k-1})^{\mathsf{T}}\mathbf{S}_k^{-1} \tag{3.50}$$

Notons que nous n'avons pas besoin d'un modèle d'observation dans notre cas car les poses sont déjà calculées à l'aide de la NDT-PSO (qui constitue le modèle d'observation dans notre système). Ainsi, nous mettons $h'(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}$ avec $\mathbf{H} = I_3$.

3.6 Expérimentation et résultats

3.6.1 Plateforme et environnement de l'éxpérience

Nous avons utilisé la plateforme *SmartTrolley* (figure 3.16) pour valider notre algorithme de fusion. Avec une architecture logicielle composée de deux nœuds ROS de la NDT-PSO, un nœud odométrique et le nœud de fusion EKF.

Lors de l'exécution de notre implémentation de la NDT-PSO pour un seul LiDAR, la fréquence de traitement de l'algorithme est d'environ 40Hz alors que le LiDAR est configuré pour publier des données à une fréquence de 8Hz. Cependant, lors de l'exécution des deux nœuds NDT-PSO simultanément, la fréquence de d'alignement globale chute à environ 9Hz.

En revanche, cela ne pose pas de problème dans notre cas car le système proposé peut fournir des estimations à la sortie de la fusion à la cadence la plus rapide, à savoir celle de l'odométrie à 50Hz. Le système réalise ensuite des corrections à une cadence maximale de 9Hz, ce qui est largement suffisant pour une exécution en temps réel. D'autant plus que que le robot ne fonctionne qu'aux basses vitesses (la vitesse maximale de déplacement est $v_{max} \approx 1.1\text{m} \cdot \text{s}^{-1} \approx 4\text{km} \cdot \text{h}^{-1}$).

Figure 3.16 – Le premier prototype de notre plateforme mobile expérimentale *SmartTrolley* dans la salle d'essai.

Pour valider notre plateforme expérimentale et notre système de fusion, nous avons réalisé une expérience dans les locaux de l'entreprise ez-Wheel à La Couronne, dans la salle montrée sur la figure 3.17. Cette dernière mesure 11m de longueur et 6.7m de largeur dans la région la plus large et 4.25m dans la plus étroite. Pendants nos tests, nous avons piloté le robot à distance à l'aide d'une manette sans fil de *Xbox*.

3.6.2 Résultats et discussion

Deux séquences ont été enregistrées, chacune commence avec le *SmartTrolley* à la position indiquée par une étoile rouge dans les deux figures 3.18 et 3.20. Le robot a parcouru une distance de 4.5m dans la première séquence et de 2.5m dans la deuxième séquence, avant d'effectuer quelques rotations sur lui-même dans le sens direct et dans le sens inverse, et puis, il revient au près de sa position de départ en marche arrière.

Les figures 3.19 et 3.21 montrent, à partir des séquences 1 et 2, la trajectoire odométrique, les deux trajectoires NDT-PSO générées à partir des LiDARs de l'avant et du derrière, ainsi que les cartes associées tracées à partir des données de ces derniers. Les bords peu réguliers

en haut à droite des cartes 3.19 et 3.21 ne sont pas dus à des imprécisions de calcul, mais à la forme réelle de l'environnement. En effet, cette zone correspond dans l'environnement réel à des chariots avec habillage en grillage, visibles sur la droite de la figure 3.17.

Figure 3.17 – Le *SmartTrolley* dans la salle d'essai, la région étroite étant au fond de l'image.

Nous pouvons remarquer qu'au début, l'odométrie était proche des poses estimées à partir des deux LiDARs, mais qu'elle s'est écartée de celles-ci au fil du temps. À la fin de l'expérience, la distance entre la position réelle du robot et le point estimé grace à l'odométrie était de 0.75m sur la première séquence. Cet effet est bien connu pour l'odométrie des roues. Le couplage mécanique, les glissements et les dérapages des roues génèrent des erreurs à chaque itération de calcul. Ces erreurs s'accumulent au fil du temps, conduisant finalement à une dérive de l'estimation à moyen ou long terme.

Cette tendance est particulièrement prononcée dans le cas de la trajectoire odométrique de la deuxième séquence (comme illustré dans la figure 3.20), où une erreur beaucoup plus significative s'est accumulée en raison d'un important dérapage constaté sur l'une des deux roues.

Toutefois, il convient de noter que cette accumulation d'erreur n'a pas eu d'impact perturbant sur l'estimation obtenue à la sortie de la fusion EKF, puisque dans ce processus, l'odométrie est utilisée pour la prédiction de manière relative. En d'autres termes, seuls les mouvements relatifs sont pris en compte, et non pas la somme cumulatée de ces mouvements. Ceci peut être confirmé en observant la carte générée sur la figure 3.21, l'absence de décalage important entre les scans d'un même LiDAR montre que l'estimation à partir des LiDARs n'a pas été biaisée par l'erreur de l'odométrie.

Nous pouvons également observer un petit décalage entre les deux cartes estimées à partir des données des LiDARs d'avant et de derrière. Ceci est dû aux imperfections structurelles discutées précédemment, ce décalage peut également affecter les poses associées à chaque carte.

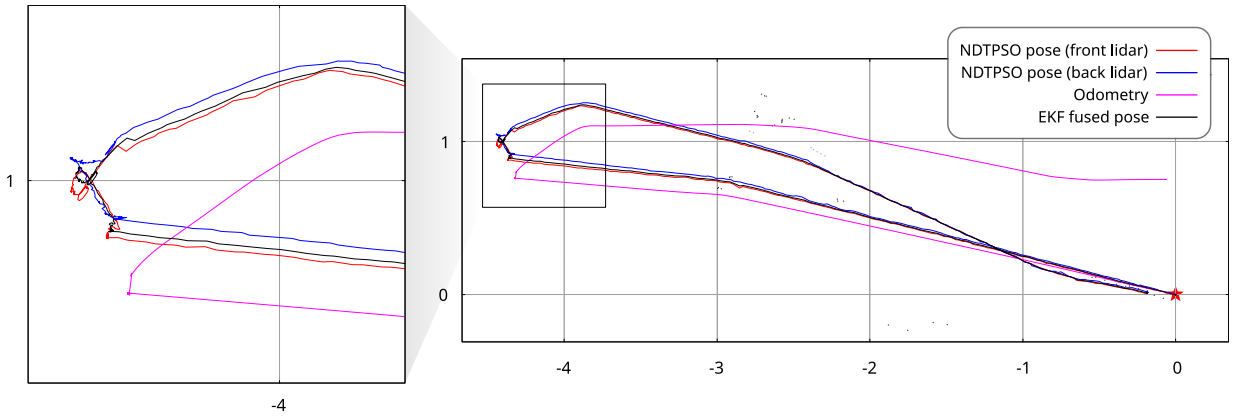


Figure 3.18 – La trajectoire de la première séquence estimée par la fusion EKF, ainsi que les données d'entrée de l'odométrie et des deux poses calculées à partir des données des LiDARs (distances en *mètres*). La position de départ est située au point $(0, 0)$ marqué d'une étoile rouge.

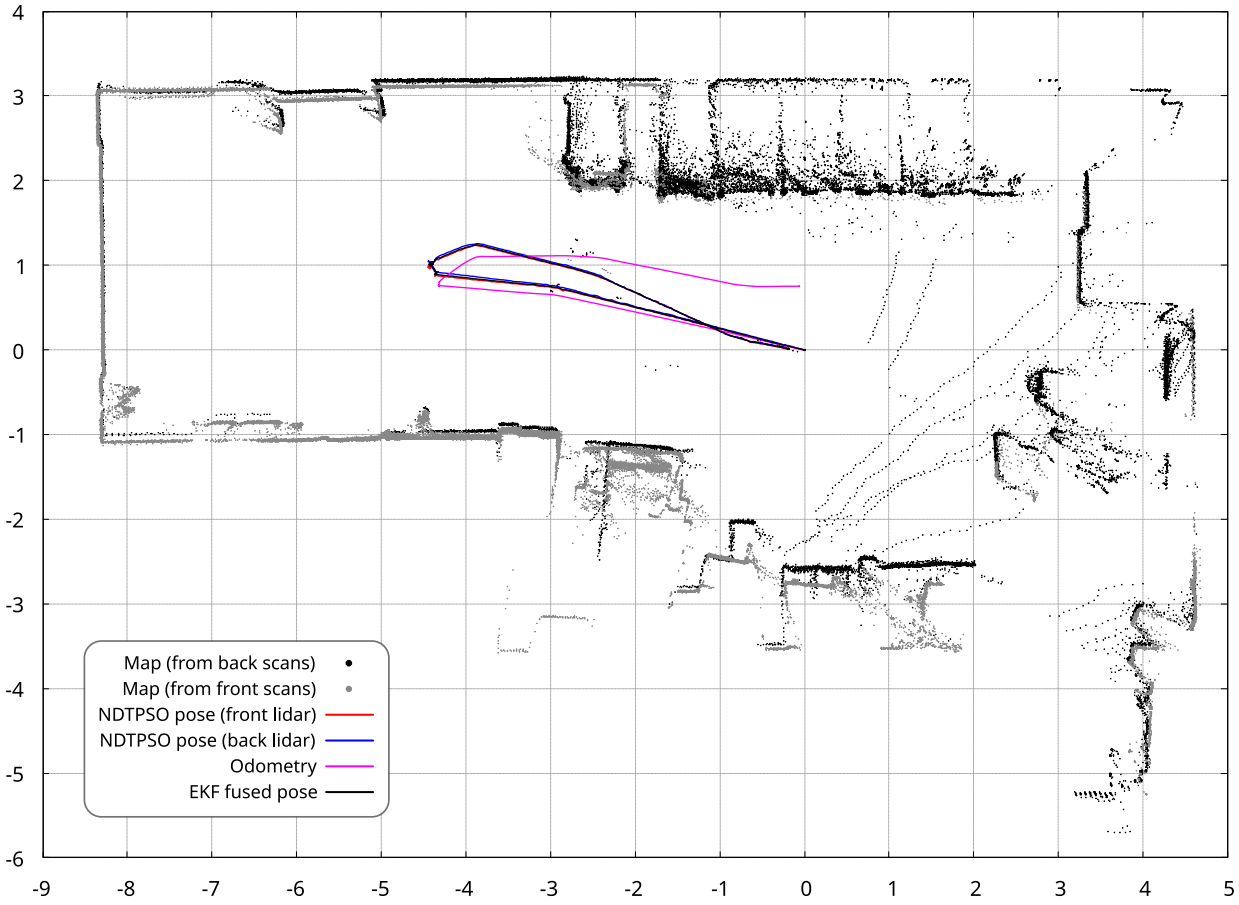


Figure 3.19 – Les cartes générées par l'NDT-PSO à partir des deux LiDARs sur la première séquence, avec les poses associées et l'odométrie (distances en *mètres*).

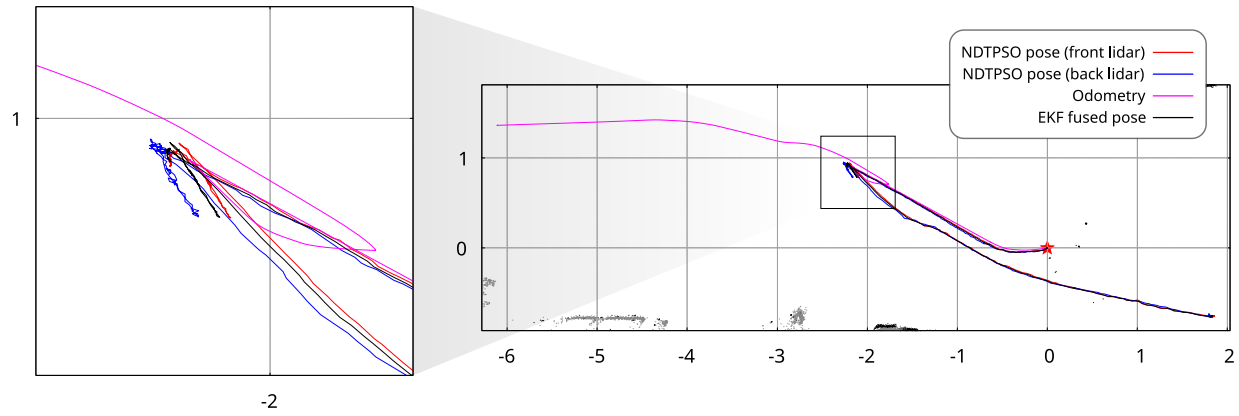


Figure 3.20 – La trajectoire de la deuxième séquence estimée par la fusion EKF, ainsi que les données d'entrée de l'odométrie et des deux poses calculées à partir des données des LiDARs (distances en *mètres*). La position de départ est située au point $(0, 0)$ marqué d'une étoile rouge.

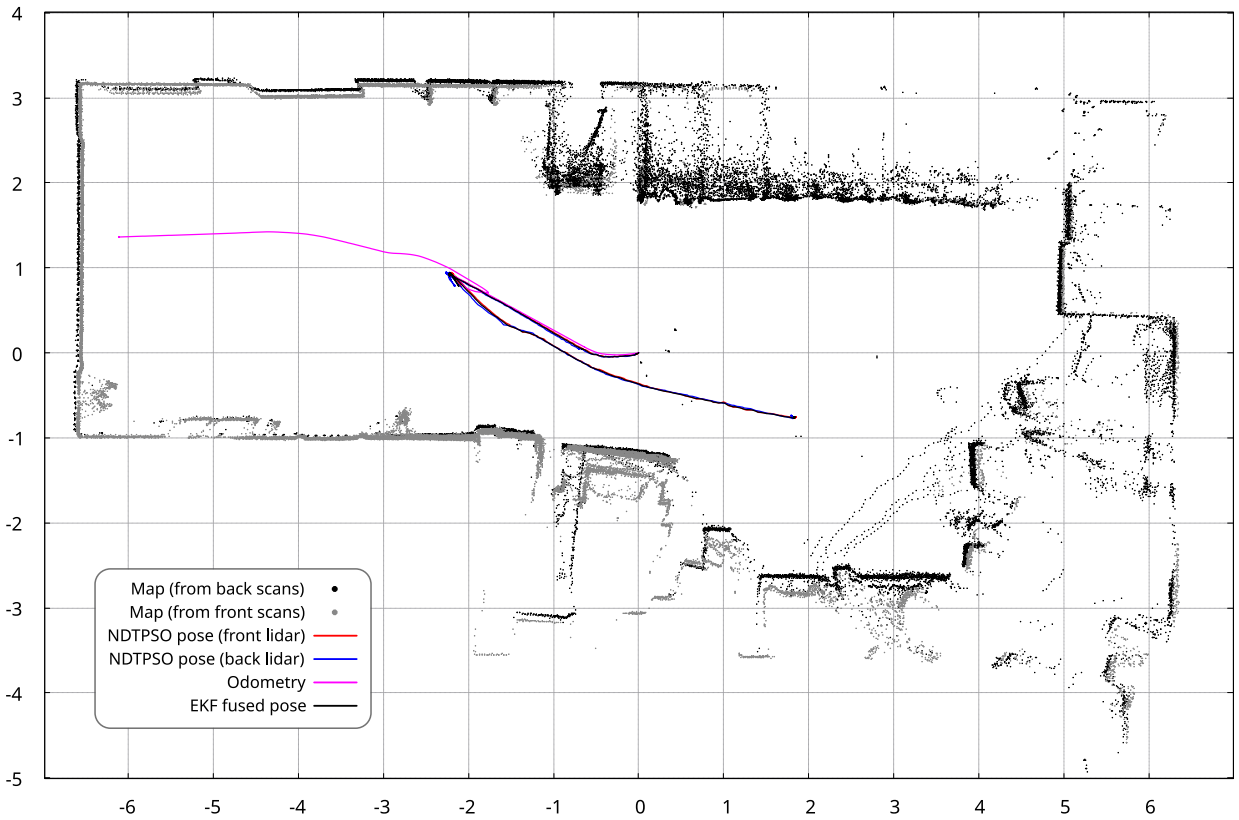


Figure 3.21 – Les cartes générées par l'NDT-PSO à partir des deux LiDARs sur la deuxième séquence, avec les poses associées et l'odométrie (distances en *mètres*).

En l'absence d'une mesure de référence (une vérité terrain) pour la comparaison, nous présentons le tableau 3.4 qui quantifie l'erreur calculée entre les entrées (l'odométrie des roues et les poses calculées à partir des deux LiDARs) et la sortie (la pose fusionnée). Ces valeurs donnent une quantification quant à la contribution de chaque source de données à la pose fusionnée par le filtre de KALMAN étendu.

Table 3.4 – L'erreur moyenne absolue (MAE) et l'erreur carrée moyenne (MSE), sur la première séquence.

	MAE (m m rad)	**MSE** (m^2 m^2 rad^2)
Odométrie	(0.125 0.157 0.134)	(0.0267 0.0640 0.0382)
NDT-PSO sur LiDAR avant	(0.016 0.016 0.010)	(0.0004 0.0007 0.0004)
NDT-PSO sur LiDAR arrière	(0.013 0.010 0.009)	(0.0003 0.0002 0.0004)

En plus de la validation de notre premier prototype, les tests que nous avons menés nous ont permis aussi de proposer des modifications sur la structure mécanique du robot afin d'améliorer le contact roue-sol et minimiser l'éffet du patinage des roues.

3.7 Conclusion

Dans ce chapitre, nous avons conçu, mis en œuvre et validé une plateforme expérimentale modulaire désignée par *SmartTrolley*. Cette plateforme est basée sur un châssis d'un chariot de fortes charges équipé de deux prototypes de la roue ez-Wheel SWD 150 et de deux LiDARs de sécurité.

Ce travail nous a permis dans un premier temps de disposer d'une première version du prototype pour évaluer le contrôleur de mouvements, l'odométrie des roues et les fonctionnalités de la sûreté des LiDARs. Et dans un second temps, cela nous a permis d'exploiter les données issues des deux LiDARs pour proposer une approche de localisation multi-LiDARs à base de la NDT-PSO *(Normal Distribution Transform with Particle Swarm Optimization)* et du filtre de KALMAN étendu *(EKF)*.

Nous voulons souligner que l'expérience menée pour valider la plateforme expérimentale a été réalisée dans un contexte particulier en raison de la crise sanitaire de la Covid-19. Initialement, les séquences utilisées dans ce chapitre étaient destinées à une validation préliminaire, tandis que d'autres séquences accompagnées d'une réalité terrain étaient prévues. Cependant, en raison des difficultés rencontrées pendant cette période, nous n'avons pas pu mener ces expérience. Par conséquent, nous avons exploité ces données, même en l'absence d'une réalité terrain, afin de progresser dans nos travaux.

En raison de l'absence d'une réalité terrain, il n'a pas été possible de réaliser une analyse quantitative des résultats obtenus, particulièrement en ce qui concerne la précision des trajectoires et des cartes estimées. Cependant, nous pouvons évaluer la précision des trajectoires qualitativement en examinant les cartes estimées. En effet, le fait d'avoir un recoupement cohérent des nuages de points (correspondants aux mures, obstacles présents dans l'environnement, coins, *etc.*) provenant des différents scans du même LiDAR (comme

illustré dans les figures 3.19 et 3.21) peut nous permettre de conclure que les estimations des trajectoires sont aussi précises que la qualité des cartes constituées de nuages de points, car chaque pose est associée à un scan.

La modélisation de l'environnement en carte NDT et la spécificité de notre implémentation des cellules en tampon circulaire permettent à cette méthode de fonctionner dans des environnements statiques mais aussi dans des environnements faiblement dynamiques. En revanche, pour les applications ciblées par notre travail, nous nous attendons à des environnements dynamiques à fortement dynamiques (usines, entrepôts, *etc.*) ce qui rend l'utilisation d'une telle approche moins efficace. Pour remédier à cela, nous présentons dans le prochain chapitre une autre approche au problème de la localisation qui serait plus adaptée aux environnemnets dynamiques à fortes contraintes tels qu'identifiés par l'entreprise ez-Wheel.

Les résultats de ce travail ont été valorisés par une publication dans une conférence internationale avec acte et comité de lecture.

- [94] Abdelhak Bougouffa, Emmanuel Seignez, Samir Bouaziz, and Florian Gardes. *« SmartTrolley : An Experimental Mobile Platform for Indoor Localization in Warehouses. »* In 2020 3rd International Conference on Robotics, Control and Automation Engineering (RCAE), 108–15, 2020. DOI : 10.1109/RCAE51546.2020.9294484.

CHAPITRE 4

Localisation indoor par odométrie visuelle à caméra verticale

4.1 Introduction

Plusieurs capteurs peuvent être utilisés pour estimer la pose du robot en environnement indoor. La méthode la plus simple consiste à utiliser les codeurs de roues pour estimer la pose incrémentalement, ce que nous désignons par l'*odométrie des roues*. Cependant, en raison du couplage mécanique, du glissement et du dérapage des roues et de l'absence de correction externe, la trajectoire estimée uniquement à partir de l'odométrie des roues accumule les erreurs rapidement [11], ce qui la rend peu fiable pour une utilisation sur des longues distances.

Alternativement à l'odométrie des roues, nous pouvons utiliser des caméras pour calculer progressivement les changements de position et d'orientation d'un robot en mouvement ; cette technique est connue sous le nom d'*odométrie visuelle (VO - Visual Odometry)* [113]. Le terme *odométrie visuelle* a été introduit par Nister et al. [59], elle consiste à estimer incrémentalement les mouvements relatifs d'une caméra à partir d'un flux d'images perçues.

Notre travail porte sur la localisation en environnements industriels dynamiques. Dans ce type de situations, la présence d'objets mobiles autour du robot peut biaiser son estimation de sa pose. Afin de remédier à cela, nous proposons dans ce chapitre une stratégie de localisation qui sépare l'espace de navigation (utilisé pour la détection d'obstacles et la sûreté) de l'espace de localisation (utilisé pour l'observation des amers). En d'autres termes, pour éliminer l'effet des objets mobiles sur l'estimation, nous n'avons qu'à éviter de les observer. Pour permettre cela, nous proposons d'utiliser le LiDAR présent sur le robot pour la détection d'obstacles et l'implémentation des fonctionalités de sûreté. Tandis que pour assurer la localisation, nous utiliserons une caméra dite verticale, *c.-à-d.*, orientée vers le haut, afin d'observer le plafond et d'estimer les mouvements du robot par rapport à ce dernier.

Dans ce chapitre, nous introduisons les notions de base de la vision par ordinateur, l'odométrie visuelle, et nous menons un état de l'art sur les approches de localisation par caméra verticale (ou *vision au plafond, Ceiling Vision)*. Ensuite, nous proposons et évaluons

la *Ceiling-DSO* [114], un système d'odométrie visuelle basé sur l'approche *Direct Sparse Odometry (DSO)* avec une caméra de vision au plafond. Nous avons mené nos expériences dans un environnement réel, en utilisant la plateforme expérimentale d'ez-Wheel, nommée *SWD® Starter Kit*. Après avoir testé différents paramètres de la DSO, nous avons observé et analysé leurs effets sur la qualité des trajectoires estimées et sur les performances en temps de calcul. La vérité de terrain, utilisée comme référence de comparaison, a été estimée à partir des données du LiDAR à l'aide de la méthode LaMa SLAM [115, 116]. Ainsi, nous fournissons à la fin de ce chapitre une analyse qualitative et quantitative des résultats obtenus.

4.2 Notions de base en vision par ordinateur

Pour développer une méthode de localisation basée sur la vision, il est essentiel de maîtriser un ensemble d'outils techniques et mathématiques permettant de comprendre et de modéliser les images, ainsi que les opérations qui y sont associées. Dans cette section, nous introduisons les concepts fondamentaux de la vision par ordinateur que nous utiliserons dans le reste de ce manuscrit.

La vision par ordinateur est le domaine de l'informatique-électronique qui s'intéresse à l'extraction automatisée d'informations à partir d'images. Les informations dans ce contexte peuvent avoir différentes significations, allant de l'extraction de modèles 3D et l'estimation de la position de la caméra, à la détection et la reconnaissance d'objets dans une scène, en passant par la recherche de contenu dans les images, *etc.* [117].

Pour nous, les humains, la vision est un acquis, nous pouvons sans difficulté reconnaitre les objets, les personnes et la structure qui nous entoure. Par contre, nous ne sommes pas en mesure de décrire comment nous le faisons. Les chercheurs en *psychologie de perception* ont passé des décennies à essayer de comprendre le fonctionnement du système visuel et, même s'ils peuvent concevoir des illusions d'optique pour démêler certains de ses principes, une solution complète à ce casse-tête reste insaisissable [118].

Par conséquent, le problème de la vision par ordinateur est considéré comme un *problème inverse*. Ainsi, nous cherchons à récupérer certaines inconnues (la pose de la caméra, la description d'une scène, la segmentation sémantique de l'image en objets, l'estimation de la pose des objets, la détection de la présence d'objets/formes d'intérêt, *etc.*), étant donné des informations insuffisantes pour spécifier pleinement la solution, *c.-à-d.*, une ou plusieurs images.

Nous utilisons dans la vision par ordinateur certains *modèles directs* qui sont développés à l'origine en physique (radiométrie, optique et conception de capteurs) et en infographie *(computer graphics)*. Ces deux champs modélisent la façon dont les objets se déplacent et se sont animés, comment la lumière se propage, se reflète sur leurs surfaces, se disperse dans l'atmosphère, se réfracte à travers des lentilles de la caméra, et finalement se projette sur un plan d'image [118]. La figure 4.1 illustre le positionnement de la vision par ordinateur par rapport à l'infographie.

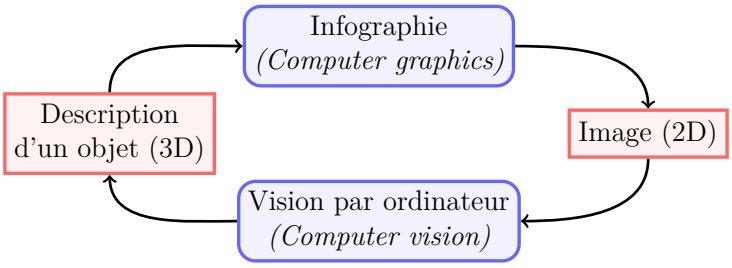


Figure 4.1 – Positionnement de la vision par ordinateur comme problème inverse. En contraste à l'infographie qui reconstitue la projection sur un plan d'image à partir d'une description du monde réel (objets 3D). La vision par ordinateur parcourt le chemin inverse, en extrayant des informations sur le monde réel tridimensionnel en partant d'images 2D.

4.2.1 Modélisation et calibrage de la caméra

En vision par ordinateur, nous définissons un ensemble de systèmes de coordonnées qui nous permettront de raisonner, dans un cadre mathématique bien défini, sur les déplacements de la caméra et le processus de projection d'un point 3D du monde réel vers le plan d'image de la caméra. Ces systèmes de coordonnées sont :

1. Le système de coordonnées de l'*objet* (nommé parfois *repère monde* ou *repère global*) $S_o : [X, Y, Z]^\mathsf{T}$, représente le repère du monde réel où sont situés les objets 3D.

2. Le système de coordonnées de la *caméra* $S_c : [{}^cX, {}^cY, {}^cZ]^\mathsf{T}$, est le repère dont l'origine $\mathcal{O}$ est situé au *centre de la projection*.

3. Le système de coordonnées du *plan image* $S_i : [{}^ix, {}^iy]^\mathsf{T}$, est le plan parallèle au plan du capteur, son centre $\mathcal{H}$, appelé le *point principal*, se situe au point d'intersection de ce plan avec la prolongation de l'axe centrale de la caméra $(\mathcal{O}^cZ)$.

4. Le système de coordonnées du *capteur* $S_s : [{}^su, {}^sv]^\mathsf{T}$, est le repère physique lié au capteur optique de la caméra, son origine est le centre du pixel $(0, 0)$, situé généralement au coin haut-gauche du capteur.

Nous voulons trouver la formule qui nous permet de projeter un point ${}^o\mathcal{P}$ du repère d'objet S_o au point ${}^s\mathcal{P}$ au repère du capteur (pixels) S_s, en passant par les repères S_c et S_i. La figure 4.2 montre l'arrangement des différents systèmes de coordonnées.

Le but donc est de déterminer la relation, que pour l'instant, nous supposons linéaire, entre le point ${}^s\mathcal{P} = [{}^su, {}^sv]^\mathsf{T}$ et le point ${}^o\mathcal{P} = [X, Y, Z]^\mathsf{T}$, tel que :

$$\begin{bmatrix} {}^su \\ {}^sv \\ 1 \end{bmatrix} = {}^s\mathbf{H}_i \, {}^i\mathbf{P}_c \, {}^c\mathbf{T}_o \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.1}$$

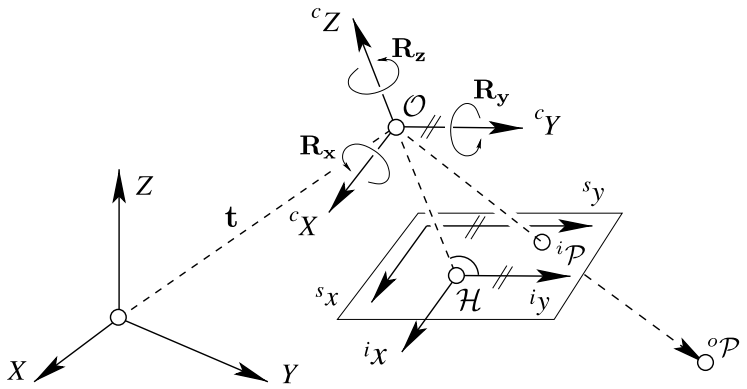


Figure 4.2 – La projection perspective du point $^o\mathcal{P}$ au point $^i\mathcal{P}$ sur le plan de l'image. Le vecteur $\mathbf{t}$ et la matrice $\mathbf{R} = (\mathbf{R_x}, \mathbf{R_y}, \mathbf{R_z})$ représentent respectivement la translation et la rotation entre le repère d'objet $[X, Y, Z]^\mathsf{T}$ et le repère caméra $[^cX, ^cY, ^cZ]^\mathsf{T}$. Le point $\mathcal{H}$ représente le *point principal* de la caméra, et l'axe principal est la ligne qui passe par le point principal $\mathcal{H}$ et le centre de projection $\mathcal{O}$ (adaptée de [119]).

4.2.1.1 Paramètres extrinsèques

Les paramètres extrinsèques sont tous les paramètres de la chaîne de transformations (illustrée dans la figure 4.3) qui se produisent à l'extérieur de la caméra. Cela correspond aux paramètres de la *transformation de corps rigide* $^c\mathbf{T}_o \in \mathrm{SE}(3)$ de 6 degrés de liberté $(X, Y, Z, \theta, \varphi, \psi)$. Cette transformation représente la pose de la caméra dans le repère d'objet, *c.-à-d.*, la position et l'orientation du repère de la *caméra* S_c exprimées dans le repère global S_o.

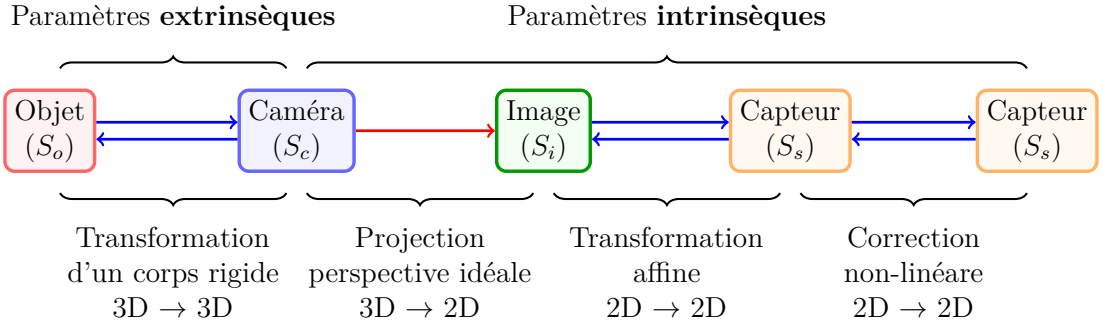


Figure 4.3 – La chaîne des transformations du repère *objet* au repère *capteur*. La *transformation du corps rigide* contient les paramètres dites *extrinsèques*, qui englobent 6 paramètres, 3 pour la position et 3 pour l'orientation. La *projection perspective idéale* est irreversible, en passant d'un point en 3D à sa projection en 2D, nous perdons l'information de la profondeur. Les transformations à droite forment l'ensemble des paramètres *intrinsèques*. La dernière transformation est réalisée dans le plan du capteur et consiste à effectuer des corrections non linéaires (les distorsions, les aberrations chromatiques, *etc.*).

Ainsi, pour un point $^o\mathcal{P}$ représenté dans le repère S_o aux coordonnées $^o\mathbf{X}_\mathcal{P}$, les coordonnées $^c\mathbf{X}_\mathcal{P}$ du point $^c\mathcal{P}$ dans le repère S_c sont données par :

$$ {}^{c}\mathbf{X}_{\mathcal{P}} = \mathbf{R}({}^{o}\mathbf{X}_{\mathcal{P}} - {}^{o}\mathbf{X}_{\mathcal{O}}) \tag{4.2} $$

Les composantes des axes X, Y et Z de la matrice de rotation $\mathbf{R} \in \mathrm{SO}(3)$, notées $\mathbf{R_x}$, $\mathbf{R_y}$, et $\mathbf{R_z}$, respectivement, dépendent des angles d'Euler (θ, φ, ψ), et sont données par :

$$ \mathbf{R} = \mathbf{R_z}(\psi)\mathbf{R_y}(\varphi)\mathbf{R_x}(\theta) \quad \text{avec} \quad \begin{cases} \mathbf{R_x}(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \\ \mathbf{R_y}(\varphi) &= \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \\ \mathbf{R_z}(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{cases} \tag{4.3} $$

Notez que l'ordre des opérations dans l'équation (4.3) est important. La matrice de rotation $\mathbf{R}$ est une matrice orthogonale de déterminant 1, autrement dit toute matrice de rotation doit satisfaire les contraintes suivantes :

$$ \mathbf{R} \in \mathrm{SO}(3) \Longleftrightarrow \begin{cases} \mathbf{I} &= \mathbf{R}\mathbf{R}^{\mathsf{T}} = \mathbf{R}^{\mathsf{T}}\mathbf{R} \\ \det(\mathbf{R}) &= 1 \end{cases} \tag{4.4} $$

4.2.1.2 Coordonnées homogènes

L'équation (4.2) est la représentation en espace euclidien de la transformation du corps rigide ${}^{c}\mathbf{T}_{o}$, dans cette représentation, les translations sont enchainées par addition et les rotations sont enchainées par multiplication. En vision par ordinateur, nous préférons l'utilisation des coordonnées homogènes qui sont plus adaptées pour la géométrie projective, car ils facilitent l'enchainement des transformations.

Les coordonnées homogènes reposent sur l'idée de représenter un vecteur à N dimensions par un vecteur à $N+1$ dimensions. Cette simple idée donne des propriétés intéressantes pour cette représentation des points en géométrie projective, notamment la possibilité de représenter des points à l'infini par des coordonnées finies et de simplifier l'enchainement des transformations affines.

Un point tridimensionnel $\mathbf{X} = [X\ Y\ Z]^{\mathsf{T}}$ peut-être représenté en coordonnées homogènes par $\boldsymbol{X} = [X\ Y\ Z\ 1]^{\mathsf{T}}$, la conversion inverse dépend du quatrième paramètre, ainsi, pour un point en coordonnées homogènes $\boldsymbol{X} = [X\ Y\ Z\ W]^{\mathsf{T}}$, le point 3D associé serait $\mathbf{X} = [\frac{X}{W}\ \frac{Y}{W}\ \frac{Z}{W}]^{\mathsf{T}}$; l'équation (4.2) devient alors :

$$\begin{aligned}\begin{bmatrix} {}^{c}\mathbf{X}_{\mathcal{P}} \\ 1 \end{bmatrix} &= \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I_3} & -{}^{o}\mathbf{X}_{\mathcal{O}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} {}^{o}\mathbf{X}_{\mathcal{P}} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R} & -\mathbf{R}{}^{o}\mathbf{X}_{\mathcal{O}} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} {}^{o}\mathbf{X}_{\mathcal{P}} \\ 1 \end{bmatrix}\end{aligned} \tag{4.5}$$

La matrice ${}^{c}\mathbf{T}_{o}$ des paramètres extrinsèques, qui encode la pose de la caméra dans le repère objet S_o, devient alors :

$${}^{c}\mathbf{T}_{o} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{\mathsf{T}} & 1 \end{bmatrix} \quad \text{avec} \quad \mathbf{t} = -\mathbf{R}{}^{o}\mathbf{X}_{\mathcal{O}} \tag{4.6}$$

4.2.1.3 Paramètres intrinsèques

Les paramètres intrinsèques sont tous les paramètres qui modélisent les phénomènes qui se produisent à l'intérieur de la caméra. Ils définissent le processus de la transformation d'un point 3D vers sa projection sur le capteur de la caméra, cette transformation se réalise en trois étapes :

1. La projection perspective idéale, du repère caméra (S_c) vers le plan de l'image (S_i).
2. La transformation des points du plan de l'image (S_i) vers le plan du capteur (S_s) (plan des pixels).
3. La correction des erreurs de projections et des distorsions causées par la réfraction de la lumière sur l'objectif de la caméra dans le repère du capteur (S_s).

Le processus de projection des points 3D sur un plan 2D de l'image peut être décrit par un modèle géométrique. Il existe une multitude de modèles pour décrire ceci, mais le modèle sténopé (*pinhole camera model*) [55] est sans doute le plus simple et le plus communément utilisé, en particulier pour les caméras à faibles/moyens *champs de vision (FoV - Field of View)* [118].

Dans la chaîne de transformations illustrée précédemment dans la figure 4.3, la projection est supposée idéale selon *le modèle sténopé de la caméra*, cela veut dire [55] :

- Tous les rayons entre les points du repère objet et leurs projections sur le plan d'image sont des lignes droites qui passeront par le point central de la projection $\mathcal{O}$ *(le sténopé)* ;
- Tous les rayons en provenance d'un point du repère objet intersectent le plan de l'image en un seul point.
- La distance entre le centre de projection $\mathcal{O}$ et le plan d'image est la constante de la caméra (f), cette valeur peut être fournie par le constructeur ou calculé lors du calibrage de la caméra.

Bien que nous supposions une projection parfaite, les effets des déformations et des distorsions causées par la projection ne sont pour autant pas omis, ils seront compensés dans les deux étapes qui suivant la projection.

De la figure 4.4, nous pouvons déduire géométriquement la relation entre le point ${}^{c}\mathcal{P} = [{}^{c}X_{\mathcal{P}}, {}^{c}Y_{\mathcal{P}}, {}^{c}Z_{\mathcal{P}}]$ et sa projection sur le plan d'image ${}^{i}\mathcal{P} = [{}^{i}x_{\mathcal{P}}, {}^{i}y_{\mathcal{P}}]$, donnant :

$$ {}^{i}x_{\mathcal{P}} = f\frac{{}^{c}X_{\mathcal{P}}}{{}^{c}Z_{\mathcal{P}}} \qquad \text{et} \qquad {}^{i}y_{\mathcal{P}} = f\frac{{}^{c}Y_{\mathcal{P}}}{{}^{c}Z_{\mathcal{P}}} \tag{4.7} $$

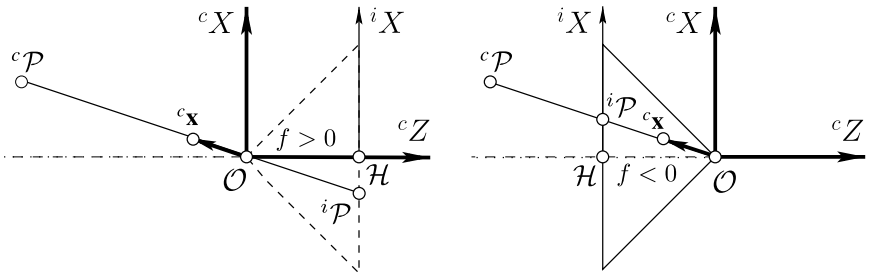


Figure 4.4 – La représentation du plan d'image dans le modèle sténopé. À gauche, la projection en modèle physique, avec la distance focale $f > 0$, le plan image tombe derrière le centre de la projection, mais l'image est inversée. À droite, le modèle conventionnel avec $f < 0$, le plan image est retourné de 180°, le plan image est donc devant le centre de la caméra (adaptée de [119]).

Nous définissons ainsi la projection perspective idéale.

$$ \lambda \begin{bmatrix} {}^{i}x_{\mathcal{P}} \\ {}^{i}y_{\mathcal{P}} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{c}X_{\mathcal{P}} \\ {}^{c}Y_{\mathcal{P}} \\ {}^{c}Z_{\mathcal{P}} \\ 1 \end{bmatrix} \tag{4.8} $$

Avec $\lambda \in \mathbb{R}^{+}$ le facteur d'échelle, la *matrice de la projection de la caméra*, notée ${}^{i}\mathbf{P}_{c}$, contient la constante de la caméra f, qui représente l'équivalent physique de la distance focale.

$$ {}^{i}\mathbf{P}_{c} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9} $$

La dernière transformation linéaire ${}^{s}\mathbf{H}_{i}$ dans l'équation (4.1) consiste à aligner le plan d'image et le plan du capteur, cet alignement se traduit à une translation caractérisée par le couple $(c_x\ c_y)$, une constante m qui décrit la différence de l'échelle sur les axes x et y, ainsi qu'un potentiel *cisaillement (shear)* caractérisé par le scalaire s. Ce dernier est généralement égal à zéro dans les caméras numériques modernes.

$$ {}^{s}\mathbf{H}_{i} = \begin{bmatrix} 1 & s & c_x \\ 0 & 1+m & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10} $$

La composition des matrices ${}^{i}\mathbf{P}_{c}$ et ${}^{s}\mathbf{H}_{i}$ donnent *la matrice de la caméra projective* $\mathbf{K}$ qui englobe tous les paramètres intrinsèques *linéaires* de la caméra.

$$ \mathbf{K} \triangleq {}^{s}\mathbf{H}_{i}{}^{i}\mathbf{P}_{c} = \begin{bmatrix} f & sf & c_x \\ 0 & (m+1)f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11} $$

À partir des équations (4.1), (4.6) et (4.11), nous pouvons définir la fonction de projection $\Pi_{\mathbf{K}} : \mathbb{R}^3 \to \Omega$ qui permet de projeter un point tridimensionnel du repère d'objet ${}^o\mathcal{P} = [X, Y, Z]^\mathsf{T}$ à sa projection ${}^s\mathcal{P} = [{}^su, {}^sv]^\mathsf{T}$ sur le repère capteur, tel que :

$$\Pi_{\mathbf{K}}({}^o\mathcal{P}) : \begin{bmatrix} {}^su \\ {}^sv \\ 1 \end{bmatrix} = \begin{bmatrix} f & sf & c_x & 0 \\ 0 & f(m+1) & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.12}$$

Nous définissons aussi le modèle de projection inverse $\Pi_{\mathbf{K}}^{-1} : \mathbb{R} \times \Omega \to \mathbb{R}^3$, qui rétro-projette un point 2D (${}^s\mathcal{P}$) du plan de capteur au repère de l'objet en connaissant sa profondeur $d_{\mathcal{P}}$:

$$\Pi_{\mathbf{K}}^{-1}({}^i\mathcal{P}, d_{\mathcal{P}}) : \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = d_{\mathcal{P}} \mathbf{K}^{-1} {}^i\mathcal{P} \tag{4.13}$$

L'équation (4.12) est appelée *la transformation linéaire directe* qui représente le modèle de la caméra affine. Néanmoins, en réalité, les caméras ne sont pas linéaires, l'optique de la caméra introduit aussi des distorsions et des aberrations chromatiques. Ces dernières peuvent être modélisées et corrigées après l'acquisition de l'image. Un modèle de correction de distorsion $[\hat{u}, \hat{v}]^\mathsf{T} = \mathcal{L}([u, v]^\mathsf{T})$ est une fonction $\mathcal{L} : S_s \to S_s$ qui transforme chaque point $[u, v]^\mathsf{T}$ observé sur le plan du capteur en un point corrigé $[\hat{u}, \hat{v}]^\mathsf{T}$. La plupart des distorsions radiales et tangentielles peuvent être corrigées par le *modèle radial-tangentiel* [118] :

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = (1 + \kappa_1 r^2 + \kappa_2 r^4) \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} 2\rho_1 uv + \rho_2(r^2 + 2u^2) \\ 2\rho_2 vu + \rho_1(r^2 + 2v^2) \end{bmatrix} \tag{4.14}$$

$$\text{Avec} \quad r = \sqrt{(u - u_{\mathcal{H}})^2 + (v - v_{\mathcal{H}})^2} \tag{4.15}$$

Le paramètre r représente la distance entre le point mesuré et le point principal de la caméra $\mathcal{H}$, et les facteurs κ_j et ρ_j sont les paramètres des distorsions radiale et tangentielle, respectivement.

Le modèle (4.14) est adapté à des distorsions modérées, pour les caméras avec des objectifs de grands angles, le modèle *équidistant* ou *œil de poisson (Fisheye)* serait plus adéquat [55] :

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \frac{\theta}{r}(\kappa_1 + \kappa_2\theta^2 + \kappa_3\theta^4 + \kappa_4\theta^6) \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.16}$$

$$\text{Avec} \quad \theta = \tan^{-1}(r) \tag{4.17}$$

La figure 4.5 illustre quelques formes communes de distorsions. La sous-figure 4.5a est la grille de référence sans distorsion. Les sous-figures 4.5b-f sont basées sur le modèle radial-tangentiel (équation 4.14) où la 4.5b est une distorsion radiale positive (ou *coussinet*), *c.-à-d.*, $\kappa_j > 0$; $\rho_j = 0$. La 4.5c est une distorsion radiale négative (ou *barillet*) avec $\kappa_j < 0$; $\rho_j = 0$. La 4.5d est une distorsion radiale complexe (ou *moustache*), représentant un mix de distorsions négative et positive, *c.-à-d.*, $\text{sign}(\kappa_1) \neq \text{sign}(\kappa_2)$. La 4.5e une distorsion tangentielle avec

$\kappa_j = 0;\ \rho_j \neq 0$. Et la 4.5f une distorsion radiale-tangentielle, *c.-à-d.*, $\kappa_j \neq 0;\ \rho_j \neq 0$. La sous-figure 4.5g est une distorsion en œil de poisson, basée sur le modèle donnée par l'équation (4.16).

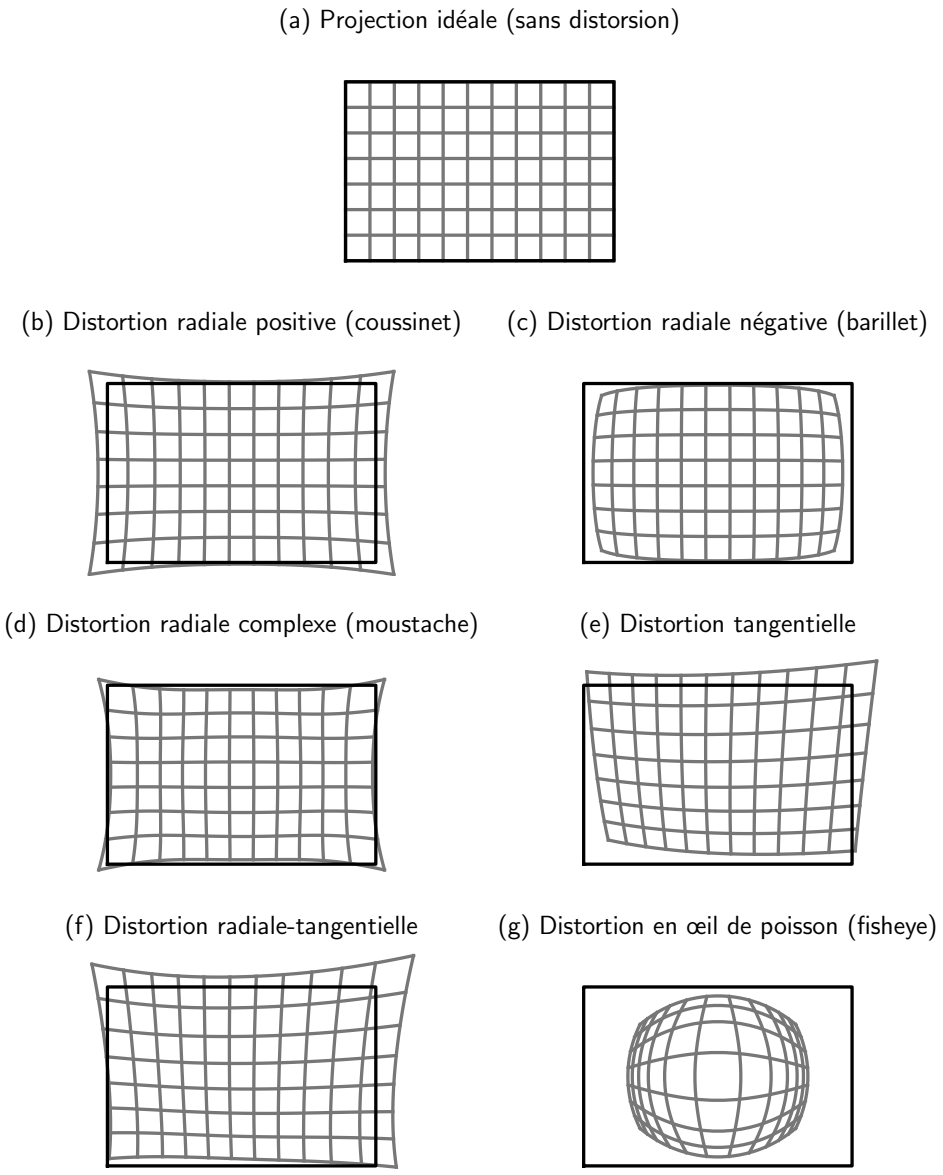


Figure 4.5 – Les types de distorsions les plus communs. Le cadre noir dans chaque figure spécifie les contours de la grille originale.

En vision par ordinateur, nous appelons *calibrage (ou étalonnage) d'une caméra* le processus de l'estimation des paramètres intrinsèques à partir d'un ensemble d'observations réalisées par cette caméra. La matrice résultante $\mathbf{K}$ est parfois appelée *la matrice de calibrage* dans la littérature scientifique. La technique la plus commune est d'utiliser une grille plane de motifs facilement reconnaissables, généralement, sous forme d'échiquier. La caméra prend plusieurs photos de l'échiquier de différents angles de vue, puis, l'algorithme de calibrage

exploite le fait que, tous les points sur l'échiquier appartenant au même plan. Cela impose une contrainte de coplanarité sur l'ensemble de coins détectés sur l'échiquier. Ainsi, chaque image est utilisée pour calculer une *homographie* distincte permettent de transformer les points du plan de l'échiquier au plan de l'image, puis, la matrice de calibrage peut être estimée en résolvant le système linéaire résultant des homographies et des contraintes connues [118].

Le terme « calibrage d'une caméra » signifie communément *le calibrage géométrique*, *c.-à-d.*, l'estimation de la matrice des paramètres intrinsèques $\mathbf{K}$ de la caméra. Néanmoins, il existe un autre type de calibrage appelé *calibrage photométrique*. Ce dernier peut s'avérer utile pour les méthodes directes. Le calibrage photométrique s'intéresse à estimer les paramètres qui agissent sur la luminosité mesurée de chaque pixel, tel que la fonction non-linéaire de transfert *(correction gamma)*, les atténuations de l'objectif *(vignettage)*, les artéfacts du dématriçage [1], ou même les distorsions géométriques causées par l'acquisition séquentielle des pixels dans un capteur à obturateur roulant *(Rolling Shutter)* [120].

4.2.2 Groupes et algèbres de Lie

Un groupe de Lie est un objet mathématique abstrait qui remonte au XIXe siècle, lorsque le mathématicien Marius Sophus Lie (1842-1899) a posé les bases de la théorie des *groupes de transformations continues*. Les groupes de Lie et leurs algèbres associés représentent un vaste domaine des mathématiques avec de nombreuses applications en mathématique, en ingénierie et en physique.

En robotique et en vision par ordinateur, nous nous intéressons à un sous-ensemble des groupes de Lie qui nous permettront de représenter des rotations, des transformations rigides et des similarités. Les groupes de Lie permettent une représentation compacte et minimale des transformations, ils sont particulièrement intéressants pour l'optimisation de la géométrie 3D [121], permettant d'avoir un cadre cohérent et robuste pour la robotique et la vision par ordinateur.

Le groupe de Lie est un concept mathématique qui combine les notions de *groupe* et de *variété lisse (smooth manifold)* dans une structure unifiée. Ainsi, un groupe de Lie est *une variété lisse* dont les éléments satisfont *les axiomes du groupe*. Une variété lisse $\mathcal{M}$ présente la particularité d'avoir un espace tangent unique (un espace vectoriel) à chaque point. L'espace tangent à l'identité ϵ de la variété $\mathcal{M}$ est appelé l'algèbre de Lie $\mathcal{T}\epsilon\mathcal{M}$ de $\mathcal{M}$. La théorie des groupes de Lie établit des relations entre la variété lisse et son algèbre de Lie associé.

La figure 4.6 illustre la relation entre un *groupe de Lie* et son *algèbre de Lie* associé. L'algèbre de Lie $\mathcal{T}\epsilon\mathcal{M}$ (plan rouge) est l'espace tangent en ϵ (l'identité) à la variété $\mathcal{M}$ du groupe de Lie (représentée par une sphère bleue). À travers l'application exponentielle, chaque chemin droit $\mathbf{v}t$ passant par l'origine sur l'algèbre de Lie produit un chemin $\exp(\mathbf{v}t)$ autour de la variété $\mathcal{M}$. Inversement, chaque élément du groupe (qui est courbe et non linéaire) possède un équivalent exact dans l'algèbre de Lie (qui est un espace vectoriel linéaire), et cette relation est valide pour presque toutes les opérations dans le groupe. Bien

1. Les artéfacts du *dématriçage* (apellés aussi *démosaïsage*, *débayerisator* ou *de-bayering artifacts*), sont les artéfacts introduits par l'interpolation des données lors de l'extraction des canaux monochromes rouge, vert et bleu à partir de la *matrice de filtres colorés* (matrice de Bayer par exemple).

que la sphère dans $\mathbb{R}^3$ ne soit pas un groupe de Lie (dans la figure 4.6, la sphère est juste une représentation), celle dans $\mathbb{R}^4$ forme un groupe de Lie qui décrit le groupe de quaternions unitaires [121].

Nous n'entrerons pas dans les détails mathématiques ici, mais pour une lecture plus approfondie sur l'utilisation des groupes et des algèbres de Lie dans le contexte de l'estimation d'état, nous recommandons l'article *« A Micro Lie Theory for State Estimation in Robotics »* de Solà et al. [121].

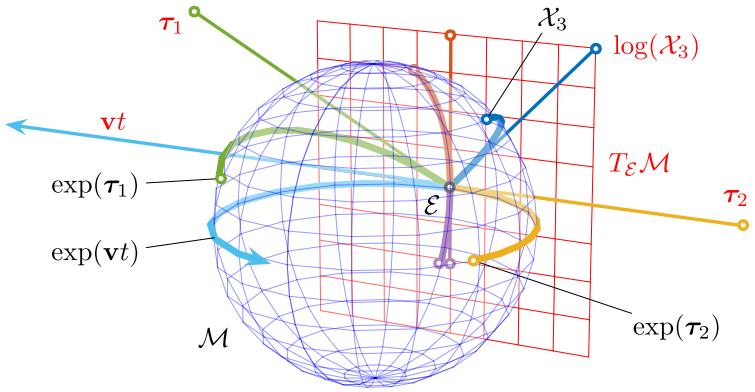


Figure 4.6 – Illustration de la relation entre un *groupe de Lie* et son *algèbre de Lie* (tirée de [121]).

Dans les prochaines sections de ce chapitre, nous exploiterons cette représentation pour modéliser les transformations rigides, les similarités, et pour représenter les paramètres à optimiser dans le contexte de l'odométrie visuelle DSO.

4.3 Odométrie visuelle

Le terme *« odométrie visuelle » (VO - Visual Odometry)* a été introduit pour la première fois par Nister et al. [59]. L'*odométrie visuelle* est une technique utilisée en robotique pour estimer le mouvement d'un robot en se basant sur les informations visuelles capturées par une ou plusieurs caméras embarquées. Elle repose sur le suivi et la reconstruction en 3D des points d'intérêt dans l'environnement pendant que le robot se déplace. En suivant les déplacements de ces points dans les images successives, il est possible d'estimer les déplacements relatifs du robot et reconstituer sa trajectoire.

L'odométrie visuelle trouve des applications dans divers domaines, tels que la navigation autonome, la localisation, la cartographie et la réalité virtuelle, augmentée et mixte.

4.3.1 Formulation du problème

Pour un robot doté d'une ou plusieurs caméras fixées sur ce dernier, nous pouvons formuler le concept de l'odométrie visuelle de la manière suivante :

- $S_c : [{}^cX, {}^cY, {}^cZ]^\mathsf{T}_k$ est le système de coordonnées mobile attaché à la caméra avec $\mathcal{O}_k$ son origine ;
- $S_o : [{}^oX, {}^oY, {}^oZ]^\mathsf{T}$ est le système de coordonnées fixe attaché à l'environnement ;

- Le robot mobile démarre d'une position et d'une orientation initiales connues ;
- Le robot se déplace et réalise des prises d'images I_k pendant ses mouvements ;
- Les images successives I_{k-1} et I_k prises par la caméra (figure 4.7) sont liées par la transformation de corps rigide du point $\mathcal{O}_{k-1}$ au point $\mathcal{O}_k$ notée $\mathbf{T}_{k|k-1} = \begin{bmatrix} \mathbf{R}_{k|k-1} & \mathbf{t}_{k|k-1} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix}$;
- Le but de l'odométrie visuelle est de calculer les transformations relatives $\mathbf{T}_{k|k-1}$ et les accumuler pour estimer la trajectoire du robot dans le repère S_o.

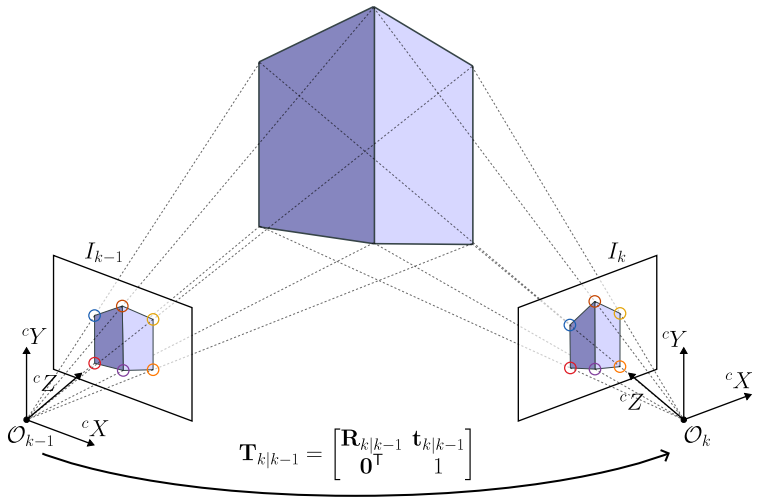


Figure 4.7 – Principe de l'odométrie visuelle. La caméra se déplace du point $\mathcal{O}_{k-1}$ au point $\mathcal{O}_k$. Le but est d'estimer la translation $\mathbf{t}_{k|k-1} \in \mathbb{R}^3$ et la rotation $\mathbf{R}_{k|k-1} \in \mathrm{SO}(3)$ entre les deux poses de la caméra. Ceci est réalisé en détectant et suivant les mêmes caractéristiques sur plusieurs images.

Les images acquises sont traitées pour extraire des caractéristiques *(features)* ou des points d'intérêt représentant des amers dans l'environnement. En vision stéréoscopique, ces caractéristiques peuvent être des points tridimensionnels (3D), tandis qu'en vision monoculaire, ils seront des points (ou des *patches*) en 2D qui représentent les projections en plan d'image des vrais points d'intérêt en 3D.

Après l'extraction des caractéristiques, l'étape suivante consiste à trouver leurs correspondances sur les images successives, *c.-à-d.* associer à chaque caractéristique observée dans une image I_k le point (ou le patch) dans l'image I_{k-1} qui correspond à cette même caractéristique. Ces correspondances sont ensuite utilisées pour estimer le mouvement de la caméra.

Les résultats de l'estimation des mouvements peuvent être améliorés via une étape d'optimisation locale (appelée aussi *ajustement de faisceaux en local* ou *Local Bundle Adjustment*). À noter que dans la VO, cette optimisation est appliquée localement, et d'une manière générale, sur une fenêtre de N dernières images. Cela permet d'améliorer l'estimation de la trajectoire tout en maîtrisant le temps d'exécution.

La structure générale d'un algorithme d'odométrie visuelle est illustrée dans la figure 4.8, mettant en évidence les différentes techniques utilisées pour l'estimation des mouvements.

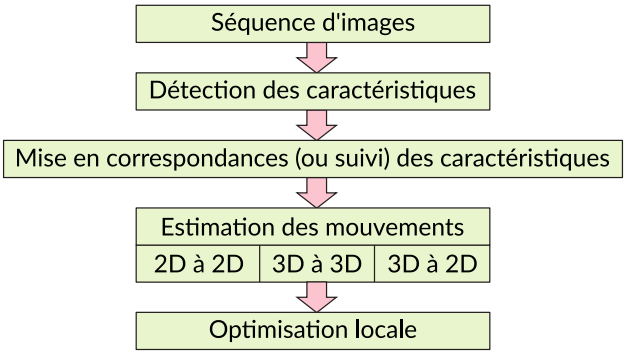


Figure 4.8 – Diagramme en blocs des étapes de l'odométrie visuelle (adaptée de [113]).

4.4 Estimation des mouvements

Les techniques de l'estimation de mouvements pour la VO peuvent être divisées en trois grandes familles selon la dimension des correspondances de caractéristiques entre les images successives I_{k-1} et I_k.

Cette estimation peut être de 3D à 3D lorsque les caractéristiques observées précédemment sont représentées en 3D et la caméra délivre des points en 3D. De 3D à 2D lorsque les points délivrés par la caméra sont en 2D. Et de 2D à 2D quand les points observés par la caméra et les caractéristiques observés précédemment sont tous les deux exprimés en 2D.

4.4.1 Estimation de mouvements 3D à 3D

Nous parlons d'une estimation de mouvements de 3D à 3D quand les caractéristiques observées dans l'image I_{k-1} sont représentées en 3D et la caméra délivre une nouvelle image de profondeur I_k (par caméras stéréo, RGB-D, ToF, *etc.*) avec des caractéristiques qui sont aussi en 3D.

L'estimation de mouvement dans ce cas est réalisée par triangulation des caractéristiques 3D observées dans une séquence d'images. Cette technique peut être vue comme un problème d'alignement de nuages de points 3D. La transformation relative $\mathbf{T}_{k|k-1}$ entre $\mathcal{O}_{k-1}$ et $\mathcal{O}_k$ est estimée en minimisant la distance euclidienne entre les caractéristiques 3D (indexées par j) observées dans l'image (k) actuelle ${}^o\mathcal{P}_{j,(k)}$ et leurs correspondances dans l'image $(k-1)$ notées ${}^o\mathcal{P}_{j,(k-1)}$. L'optimisation d'un tel système peut être modélisée par un problème de moindres carrés, avec une fonction objective de la forme suivante [55] :

$$\mathbf{T}_{k|k-1} = \arg\min_{\mathbf{T}} \sum_j \| {}^o\mathcal{P}_{j,(k)} - \mathbf{T}\ {}^o\mathcal{P}_{j,(k-1)} \|^2 \tag{4.18}$$

4.4.2 Estimation de mouvements 3D à 2D

Cette estimation est réalisée à partir d'une structure en 3D et leurs correspondances 2D dans les images. Les transformations relatives sont estimées à partir des caractéristiques bidimensionnelles extraites de la nouvelle image et les caractéristiques tridimensionnelles (la structure en 3D) observées dans les images précédentes.

Pour chaque point ${}^{i}\mathcal{P}_{j,(k)}$ mesuré dans l'image k, le point 3D ${}^{o}\mathcal{P}_{j,(k-1)}$ correspondant à la caractéristique observée dans l'image $k-1$ est projeté à l'image actuelle en utilisant la fonction de projection $\Pi_{\mathbf{K}}(\mathbf{T}, {}^{o}\mathcal{P}_{j,(k-1)})$. Ainsi, l'estimation de mouvements 3D à 2D peut être modélisée comme un problème d'optimisation de la forme [118] :

$$\mathbf{T}_{k|k-1} = \arg\min_{\mathbf{T}} \sum_{j} \| {}^{i}\mathcal{P}_{j,(k)} - \Pi_{\mathbf{K}}(\mathbf{T}, {}^{o}\mathcal{P}_{j,(k-1)}) \|^2 \tag{4.19}$$

4.4.3 Estimation de mouvements 2D à 2D

Lorsque les points observés par la caméra et les points observés précédemment sont tous les deux exprimés en 2D, l'estimation de mouvements peut être réalisée via l'exploitation de la géométrie épipolaire. La figure 4.9 illustre les éléments de la géométrie épipolaire.

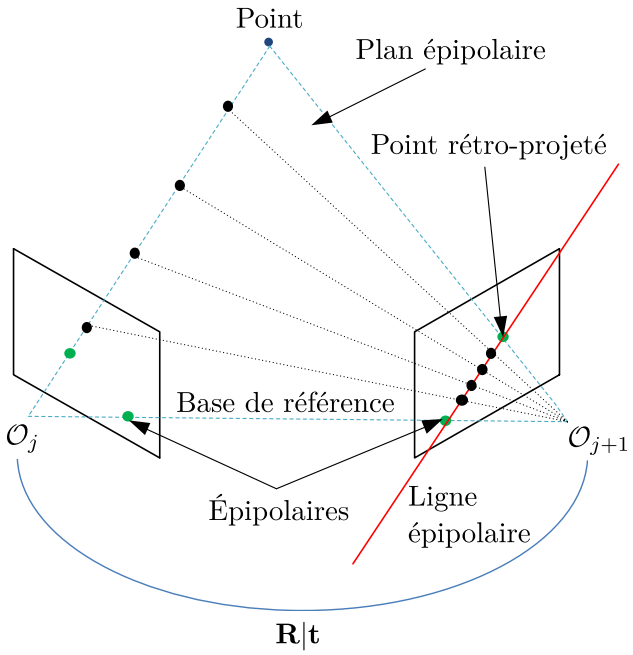


Figure 4.9 – Éléments de la géométrie épipolaire (adaptée de [122]).

À partir de points 2D exprimés en coordonnées homogènes P_k et P_{k-1}, l'estimation de mouvements 2D à 2D exploite la *contrainte épipolaire* :

$$P_{k-1}^{\mathsf{T}} \, \mathbf{E}_k \, P_k = 0 \tag{4.20}$$

Avec $\mathbf{E}_k$ est la matrice essentielle, qui peut être décomposée par la suite de la manière suivante :

$$\mathbf{E}_k \overset{\lambda}{=} \hat{\mathbf{t}}_k \, \mathbf{R}_k \tag{4.21}$$

$$\text{Avec} \quad \hat{\mathbf{t}}_k = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad \text{et} \quad \mathbf{t}_k = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{4.22}$$

Le symbole $\overset{\lambda}{=}$ dans l'équation (4.21) signifie que l'égalité est valide à une échelle λ près, les mesures étant en deux dimensions, l'échelle métrique de la transformation ne peut pas être estimée directement.

La matrice essentielle $\mathbf{E}_k$, une fois estimée, peut être décomposée en deux matrices $\hat{\mathbf{t}}_k$ et $\mathbf{R}_k$, qui représentent respectivement, la translation et la rotation, en utilisant par exemple une *décomposition en valeurs singulières (SVD - Singular Value Decomposition)*.

4.5 Classifications des méthodes de la VO

Sur le plan conceptuel, les approches d'odométrie visuelle (VO) peuvent être divisées en deux principales catégories : les *méthodes indirectes* et les *méthodes directes*. Nous pouvons également les classifier selon la quantité des caractéristiques/données traitées, donnant des méthodes *denses*, *semi-denses* ou *éparses*.

Dans cette section, nous introduisons brièvement ces concepts et nous éclaircissions les différences entre l'odométrie visuelle, le SLAM visuel et la structure à partir du mouvement (SfM).

4.5.1 Méthodes indirectes

Les *méthodes indirectes*, dites aussi, *méthodes basées sur des caractéristiques (Feature-based)*, sont des méthodes qui nécessitent une étape de prétraitement dans laquelle des techniques de détection de caractéristiques sont utilisées pour extraire un ensemble de *caractéristiques* ou de *repères visuels*. Ces *caractéristiques* sont ensuite utilisées pour minimiser *l'erreur géométrique*.

La majorité des méthodes proposées en VO et en SLAM visuel sont des méthodes indirectes. MonoSLAM [66] est un des premiers exemples d'une telle méthode. MonoSLAM utilise l'opérateur Shi-Tomasi pour détecter les caractéristiques dans les images successives. Ensuite, MonoSLAM procède à une correction réalisée grâce à une corrélation croisée appliquée aux caractéristiques précédentes et celles détectées dans l'image actuelle. Les caractéristiques détectées sont en 2D, leurs profondeurs ne sont donc pas connues, mais elle est contrainte par la droite épipolaire (orientée du centre de caméra vers le point). Dans MonoSLAM, cette droite est échantillonnée uniformément par des particules jusqu'à une profondeur limite. Au cours du mouvement de la caméra, un filtre à particules est utilisé pour estimer la profondeur à partir des particules échantillonnées. Puis, la carte est mise à

jour via un filtre de KALMAN, où un modèle de vitesse constante est utilisé pour prédire les mouvements de la caméra et une correction est appliquée sur l'ensemble de caractéristiques 3D en utilisant les points raffinés par le filtre à particules.

D'autres méthodes indirectes ont été proposées dans la littérature scientifique, notamment l'ORB SLAM [68], qui utilise le détecteur et le descripteur ORB incorporé dans une approche à base d'optimisation pour raffiner les estimations. Des méthodes indirectes sont également proposées dans le cadre de fusion de données [123, 124]. La figure 4.10 montre un exemple des caractéristiques extraites sur une image par ORB SLAM.

4.5.2 Méthodes directes

Les *méthodes directes* [62, 63], dites aussi *basées sur l'apparence (Appearance-based)*, sont des méthodes qui ne nécessitent pas l'extraction de caractéristiques, mais elles surveillent les changements dans l'apparence des images acquises. Ces méthodes utilisent directement les valeurs brutes d'intensités des pixels contenus dans les images au lieu d'extraire et de suivre des caractéristiques dans ces images.

Le principal avantage des méthodes directes c'est qu'elles s'appuient sur la minimisation d'une erreur basée sur des mesures réelles (*c.-à-d.* les valeurs d'intensité des pixels). Ces méthodes peuvent par ailleurs être rendues statistiquement robustes en raison de la redondance des informations [125].

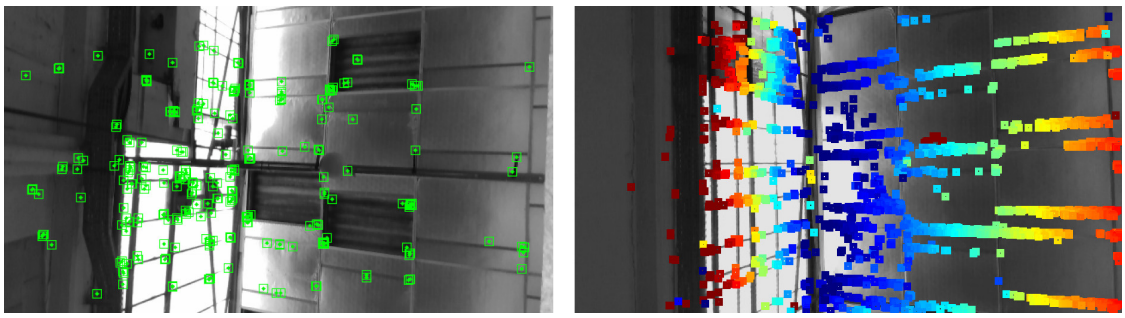


Figure 4.10 – Exemples des points suivis sur une image dans la méthode indirecte ORB-SLAM (à gauche) et la méthode directe DSO (à droite)

La majorité des méthodes directes sont basées sur une forme d'estimation du mouvement et de vitesse de la caméra à partir du *flux optique (OF - Optical flow)*. L'algorithme OF utilise les valeurs d'intensité des pixels voisins pour calculer le déplacement des motifs de luminosité d'une image à l'autre.

Les algorithmes qui estiment le déplacement pour tous les pixels de l'image sont connus sous le nom d'algorithmes *denses* de flux optique tels que l'algorithme d'HORN-SCHUNCK qui calcule le déplacement à chaque pixel en utilisant des contraintes globales. De la même manière, les algorithmes qui calculent le déplacement pour un nombre réduit de pixels dans l'image sont appelés algorithmes *épars* (*fig.* 4.10) de flux optique tels que la méthode de LUCAS-KANADE [126]. Dans les algorithmes épars, les zones d'intérêt sont choisies soigneusement en considérant les pixels dans les régions avec plus de variance, ce qui produit une estimation de déplacement plus fiable et robuste.

4.5.3 Méthodes structurelles

Les méthodes que nous appelons *structurelles* ou *contraintes* [127-129] sont généralement conçues pour les environnements bien structurés construits par l'homme. Ces méthodes sont souvent basées sur des formulations *indirectes* pour l'extraction des caractéristiques. Mais en plus de ça, ces méthodes exploitent les contraintes géométriques présentes sur ce type d'environnements (lignes droites en parallèle, angles droits sur les extrémités des murs, *etc.*). Ces contraintes sont ensuite prises en compte lors de l'optimisation et l'estimation de la pose et de la structure 3D de l'environnement.

La figure 4.11 montre un exemple de la méthode StructSLAM [127] qui exploite la régularité structurelle des environnements construits par l'homme.

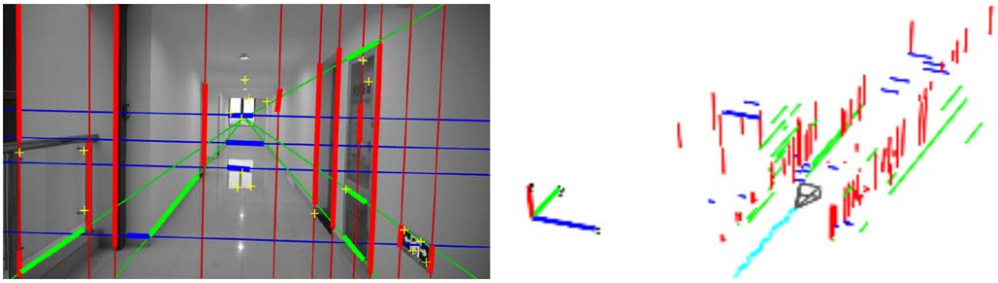


Figure 4.11 – Exemple d'une approche structurelle. À gauche : les lignes définissant la structure ; à droite : la structure 3D estimée (tirée de [127])

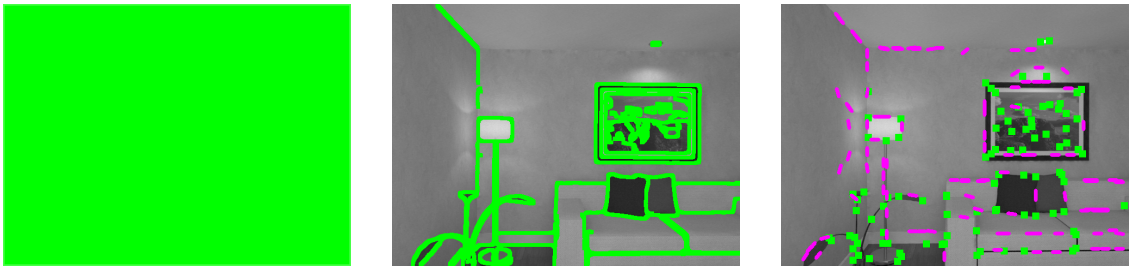
4.5.4 Dense vs. éparse

Nous pouvons également classer les approches VO par la densité des données qu'elles utilisent pour estimer les mouvements et la structure 3D. Ainsi, une VO peut être qualifiée de *dense* [130], *semi-dense* [72, 131], ou *éparse* [66, 68].

La figure 4.12 illustre en vert les pixels utilisés pour l'alignement d'image à image. Les méthodes *denses* utilisent tous les pixels de l'image (*fig.* 4.12a). Les méthodes *semi-dense* utilisent un sous échantillon des pixels, qui par exemple, prend seulement les zones avec des hautes intensités de gradients (*fig.* 4.12b). Les méthodes *éparses* utilisent les pixels soigneusement sélectionnés, comme des zones de hautes intensités avec des critères supplémentaires (*fig.* 4.12c) tels que l'utilisation des zones autour des coins ou la réalisation d'un sous échantillonnage uniforme sur les zones à hautes intensités.

4.5.5 Odométrie visuelle vs. SfM

Dans la littérature, la *structure à partir du mouvement (SfM - Structure from Motion)* est une autre technique similaire à celle de l'odométrie visuelle. Néanmoins, dans la SfM, les images *ne sont pas forcément ordonnées*, *c.-à-d.*, les images peuvent être traitées dans n'importe quel ordre. Ceci découle du but de la SfM qui s'intéresse principalement à la reconstruction en 3D de la scène à partir d'un ensemble d'images de celle-ci.



(a) Dense **(b)** Semi-dense **(c)** Éparse

Figure 4.12 – Exemple d'une image avec en vert, les pixels utilisés pour l'alignement d'image à image. Les méthodes *denses* (4.12a) utilisent tous les pixels de l'image. Les méthodes *semi-dense* (4.12b) utilisent les pixels des zones avec des hautes intensités de gradients. Les méthodes *éparses* (4.12c) utilisent les pixels des zones de hautes intensités, avec d'autres critères (dans cet exemple, l'utilisation des zones autour des coins) (tirée de [131]).

Dans ce cas, la connaissance ou l'estimation conjointe de la pose de la caméra pour chaque image est donc primordiale pour la reconstruction. Néanmoins, le problème de la SfM n'est pas centré autour de l'estimation de la pose, et, vu que les images ne sont pas forcément acquises séquentiellement, l'estimation d'une trajectoire dans ce cas ne décrit pas forcément un mouvement réel.

La VO peut être vue comme un cas spécial de la SfM dans lequel l'estimation de la trajectoire est la problématique centrale, et le flux d'images est acquis d'une manière séquentielle à partir d'une ou plusieurs caméras. Les poses estimées de cette manière décriront ainsi la vraie trajectoire de la caméra.

En plus, l'estimation de la trajectoire dans la VO doit répondre à des contraintes temporelles, car le robot doit connaitre sa pose en temps réel pour pouvoir naviguer. Tandis que dans la SfM, les algorithmes sont très lents, parce qu'ils cherchent à reconstruire la scène observée en 3D, et réalisent pour cela une optimisation globale sur toutes les images.

4.5.6 Odométrie visuelle vs. vSLAM

L'odométrie visuelle diffère du SLAM visuel (vSLAM) dans la stratégie de gestion de la carte. En VO, nous ne nous intéressons qu'à la cohérence d'une carte locale partielle et nous l'utilisons pour estimer le mouvement du robot progressivement.

En revanche, un algorithme vSLAM peut être vu comme une VO couplée avec un algorithme de correction globale de la carte ainsi qu'un mécanisme de fermeture de boucle. Le vSLAM doit ainsi maintenir une carte cohérente globalement [113].

4.6 La vision verticale *(ceiling-vision)*

Pour notre robot industriel mobile, nous nous attendons à des environnements très mouvementés, avec de nombreuses personnes et robots se déplaçant dans le même espace. Nous appelons ce type d'environnements les *environnements dynamiques*. À l'opposé des

environnements statiques, qui sont supposés figés dans le temps et dans l'espace, le cas dynamique relève de nouveaux défis. Dans ce cas, l'estimation de la VO peut s'avérer difficile, car nous devons reconnaître les objets en mouvement dans la scène afin de les exclure lors du calcul de l'égo-mouvement du robot [29, 132], ce qui a tendance à être un problème compliqué et très consommateur en matière de ressources.

Pour surmonter ces problèmes liés aux environnements dynamiques intérieurs, nous proposons d'utiliser une caméra verticale (orientée vers le haut) et d'observer les motifs au plafond, et puis de les utiliser pour le suivi et l'estimation des mouvements du robot. WooYeon et al. sont les premiers à proposer une telle démarche [133]. Ils ont utilisé une caméra monoculaire orientée vers le haut, et ensuite, ils ont exploité le détecteur de coin d'Harris pour extraire les coins des images perçues. Ces coins sont utilisés comme des caractéristiques visuelles (des points de repère) dans un cadre SLAM basé sur le filtre de Kalman étendu (EKF). WooYeon et al. ont également proposé dans leur système une description multivue des coins pour améliorer l'association des données.

Dans la plupart des approches d'odométrie ou de SLAM basées sur la vision au plafond, le système fait des hypothèses fortes sur les formes et les motifs que nous pouvons observer sur le plafond. Kim et al. ont exploité le fait que, dans leur environnement cible, les repères au plafond sont de classes connues, *c.-à-d.*, des repères circulaires représentant des *lampes*, des *hautparleurs*, des *alarmes incendie*, *etc.* (voir la figure 4.13). Ils ont alors utilisé ces informations extraites des images dans un cadre FastSLAM pour estimer la pose du robot et construire une carte [134].

Une approche similaire a été utilisée par Hwang et al., leur système détecte trois types de repères : *les coins*, *les lampes circulaires* et *les portes*. Le système extrait les coins à l'aide du détecteur FAST *(Features from Accelerated Segment Test)* et détecte les lampes à partir des zones les plus lumineuses de l'image du plafond à l'aide du détecteur de contours de Canny et d'un algorithme de *raccord géométrique de cercles (geometric circle-fitting)*. Les portes, quant à elles, sont détectées dans ce système à partir des lignes verticales, prises par paire en cherchant leur ligne horizontale d'intersection. Ces caractéristiques sont ensuite utilisées dans un cadre d'EKF-SLAM, en exploitant l'odométrie du robot pour la prédiction et les différentes caractéristiques pour apporter des corrections au filtre [135].

Figure 4.13 – Détection des cercles sur le plafond dans [134].

D'autres approches font des hypothèses sur les bords du plafond. Choi et al. ont proposé un EKF-SLAM basé sur une caméra monoculaire. Ce système utilise les bords entre le plafond et les murs (décrites sous formes de lignes avec des contraintes d'intersection) pour construire une carte des caractéristiques. Les auteurs ont exploité le fait que le plafond

occupe la majeure partie des images du plafond, tandis que les murs occupent le reste de l'image [136]. Avec cette même hypothèse, d'autres approches (figure 4.14) extraient les bords du plafond et les comparent à un plan de construction connu préalablement en utilisant une approche de localisation en Monte Carlo *(MCL - Monte Carlo Localization)*, avec un modèle d'observation basé sur la densité d'espace de plafond *(CSD - Ceiling Space Density)* [137].

Figure 4.14 – Extraction des contours du plafond pour appliquer la MCL dans [137].

Dans d'autres approches, l'environnement est augmenté par des marqueurs visuellement distinctes qui sont placées sur le plafond pour faciliter la tâche de localisation. Li et al. ont conçu et placé au plafond des repères artificiels facilement détectables (figure 4.15). Dans ce cas, les images de la caméra au plafond sont traitées pour détecter, identifier et trouver la position et l'orientation de la caméra par rapport aux repères. Les informations obtenues sont ensuite utilisées dans un algorithme d'optimisation de graphes dans un cadre d'estimation bayésien [138].

Figure 4.15 – Marqueur placé sur le plafond dans [138].

Cependant, la plupart de ces hypothèses ne tiennent pas dans des environnements comme les nôtres, à savoir, des entrepôts ouverts ou des espaces industriels.

Dans ces cas, l'espace de travail peut être trop large pour pouvoir observer le plafond et les murs sur la même image ; les plafonds de tels espaces ont tendance à être hauts (souvent répartis à différents niveaux de hauteur dans l'environnement) et peuvent contenir des surfaces inclinées.

Nous ne pouvons donc pas poser des hypothèses sur la planéité du plafond, sur l'observabilité des murs ou sur la forme ou la nature des motifs présents sur le plafond, d'où l'intérêt d'utiliser une méthode générique pour se localiser par rapport au plafond.

4.7 Notre contribution

Afin de surmonter les problèmes décrits ci-dessus, nous proposons d'appliquer une approche d'odométrie visuelle *générique* au flux d'images acquis d'une caméra verticale. Cela permet au robot d'évoluer sur différents types d'environnements sans faire d'hypothèses particulières sur la configuration du plafond et sans nécessiter des modifications intrusives de ce dernier.

Nous avons choisi d'utiliser la *Direct Sparse Odometry (DSO)*, ou l'*odométrie directe éparse*. Étant une approche directe, la DSO permet de réaliser un suivi de mouvement même dans des régions avec peu de caractéristiques distinctes et dans des zones à faible contraste et à faible texture. Notre objectif ici est de proposer et d'évaluer un cadre générique pour observer le plafond, en évitant de faire des hypothèses sur les formes ou sur les repères observables sur la surface du plafond.

Une autre caractéristique fondamentale de notre proposition réside dans son caractère non intrusif. En effet, d'une manière générale, l'intégration de capteurs ou de dispositifs (actifs ou même passifs) dans un environnement préexistant peut entraîner des perturbations, altérant ainsi la dynamique naturelle de cet espace. Cela entraîne également un coût supplémentaire de déploiement et de maintenance de cette infrastructure.

Cependant, grâce à l'utilisation d'une caméra de vision au plafond avec un algorithme de vision assez générique, notre méthode évite toute modification structurelle ou installation intrusive, réduisant ainsi le coût et la complexité de la mise en service et de la maintenance de notre système.

4.8 Évaluation de la VO verticale Ceiling-DSO

Dans cette section, nous proposons et évaluons la Ceiling-DSO, un système d'odométrie visuelle basé sur la *Direct Sparse Odometry (DSO)* et utilisé avec une caméra de vision au plafond. Ainsi, la méthode DSO serait présentée en détails dans cette section.

Pour but d'évaluer et de valider le système Ceiling-DSO, nous avons mené des expériences dans un environnement industriel réel, en utilisant une plateforme expérimentale de type *ez-Wheel SWD® Starter Kit*. Nous avons testé différents paramètres de la DSO afin d'observer et d'analyser leur influence sur les performances et la qualité des trajectoires estimées. Ces analyses ont été réalisées en s'appuyant sur des trajectoires de références estimées partir des données du LiDAR enregistrées sur les mêmes séquences.

4.8.1 La DSO (Direct Sparse Odometry)

La DSO *(Direct Sparse Odometry)*, ou *odométrie directe éparse* [62] est une méthode d'odométrie visuelle directe à base d'une caméra monoculaire. Contrairement aux méthodes indirectes (basées sur les caractéristiques), où nous n'utilisons qu'un ensemble de caractéristiques extraites des images pour estimer les mouvements de la caméra, les méthodes directes utilisent les informations de tous les pixels de l'image.

Nous pouvons diviser les méthodes directes en trois grandes catégories : *les méthodes denses*, qui traitent un grand nombre de points en utilisant tous les pixels de l'image, donnant, dans la plupart des cas, un lourd algorithme de traitement. Par ailleurs, les *méthodes semi-denses* tentent de réduire le nombre de points traités pour obtenir un temps d'exécution raisonnable. Les *méthodes éparses* quant à elles, sélectionnent uniquement un petit sous-ensemble de points à traiter.

La DSO fait partie de la troisième catégorie ; elle utilise une stratégie basée sur le gradient de l'image pour sélectionner uniformément les points candidats dans les régions à fort contraste. Ces points candidats sont ensuite utilisés dans un processus de *minimisation d'erreur photométrique*.

En tant que méthode directe, la DSO nécessite que tous les pixels d'une image soient capturés simultanément. Une caméra qui permet une telle acquisition est appelée caméra à *obturateur global (Global Shutter)*. Pour les caméras à *obturateur roulant (Rolling Shutter)*, les lignes de pixels de l'image sont capturées de manière séquentielle ; par conséquent, l'estimation de mouvement par la DSO avec une telle caméra accumulera une dérive systématique causée par cette acquisition séquentielle. Toutefois, cet effet peut être pris en compte en le modélisant ou en le mesurant. Par exemple, SCHUBERT et al. ont proposé une méthode basée sur DSO qui impose une contrainte d'obturateur roulant sur le modèle d'acquisition, cela leur permet d'estimer le temps de capture [139], et par conséquent, de compenser la dérive causée par l'acquisition séquentielle, ce qui a permis d'améliorer l'estimation de trajectoire obtenue à partir d'une caméra à obturateur roulant.

La structure de la DSO est divisée en deux parties étroitement liées qui s'exécutent en parallèle, un *front-end* et un *back-end*. Le *front-end* gère l'initialisation des paramètres de l'algorithme et s'occupe de la sélection des points et des images clés. Ces derniers sont ensuite envoyés au *back-end* qui réalise l'optimisation et la correction, produisant ainsi les estimations des poses et du nuage de points de la carte associée. L'organigramme de la figure 4.16 résume la structure globale de la DSO.

4.8.1.1 Formulation

Nous adoptons la même formulation de la DSO originale [62], ainsi, nous modélisons le capteur comme une caméra sténopé, avec $\mathbf{K}$ sa matrice intrinsèque utilisée pour la projection géométrique des points 3D dans le plan 2D d'image Ω tel que : $\Pi_{\mathbf{K}} : \mathbb{R}^3 \rightarrow \Omega$ et pour la rétroprojection vers le monde 3D à partir d'un point 2D (dans le plan d'image) et sa profondeur $\Pi_{\mathbf{K}}^{-1} : \Omega \times \mathbb{R} \rightarrow \mathbb{R}^3$.

Dans la DSO, un calibrage photométrique est pris en considération [120]. Pour une image « i », la caméra observe l'intensité brute des pixels $I_i^{RAW} : \Omega \rightarrow [0, 255]$, qui peut être définie en fonction de l'éclairement énergétique *(irradiance)* B_i, le temps d'exposition t_i, la fonction non-linéaire de réponse $G : \mathbb{R} \rightarrow [0, 255]$, et l'atténuation de l'objectif *(vignetting)* $V : \Omega \rightarrow [0, 1]$:

$$I_i^{RAW}(\mathbf{x}) = G(t_i V(\mathbf{x}) B_i(\mathbf{x})) \tag{4.23}$$

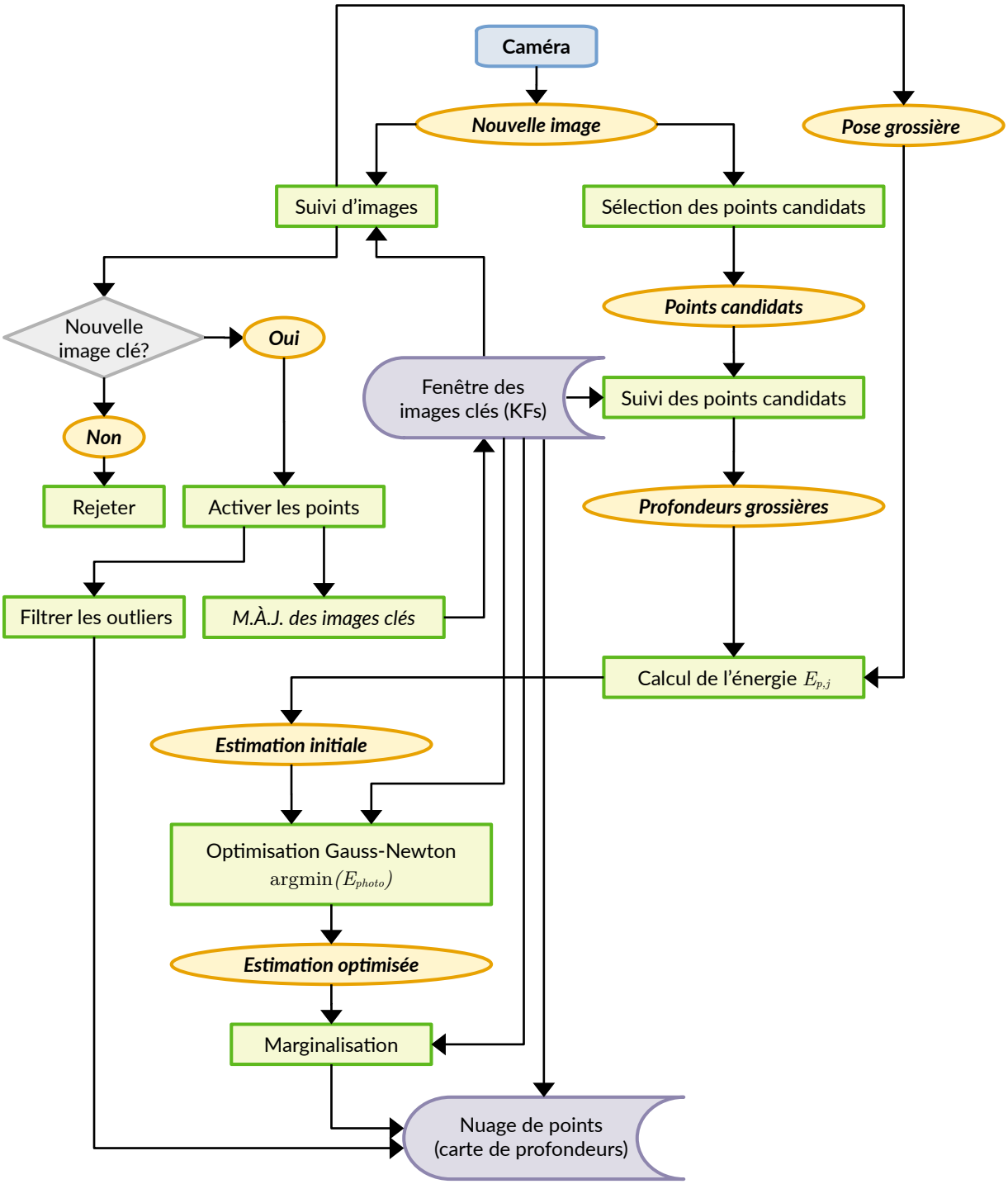


Figure 4.16 – Organigramme global de la méthode DSO

À partir de l'équation (4.23), l'image photométriquement corrigée I_i peut être calculée en inversant la fonction non-linéaire de réponse G et en supprimant l'effet de vignettage sur l'image, ce qui donne l'équation (4.24) :

$$I_i(\mathbf{x}) \triangleq t_i B_i(\mathbf{x}) = \frac{G^{-1}(I_i^{RAW}(\mathbf{x}))}{V(\mathbf{x})} \tag{4.24}$$

L'erreur photométrique sur toutes les images séléctionnées (les N dernières images clés) est définie comme suit :

$$E_{\text{photometric}} = \sum_{i \in N} \sum_{\mathbf{p} \in P_i} \sum_{j \in obs(\mathbf{p})} E_{i,\mathbf{p},j} \tag{4.25}$$

Avec N le nombre total d'images, P_i l'ensemble des points dans la i-ième image, j itère sur $obs(\mathbf{p})$ qui représente l'ensemble de toutes les images à partir desquelles le point $\mathbf{p}$ est visible, et $E_{i,\mathbf{p},j}$ est le terme d'erreur partielle définie sous forme d'une somme pondérée des normes d'Huber [140] calculées dans un motif de voisinage $\mathcal{N}_\mathbf{p}$ du point $\mathbf{p}$ (figure 4.17) :

$$E_{i,\mathbf{p},j} = \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{p}} w_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma \tag{4.26}$$

$$\begin{aligned} \text{avec la norme d'Huber} \quad \|\alpha\|_\gamma &\triangleq \begin{cases} |\alpha|^2 & \text{pour} \quad |\alpha| < \frac{\gamma}{2} \\ \gamma \cdot (|\alpha| - \frac{\gamma}{4}) & \text{sinon} \end{cases} \\ \text{et} \quad w_\mathbf{p} &\triangleq \frac{c^2}{c^2 + \|\nabla I_i(\mathbf{p})\|_2^2} \quad \text{avec} \quad c \in \mathbb{R} \end{aligned} \tag{4.27}$$

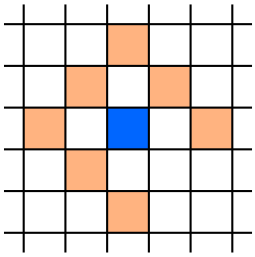


Figure 4.17 – Le motif de voisinage utilisé, tiré de [62]. Le point central (en bleu) c'est le point d'intérêt $\mathbf{p}$. Les points autour (en orange) constituent l'ensemble des points voisins $\mathcal{N}_\mathbf{p}$.

Le facteur de pondération est inversement proportionnel à la norme du gradient, il pénalise donc les pixels de haut gradient. Ceci permet de réduire l'influence des différences de luminosité dans une même image sur le suivi de mouvements sur les images successives.

La norme d'Huber est une norme hybride ℓ_1/ℓ_2, *c.-à-d.*, c'est une combinaison de la *norme absolue* avec la *norme euclidienne*, mais contrairement à la norme absolue ℓ_1, la norme d'Huber a la particularité d'être différentiable sur $\mathbb{R}$ en plus d'être robuste aux valeurs aberrantes. Par conséquent, la norme d'Huber fournit une mesure d'erreur appropriée pour une optimisation basée sur le gradient. La figure 4.18 illustre le graphe de la norme d'Huber pour des valeurs de λ de 1, 2 et 3 en comparaison avec les normes ℓ_1 et ℓ_2.

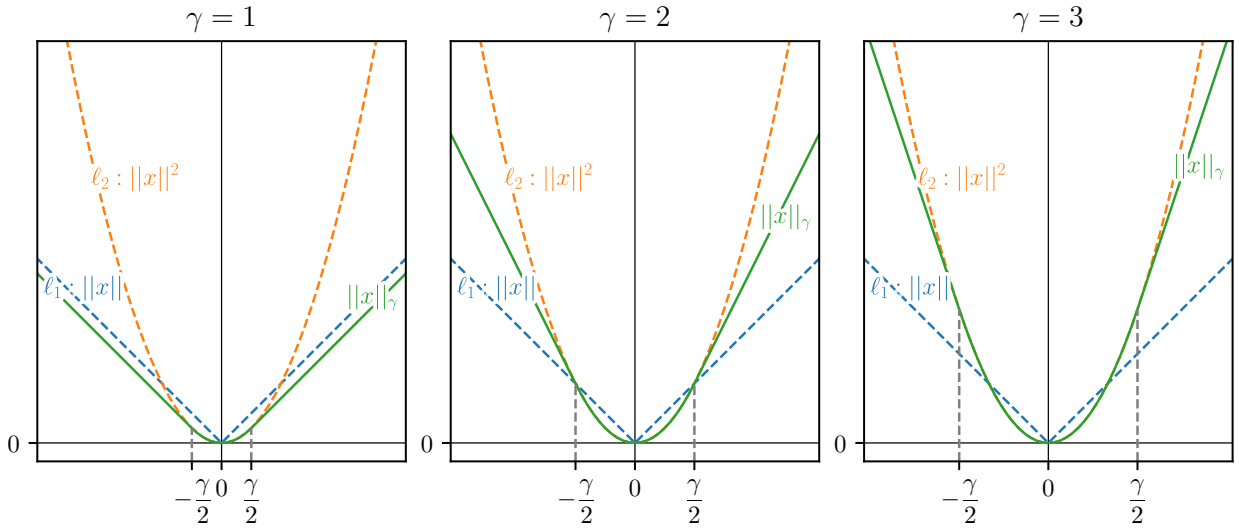


Figure 4.18 – Les graphes des normes ℓ_1 et ℓ_2 ainsi que la norme d'HUBER avec des valeurs de γ de 1, 2 et 3 du gauche à droite. La norme d'HUBER est équivalente à la norme ℓ_2 dans l'intervalle $[-\frac{\gamma}{2}, \frac{\gamma}{2}]$, et équivalente à la norme ℓ_1 au-delà (avec une pente γ), ce qui permet d'avoir une dérivabilité sur $\mathbb{R}$.

Le terme résiduel dans la somme (4.26) inclut la différence entre l'intensité du point $\mathbf{p}$ dans l'image courante i et son intensité dans toutes les images $obs(\mathbf{p})$ dans lesquelles le point $\mathbf{p}$ est visible, indexées par j.

L'alignement direct des images est basé fondamentalement sur *l'hypothèse de constance de la luminosité* [118] ; *c.-à-d.*, le niveau de luminosité entre des images successives est supposé constant. Néanmoins, cette hypothèse est fortement violée dans les scénarios réels des systèmes d'odométrie/SLAM visuels, par exemple, lorsque le temps d'exposition de la caméra est automatiquement ajusté pour mieux s'adapter à la luminosité moyenne de la scène.

Pour permettre le fonctionnement avec des temps d'exposition inconnus, l'intensité est modélisée sous forme d'une *fonction affine de transfert de luminosité*. Cela signifie que la DSO suppose que la luminosité d'un pixel évolue, d'image en image, d'une façon affine, donnant possibilité à modéliser des changements de luminosité dû aux différences de temps d'expositions entre les images.

$$I_{i+1} \triangleq e^{-a_i}(I_i - b_i) \tag{4.28}$$

Le terme e^{a_i} est paramétré logarithmiquement pour éviter qu'il devienne négatif ou qu'il déborde à cause de l'accumulation multiplicative d'erreurs.

Le point $\mathbf{p}'$ dans l'équation (4.26) représente la projection dans la j-ième image, du point $\mathbf{p}$ vu dans la i-ième image, avec $d_\mathbf{p}$ sa profondeur estimée. La matrice de projection dépend donc de la transformation partielle de mouvement de la caméra $\Delta\mathbf{T}_{j,i}$ entre les poses de la caméra aux instants « i » et « j », $\mathbf{T}_i$ et $\mathbf{T}_j$, respectivement.

$$\mathbf{p}' = \Pi_\mathbf{K}(\mathbf{R}\Pi_\mathbf{K}^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}) \quad \text{avec} \quad \Delta\mathbf{T}_{j,i} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} = \mathbf{T}_j\mathbf{T}_i^{-1} \tag{4.29}$$

Le terme de l'erreur dans l'équation (4.26) est ensuite minimisé par 6 itérations de l'optimisation de Gauss-Newton. L'optimisation se fait sur l'algèbre de Lie $\mathfrak{se}(3)$ [121], avec l'opérateur $\oplus$-gauche défini comme suit :

$$\oplus : \mathfrak{se}(3) \times \mathrm{SE}(3) \to \mathrm{SE}(3) \quad \text{avec} \quad \begin{cases} \mathbf{x}_i \in \mathfrak{se}(3), \quad \mathbf{T}_i \in \mathrm{SE}(3) \\ \mathbf{x}_i \oplus \mathbf{T}_i \triangleq e^{\widehat{\mathbf{x}_i}} \cdot \mathbf{T}_i \end{cases} \tag{4.30}$$

Les paramètres optimisés $\zeta \in \mathrm{SE}(3)^n \times \mathbb{R}^m$ incluent les paramètres géométriques extrinsèques (poses de la caméra et valeurs de profondeur inverses des points), intrinsèques (matrice de la caméra), et les paramètres photométriques (les paramètres du changement affine de la luminosité (a_i, b_i)).

Sur le manifold des mouvements rigides SE(3), soit ζ_0 le point d'évaluation de l'espace tangent du manifold, et $\mathbf{x} \in \mathfrak{se}(3)^n \times \mathbb{R}^m$ les mises à jour accumulées. L'estimation de l'état actuel devient alors $\zeta = \mathbf{x} \oplus \zeta_0$ avec l'opérateur $\oplus$-gauche étendu au-delà de SE(3) éléments comme une addition régulière.

L'erreur photométrique (4.25) est optimisée à l'aide d'un système Gauss-Newton défini par :

$$\mathbf{H} = \mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{J} \tag{4.31}$$

$$\mathbf{b} = -\mathbf{J}^\mathsf{T}\mathbf{W}\mathbf{r} \tag{4.32}$$

Où $\mathbf{r} \in \mathbb{R}^n$ est un vecteur regroupant tous les résidus, $\mathbf{W} \in \mathbb{R}^{n \times n}$ est la matrice diagonale de pondération, et $\mathbf{J} \in \mathbb{R}^{n \times d}$ est la jacobienne du vecteur des résidus $\mathbf{r}$.

Soit r_k un élément du vecteur des résidus $\mathbf{r}$, et $\mathbf{J}_k$ sa ligne associée dans la jacobienne. Le résidu regroupe : $\mathbf{T}_i$ et $\mathbf{T}_j$ les poses de la caméra aux images « i » et « j », respectivement, $d_\mathbf{p}^{-1}$ la profondeur inverse, $\mathbf{K}$ la matrice intrinsèque, et a_i, a_j, b_i, b_j les paramètres de la fonction affine de transfert de luminosité, donnant ainsi $(\mathbf{T}_i, \mathbf{T}_j, d_\mathbf{p}, a_i, a_j, b_i, b_j) = \mathbf{x} \oplus \zeta_0$.

$$r_k = (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}}(I_i - b_i) \tag{4.33}$$

$$\mathbf{J}_k = \frac{\partial r_k((\delta + \mathbf{x}) \oplus \zeta_0)}{\partial \delta} \tag{4.34}$$

$$\text{Avec} \quad \mathbf{p}' = \Pi_\mathbf{K}(\mathbf{R}\Pi_\mathbf{K}^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}) \quad \text{et} \quad \Delta\mathbf{T}_{j,i} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} = \mathbf{T}_j\mathbf{T}_i^{-1}$$

L'optimisation de Gauss-Newton est effectuée ensuite sur une fenêtre glissante de N_f images clés, les points au-delà de cette fenêtre sont marginalisés.

Dans notre implémentation de la Ceiling-DSO, nous avons utilisé une version simplifiée de la DSO originale. Nous avons utilisé une caméra avec objectif sans effet de vignettage, par conséquent, il n'est pas nécessaire de réaliser la correction des effets d'atténuations, nous avons donc supposé :

$$V(\mathbf{x}) = 1; \qquad \forall \mathbf{x} \in \Omega \tag{4.35}$$

En plus, la réponse du capteur utilisé étant quasiment linéaire, nous avons donc supposé une fonction de réponse linéaire tel que :

$$G(\mathbf{x}) = \mathbf{x}; \qquad \forall \mathbf{x} \in \Omega \tag{4.36}$$

4.9 Résultats et validation

Dans cette section, nous présentons les outils matériels, logiciels et méthodologiques que nous avons utilisé pour valider notre approche. Nous commençons par décrire la plateforme expérimentale utilisée, puis le processus de la construction du jeu de données, et à la fin la présentation et l'analyse des résultats.

4.9.1 Plateforme expérimentale

Dans nos expérimentations, nous avons utilisé un prototype du robot industriel mobile nommé *SWD® Starter Kit* (figure 4.19). La plateforme est un robot modulaire à entraînement différentiel (*differential-drive*) propulsé par deux roues motorisées autonomes d'ez-Wheel, dotées de la nouvelle technologie *Safety Wheel Drive (SWD®)*.

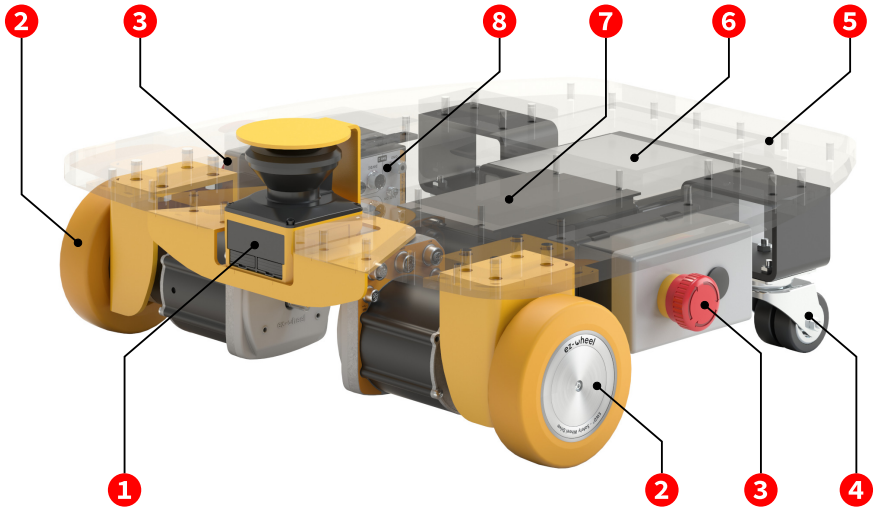


Figure 4.19 – Illustration des différents composants de la plateforme SWD® StarterKit. (1) Un LiDAR sécuritaire IDEC S2L ; (2) Deux roues ez-Wheel à base du moteur SWD® Core ; (3) Deux boutons d'arrêt d'urgence sur chaque côté ; (4) Deux roues folles (roulettes) ; (7) Châssis en plexiglass ; (6) Bloc de batteries ; (7) Un ordinateur embarqué *Neousys Nuvo-7002LP* ; (8) ez-Wheel SafetyHub, un boîtier d'interconnexion entre les roues, l'ordinateur embarqué et les batteries.

Ce prototype est une évolution de notre plateforme SmartTrolley [94], présentée précédemment dans le chapitre 3.

Dans sa version définitive, ce robot sera destiné au déplacement de fortes charges en environnement indoor, principalement pour une utilisation en milieux industriels. Les roues sont conçues et dimensionnées pour déplacer un maximum de 2 tonnes (2000kg) de charges. En revanche, le châssis de ce prototype est conçu en plexiglass pour faciliter son déplacement et sa maintenance lors des travaux de R&D, il n'est donc pas destiné à supporter des lourdes charges.

Pour des raisons de sécurité, le robot ne fonctionne qu'à faibles vitesses, atteignant une vitesse maximale de $5\text{km} \cdot \text{h}^{-1} \approx 1.4\text{m} \cdot \text{s}^{-1}$.

Le robot est équipé de deux codeurs incrémentaux, de deux caméras, une *Intel® RealSense™ D435i* orientée vers l'avant et une *Intel® RealSense™ 455* orientée vers le haut. Le robot intègre également un scrutateur laser sécuritaire *(Safety LiDAR)* de type *IDEC S2L*, d'une portée maximale de 40m.

La plateforme expérimentale (figure 4.20) intègre un ordinateur industriel embarqué *Neousys Nuvo-7002LP*, basé sur un processeur *Intel® Coffee lake Core™ i5* de 8e génération et une *SDRAM DDR4 2666/2400* de 16Go.

La partie logicielle repose sur le système d'exploitation *Ubuntu 20.04 LTS* avec à la fois le middleware robotique *ROS Noetic Ninjemys* et la nouvelle version *ROS 2 Iron Irwini*. Cela permettrait de porter et tester progressivement les briques que nous avons développées sous ROS à ROS 2. Nous avons utilisé ce calculateur pour collecter les données brutes des capteurs, ainsi que pour embarquer le contrôleur du robot [2] que nous avons implémenté à base du modèle cinématique à entraînement différentiel décrit dans le chapitre 3. Nous avons connecté l'ordinateur embarqué à un émetteur-récepteur RF, que nous utilisons pour piloter le robot à distance à l'aide d'une manette sans fil.

Figure 4.20 – Le robot mobile SWD® Starter Kit d'ez-Wheel.

2. Le code source du contrôleur du robot est disponible publiquement sur GitHub sous licence LGPL-2.1. Il est destiné aux robots à base de roues SWD® d'ez-Wheel et supporte ROS `ezWheelSAS/swd_ros_controllers`, et ROS 2 `ezWheelSAS/swd_ros2_controllers`.

La plateforme exploite la fonctionnalité de détection d'obstacles du bas-niveau fournie par le LiDAR de sécurité et intégrée dans les calculateurs des roues (concept décrit dans le chapitre 3). Ceci nous permet d'assurer une sécurité absolue. En effet, il n'est pas raisonnable de laisser le robot livré à lui-même si la partie logicielle présente des dysfonctionnements, notamment pendant la phase de développement. Le robot étant destiné au déplacement des fortes charges, il est extrêmement important de s'assurer que ce dernier arrive à éviter, d'une manière sûre, toutes sortes de collisions.

Pour assurer cela, nous avons configuré deux zones de détection au niveau du LiDAR. Une grande zone de limitation de vitesse *(Safety-Limited Speed - SLS)* déclenchée lorsque le robot s'approche d'un obstacle et une zone plus petite d'interdiction directionnelle *(Safe Direction Indication - SDI)*, utilisée pour interdire le déplacement vers un obstacle proche.

Le SDI est implémenté d'une manière intelligente, il impose l'interdiction d'avancer si l'obstacle est devant le robot et/ou l'interdiction de reculer si l'obstacle est derrière. Nous avons implémenté ces comportements de détection et de réponse au niveau le plus bas, entre les calculateurs des roues et le LiDAR de sécurité. Les signaux SDI et SLS sont envoyés du LiDAR aux roues motorisées directement via des sorties sécurisées *(OSSD - Output Signal Switching Device)*. Voir le chapitre 3 pour plus de détails.

4.9.2 Jeu de données

Nous avons réalisé notre expérience dans un espace ouvert intérieur de 21×15m dans les locaux de l'entreprise ez-Wheel. Nous avons recueilli un jeu de données comportant les données brutes de l'odométrie, les images de la caméra stéréoscopique orientée vers le haut, les images de la caméra stéréoscopique orientée vers l'avant et les données du scrutateur laser (LiDAR).

Le plafond de notre environnement d'essai est incliné, avec des hauteurs comprises entre 4 et 6 mètres. Les formes et les repères sur le plafond peuvent différer d'une région à une autre ; la figure 4.21 présente quelques exemples d'images du plafond vues par la caméra orientée vers le haut.

Figure 4.21 – Exemples d'images du plafond de l'environnement d'essai vu par la caméra verticale.

4.9.3 Méthodologie

Nous avons évalué la Ceiling-DSO sur un ensemble de séquences du jeu de données que nous avons collecté. Nous avons étudié l'influence du changement de la *taille des images d'entrée*, de la *fréquence d'images* et de la *taille maximale de la fenêtre d'optimisation* sur la qualité de la trajectoire estimée et le temps d'exécution. Le but d'une telle évaluation est d'identifier les bonnes combinaisons de paramètres qui permettent une utilisation en temps réel.

Nous avons utilisé les données du LiDAR pour calculer une trajectoire de vérité terrain à l'aide du paquet ROS LaMa SLAM [116], disponible en open-source. La carte de l'environnement d'essai calculée à l'aide de cet algorithme SLAM est représentée sur la figure 4.22.

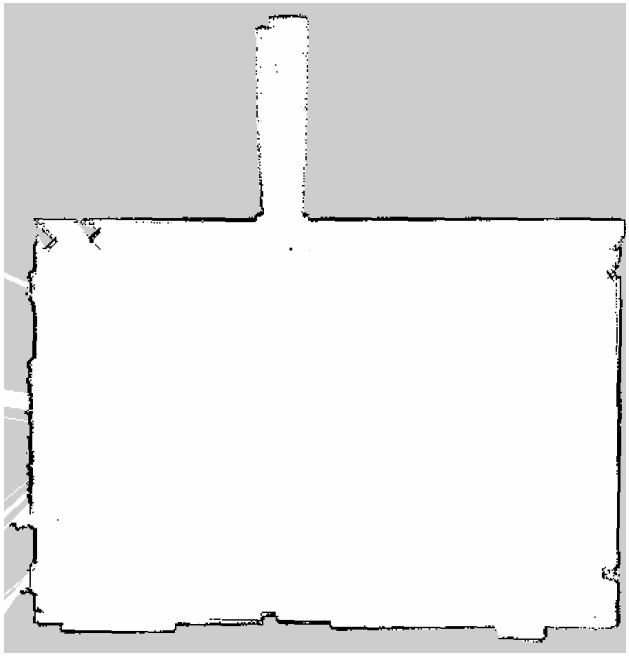


Figure 4.22 – La carte de l'environnement d'essai estimée à l'aide du LaMa SLAM à partir des données du LiDAR.

Nous avons mené nos expérimentations sur une surface plate, avec une caméra orientée vers le plafond et un LiDAR 2D monté en avant du robot, lui permettant ainsi d'observer un plan horizontal autour de lui. Nous désignons la trajectoire 2D de la vérité terrain par $\mathcal{G}$, la DSO utilise une caméra monoculaire, par conséquent, sa trajectoire estimée $\mathcal{P}$ n'est valide qu'à une échelle près $\lambda \in \mathbb{R}^+$. La trajectoire de la DSO est tridimensionnelle, alors que la vérité terrain n'est que bidimensionnelle. Ayant réalisé les mouvements pendant nos expériences sur une surface plane (2D), nous pouvons comparer les deux trajectoires $\mathcal{P}$ et $\mathcal{G}$ après un alignement.

Pour aligner les trajectoires estimées et les comparer à la réalité terrain, nous utilisons une approche similaire à celle de Zhang et al. [141]. Nous synchronisons d'abord les deux trajectoires en utilisant l'horloge globale du système ROS. Notons les n positions synchronisées de la vérité terrain et de l'odométrie visuelle par $\mathcal{G}'$ et $\mathcal{P}'$, respectivement, définies comme :

$$\begin{aligned} \mathcal{G}' &= \{\mathbf{g} \mid \|t_{\mathbf{p}} - t_{\mathbf{g}}\| < \tau\} \\ \mathcal{P}' &= \{\mathbf{p} \mid \|t_{\mathbf{p}} - t_{\mathbf{g}}\| < \tau\} \end{aligned} \quad \text{avec} \quad \begin{cases} \forall \mathbf{g} \in \mathcal{G} \\ \forall \mathbf{p} \in \mathcal{P} \\ \tau \in \mathbb{R}^+ \end{cases} \tag{4.37}$$

Avec $t_{\mathbf{p}}$ et $t_{\mathbf{g}}$ sont les estampilles temporelles (horodatages, *timestamps*) des points $\mathbf{p}$ et $\mathbf{g}$, respectivement, et τ est le seuil de synchronicité.

Nous estimons la similarité globale $\mathbf{S} \in \mathrm{Sim}(3)$ entre les trajectoires synchronisées de la vérité terrain et l'odométrie visuelle, $\mathcal{G}'$ et $\mathcal{P}'$ respectivement. Nous définissons la transformation de similarité $\mathbf{S}$ tel que :

$$\mathbf{S} \triangleq \begin{bmatrix} \lambda \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \tag{4.38}$$

Par convenance, nous notons $\mathbf{S} = (\mathbf{R}, \mathbf{t}, \lambda)$, où $\mathbf{R} \in \mathrm{SO}(3)$ est la rotation tridimensionnelle, $\mathbf{t} \in \mathbb{R}^3$ est la translation tridimensionnelle, et $\lambda \in \mathbb{R}^+$ est le facteur d'échelle.

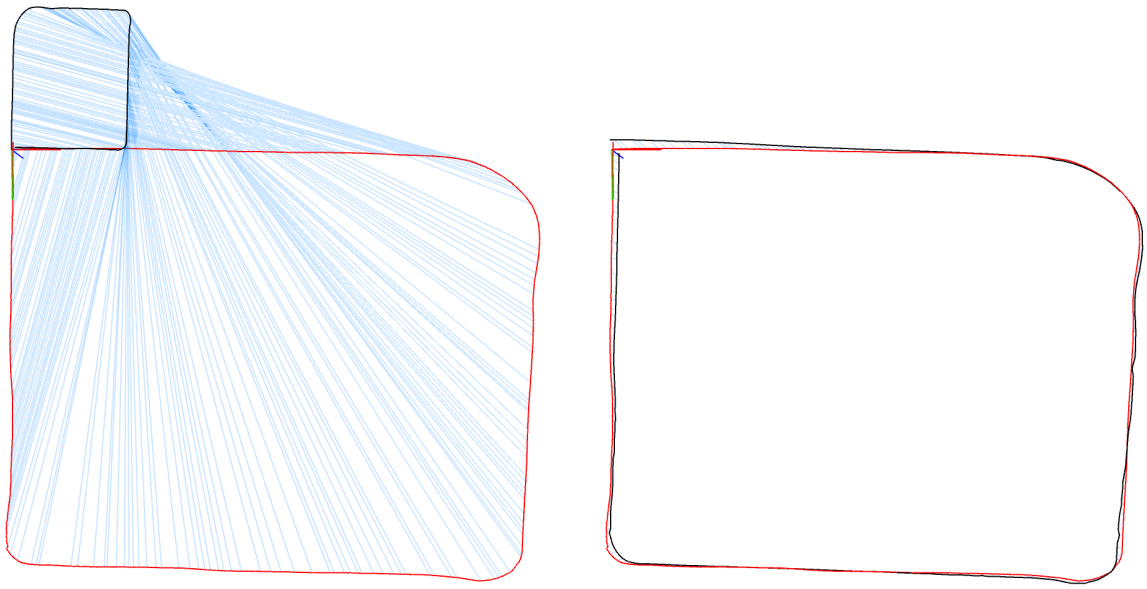
Nous pouvons exprimer l'alignement comme un problème de moindres carrés, pour minimiser l'erreur entre les paires de positions synchrones $\mathbf{p}_i \in \mathcal{P}'$ et $\mathbf{g}_i \in \mathcal{G}'$. La transformation qui décrit l'alignement global optimal $\overset{\star}{\mathbf{S}} = (\overset{\star}{\mathbf{R}}, \overset{\star}{\mathbf{t}}, \overset{\star}{\lambda})$ peut-être exprimé de la manière suivante :

$$\overset{\star}{\mathbf{S}} = \underset{\mathbf{R},\mathbf{t},\lambda}{\arg\min} \sum_{i=0}^{n} \|\mathbf{g}_i - (\lambda \mathbf{R} \mathbf{p}_i + \mathbf{t})\|^2 \tag{4.39}$$

La trajectoire estimée de l'odométrie visuelle $\mathcal{P}$ est ensuite alignée en utilisant la similarité optimale $\overset{\star}{\mathbf{S}}$, donnant ainsi la trajectoire corrigée $\overset{\star}{\mathcal{P}}$, tel que :

$$\forall \mathbf{p} \in \mathcal{P} : \ \overset{\star}{\mathcal{P}} = \{\overset{\star}{\lambda} \overset{\star}{\mathbf{R}} \mathbf{p} + \overset{\star}{\mathbf{t}}\} \tag{4.40}$$

La figure 4.23 montre une trajectoire d'odométrie visuelle tracée en noir et la trajectoire de référence tracée en rouge, avant (*fig.* 4.23a) et après (*fig.* 4.23b) l'alignement. Les lignes en bleu entre les deux trajectoires représentent les distances entre les points synchronisés, le système de l'équation (4.39) minimise ces distances.



(a) Avant l'alignement **(b)** Après l'alignement

Figure 4.23 – Exemple d'une trajectoire de la séquence (Seq1) ; l'odométrie visuelle (en noir) et la trajectoire de référence (en rouge). Les lignes (en bleu) représentent les distances entre les points synchronisés.

Ainsi, pour réaliser notre analyse quantitative des résultats, nous évaluons les *erreurs relatives (ER)* [141] entre la vérité terrain et les trajectoires de Ceiling-DSO. Nous considérons ensuite la norme euclidienne de l'erreur de position pour évaluer les résultats obtenus.

4.9.4 Résultats et discussion

Nous avons testé Ceiling-DSO sur plusieurs séquences de notre jeu de données. Nous avons sélectionné deux séquences (que nous notons ici par *Seq1* et *Seq2*) à présenter dans cette section. La première est une trajectoire simple en forme de carré et la seconde est une trajectoire plus complexe avec une grande fermeture de boucle composée de plusieurs petites boucles.

Premièrement, nous fournissons des résultats qualitatifs où nous alignons et traçons les trajectoires alignées par rapport à la vérité terrain. Nous avons testé un total de 24 trajectoires par séquence, en itérant sur toutes les combinaisons de paramètres testés, à savoir, une *taille d'image* de 848×480 ou 424×240, une *fréquence d'images* de 3, 6, 15 ou 30fps et une *taille maximale de la fenêtre d'optimisation* de 5, 7 ou 15.

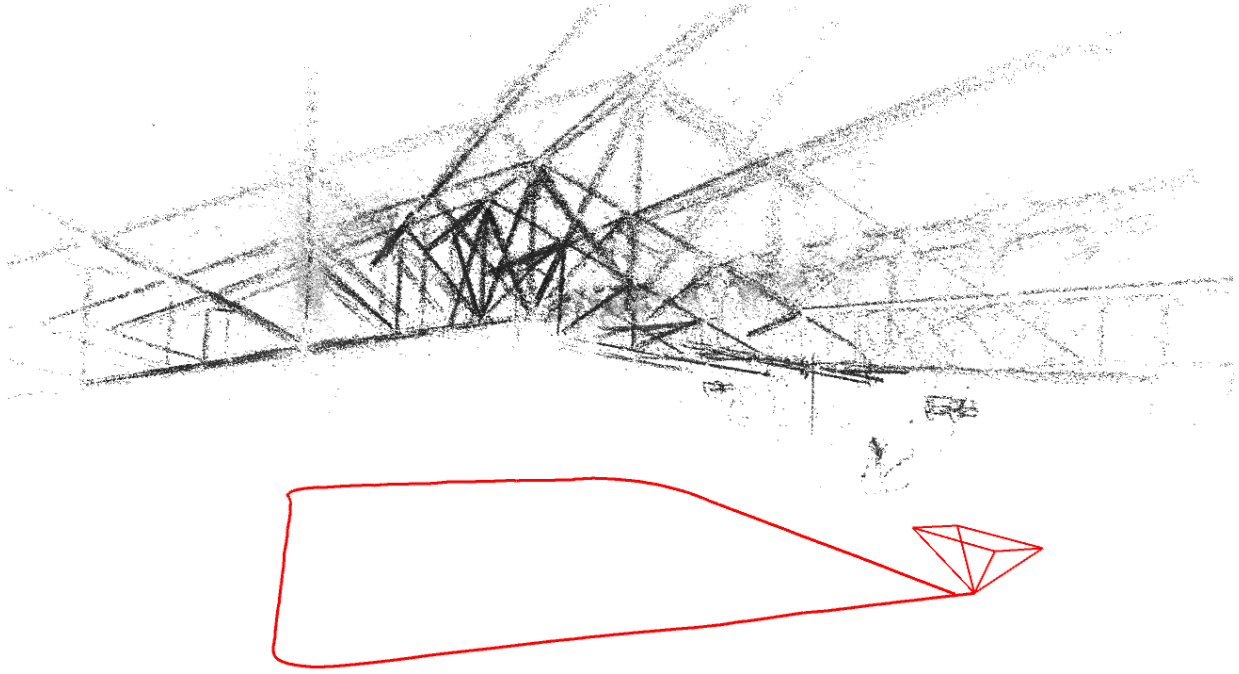


Figure 4.24 – La trajectoire et le nuage de points de la carte construite (*Seq1*).

La figure 4.24 montre la trajectoire ainsi que le nuage de points résultant de la Ceiling-DSO sur la séquence (*Seq1*), la consistance du nuage de points peut permettre une appréciation visuelle de la qualité de la carte, et conjointement, de la trajectoire estimée.

Compte tenu des trajectoires estimées à partir des deux séquences, les figures 4.25 et 4.26 montrent l'effet de la modification de la fréquence d'images et de la taille de l'image tout en fixant la taille maximale de la fenêtre d'optimisation à 7. Nous notons que l'erreur sur la trajectoire augmente lors de l'utilisation de basses fréquences d'images qui sont inférieures à 15 images par seconde. La différence est plus visible dans la trajectoire complexe (figure 4.26).

Nous devons souligner que le choix de la fréquence d'images est également lié à la vitesse du robot. Pour les robots se déplaçant à grande vitesse, des flux de fréquences d'images plus élevées seront nécessaires.

Étant donné que la Ceiling-DSO exécute une mise en correspondance *multi-échelles* ou *grossière à fine (coarse-to-fine matching)*, l'augmentation de la taille de l'image peut théoriquement aider à raffiner l'estimation ; cependant, nos tests n'ont montré aucun effet significatif lors de la réduction de la taille de l'image de 848×480 à 424×240.

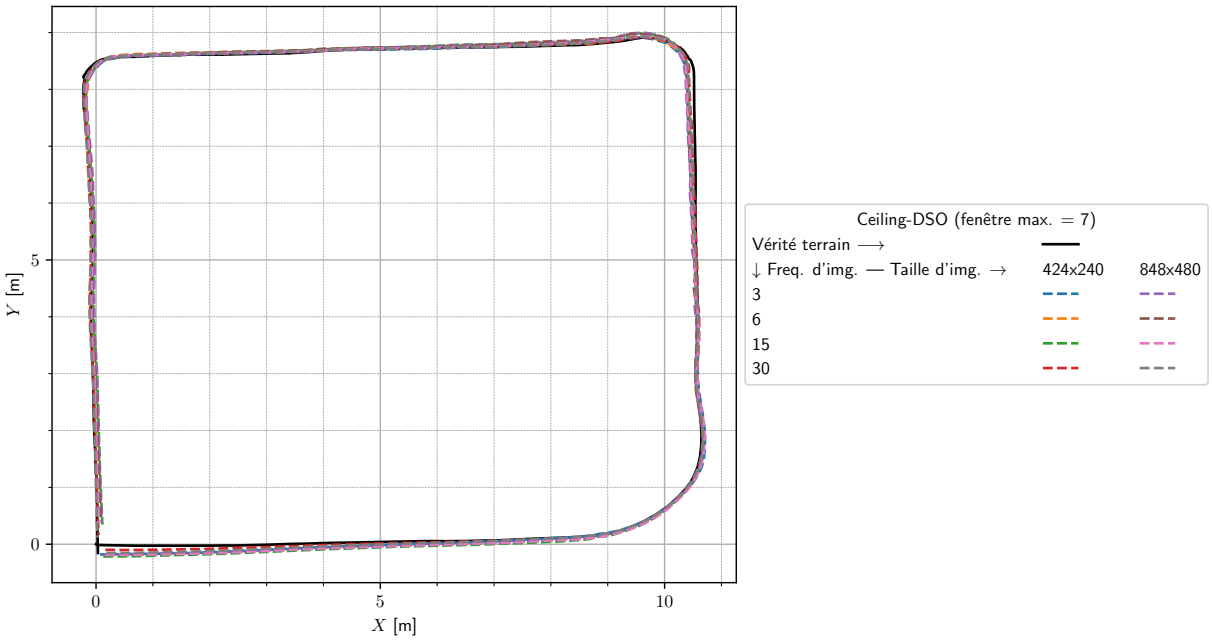


Figure 4.25 – Trajectoires calculées en variant la taille et la fréquence d'images d'entrée (*Seq1*).

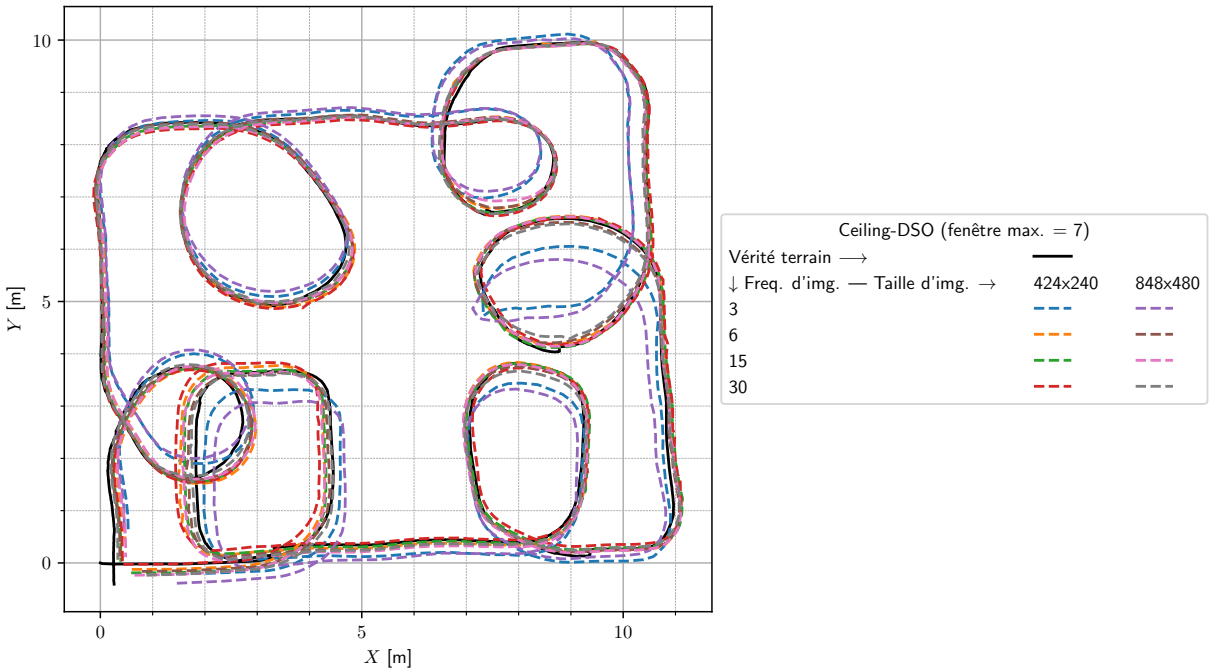


Figure 4.26 – Trajectoires calculées en variant la taille et la fréquence d'images d'entrée (*Seq2*).

Les figures 4.27 et 4.28 montrent l'effet de la modification de la taille de l'image et de la taille maximale de la fenêtre d'optimisation sur les trajectoires tout en fixant la fréquence d'images à 30fps. Ces trajectoires montrent de légères améliorations lors de l'augmentation de la taille maximale de la fenêtre.

Ceci se justifie par le fait que l'étape d'optimisation effectue un *ajustement de faisceaux (BA - Bundle Adjustment)* en local[3]. Ainsi, l'utilisation d'une taille de fenêtre plus grande devrait aider à augmenter la précision dans des trajectoires plus complexes. Toutefois, nos tests ont montré qu'une précision acceptable peut être atteinte avec l'utilisation d'une fenêtre de taille maximale de 7.

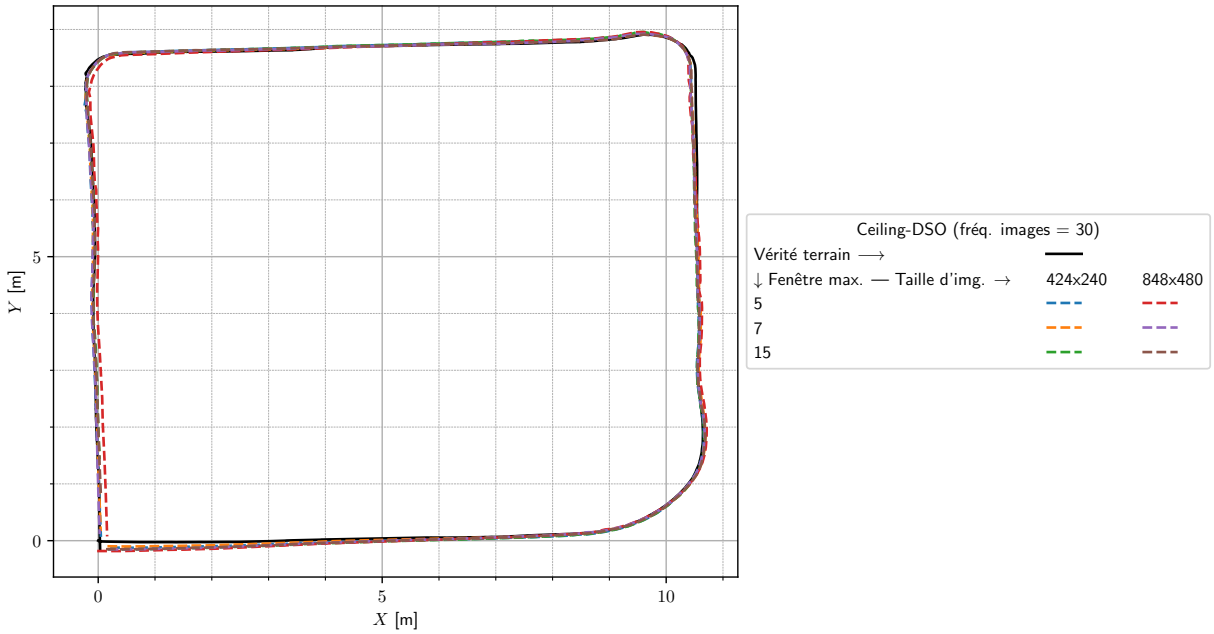


Figure 4.27 – Trajectoires calculées en variant la taille d'image et la taille de fenêtre maximale (*Seq1*).

Pour l'analyse quantitative, nous traçons la norme euclidienne de l'erreur relative sur la position $\epsilon_i = \sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}$ en fonction de la distance parcourue. Les figures 4.29 et 4.30 résument l'erreur relative pour les deux séquences, en faisant varier un paramètre à la fois tout en fixant les deux autres à des valeurs fixes (valeurs de 848×420 pour la taille de l'image, de 30 pour la fréquence d'images et de 7 pour la taille maximale de la fenêtre d'optimisation).

Nous pouvons observer que les erreurs les plus importantes se sont produites lors de la diminution de la fréquence d'images au dessous de 15 images par seconde. Tandis que pour les autres paramètres, nous n'avons constaté qu'une influence minime.

3. L'*ajustement de faisceaux (ou Bundle Adjustment)* est évoqué généralement dans un contexte de minimisation d'une erreur géométrique (erreur de reprojection). Mais dans le contexte d'une méthode directe, l'ajustement de faisceaux est réalisé en minimisant l'erreur photométrique (l'équation 4.25 dans le cas de DSO).

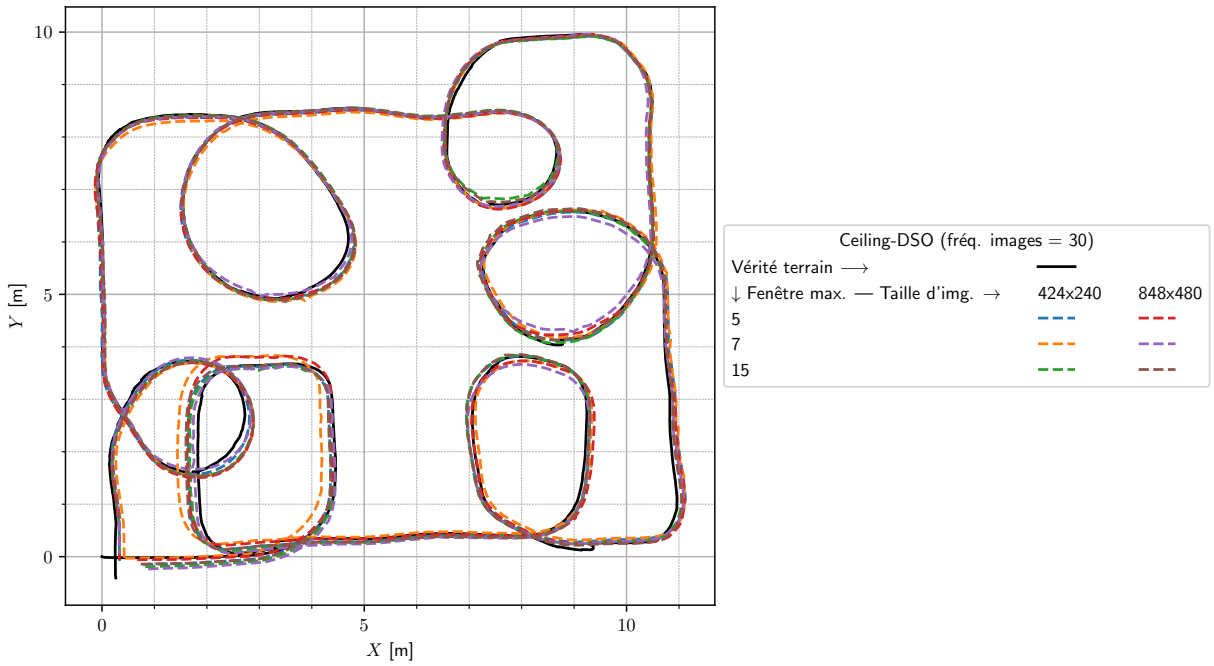


Figure 4.28 – Trajectoires calculées en variant la taille d'image et la taille de fenêtre maximale (*Seq2*).

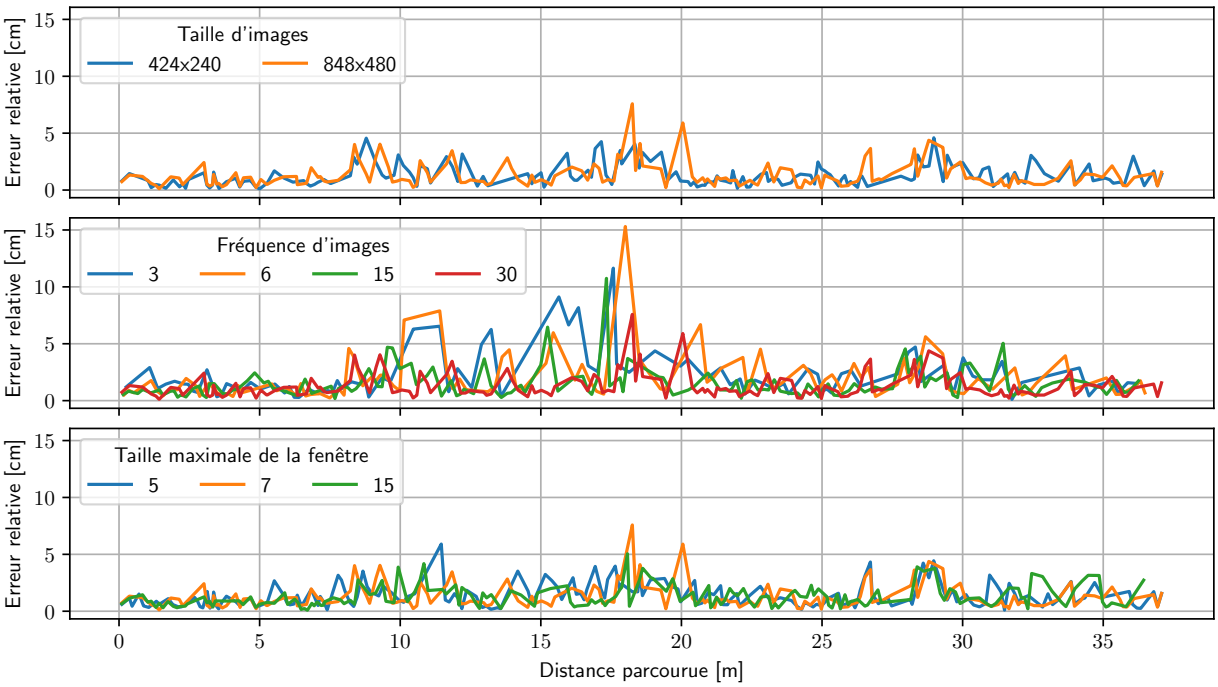


Figure 4.29 – L'erreur relative entre la vérité terrain et la trajectoire Ceiling-DSO en variant la fréquence d'images, la taille d'image et la taille maximale de fenêtre d'optimisation (*Seq1*).

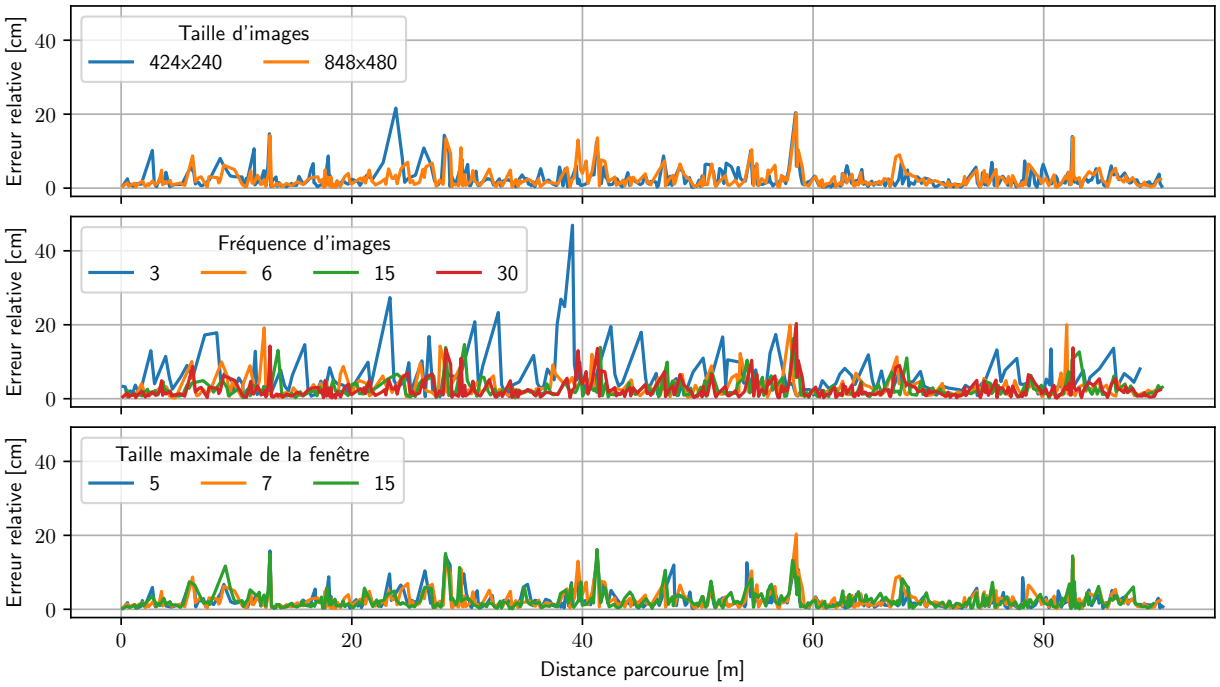


Figure 4.30 – L'erreur relative entre la vérité terrain et la trajectoire Ceiling-DSO en variant la fréquence d'images, la taille d'image et la taille maximale de fenêtre d'optimisation (*Seq2*).

Pour mieux comprendre la distribution de l'erreur relative tout au long de la trajectoire et pour chaque combinaison de paramètres, nous fournissons la figure 4.31. Cette figure montre des *boîtes à moustaches* (ou *diagramme en boîtes, box plots*) de l'erreur relative par rapport aux paramètres testés. Cette représentation met en évidence, pour chaque combinaison, la distribution estimée caractérisée par la boîte qui délimite les premier et troisième quartiles Q_1 et Q_3, ainsi que le deuxième quartile Q_2, qui représente la médiane. Les moustaches (les lignes) qui se prolongent au-delà de la boîte marquant la variabilité en dehors des quartiles inférieurs et supérieurs dans la limite de $1.5 \times \text{IQR} = 1.5 \times (Q_3 - Q_1)$, IQR pour *intervalle interquartile*. Le graphique montre côte à côte des distributions d'erreurs relatives pour les deux séquences.

La sous-figure de droite de 4.31 montre le temps d'exécution normalisé, calculé comme $t_F \cdot f$ avec t_F le temps d'exécution par image en secondes et f la fréquence d'images en hertz (les deux valeurs extraites du tableau 4.1). Ce chiffre représente le temps nécessaire pour traiter une seconde d'images d'entrée. La ligne verte verticale tracée à $t = 1000$ms montre la limite pour réaliser le calcul en temps réel.

Sur la figure 4.31, la ligne rouge en pointillés coupe en deux les médianes triées des erreurs relatives. Nous nous concentrons sur les combinaisons de paramètres de la première moitié, car elles présentent des erreurs relatives plus faibles. Inversement, la seconde moitié, qui comprend des combinaisons associées à des erreurs relatives plus élevées, est principalement caractérisée par des fréquences d'images inférieures à 15 images par seconde. Ces erreurs sont plus prononcées sur la deuxième trajectoire.

Sur la base de ces observations, nous pouvons proposer une stratégie de régulation de la fréquence d'images qui s'aligne sur le mouvement du robot. Plus précisément, lorsque le robot se déplace en ligne droite, une fréquence d'images plus faible peut être suffisante, tandis qu'une fréquence d'images plus élevée serait préférable pendant les manœuvres de braquage.

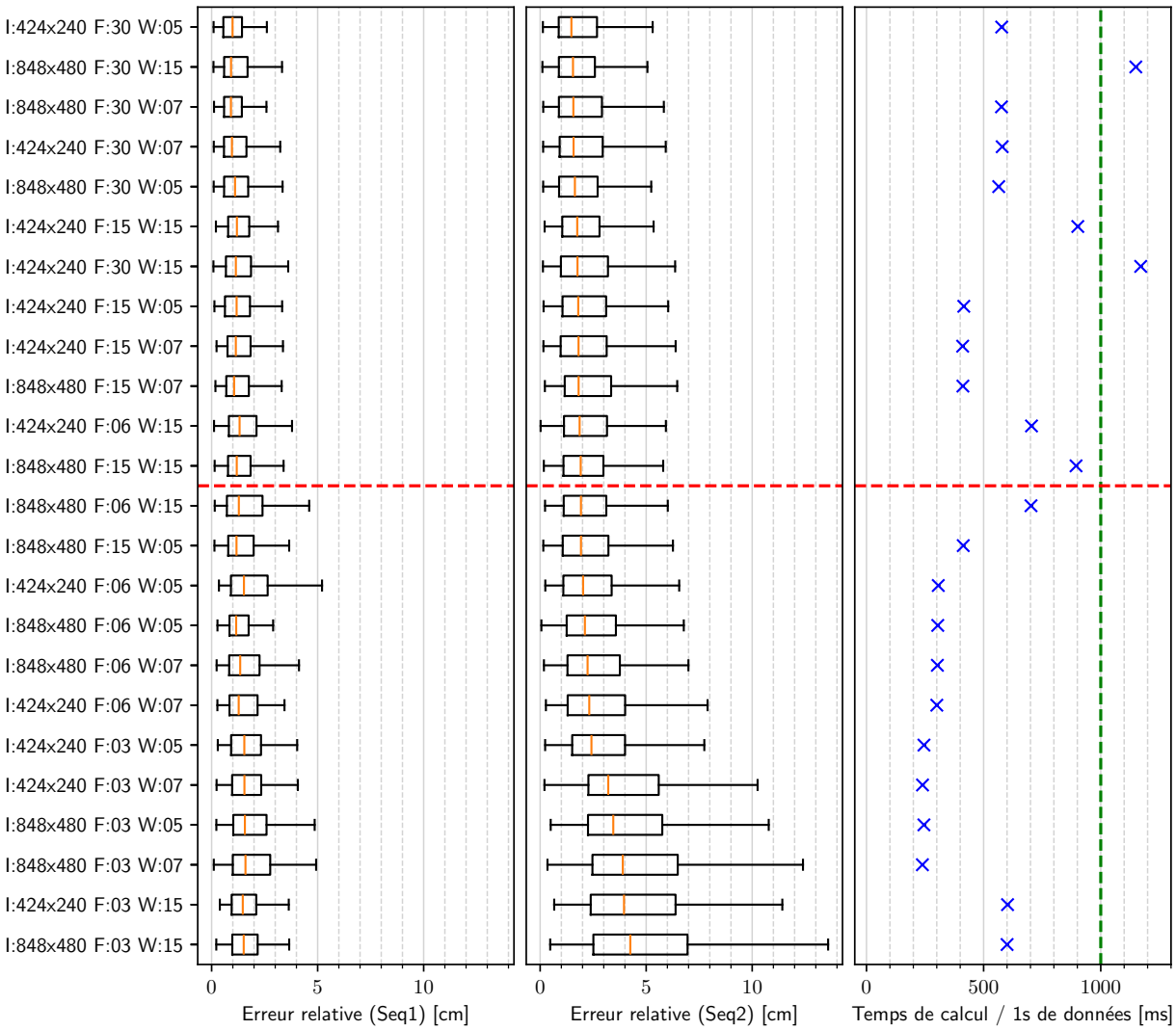


Figure 4.31 – Graphe en boîtes à moustaches pour les séquences d'*erreurs relatives*, et graphe du *temps de calcul normalisé (le temps de calcul par image × la fréquence d'images)* pour chaque combinaison de paramètres testés (I : *taille de l'image* , F : *fréquence d'images*, W : *taille de la fenêtre d'optimisation*).

Le temps d'exécution moyen par image a été mesuré pour chaque combinaison des paramètres testés. Les temps mesurés (en millisecondes) dans le tableau 4.1 donnent un aperçu du temps d'exécution moyen par image. Il est surprenant de constater que la réduction

de la taille de l'image n'a pas eu d'impact significatif sur le temps d'exécution. Cependant, l'augmentation de la taille de la fenêtre d'optimisation a entraîné des temps d'exécution plus longs en raison de l'augmentation de la charge de travail à chaque étape d'optimisation.

Table 4.1 – *Temps moyen de traitement par image* (en millisecondes) par rapport aux paramètres testés.

Fenêtre	**Fréq. d'images** / **Image**	3	6	15	30
5	848 × 480	81.7	50.6	27.5	18.8
	424 × 240	81.8	51.1	27.7	19.2
7	848 × 480	79.6	50.4	27.4	19.2
	424 × 240	79.8	50.0	27.4	19.3
15	848 × 480	200.0	117.0	59.6	38.3
	424 × 240	200.7	117.4	60.1	39.0

Il est intéressant de noter que la diminution de la fréquence des images a été associée à une augmentation du temps d'exécution par image. Cette observation contre-intuitive peut s'expliquer par le fait que le système crée des images clés (qui sont utilisées dans le processus d'optimisation) plus fréquemment lorsque la fréquence d'images est faible. Pour étayer cette observation, le tableau 4.2 présente le rapport entre le nombre des images clés et le nombre de toutes les images d'entrée, cet indicateur montre qu'une portion plus grande d'images est prise comme *images clés* lorsque la fréquence d'images d'entrées diminue, allant d'environs 60% à 3fps jusqu'à 11% à 30fps. Le tableau 4.2 montre par ailleurs qu'il n'y a aucune corrélation entre le rapport *images clés/images* et la taille de la fenêtre ou la taille des images.

Table 4.2 – *Rapport moyen entre le nombre d'images clés et le nombre d'images traitées* evalué par rapport aux paramètres testés.

Fenêtre	**Fréq. d'images** / **Image**	3	6	15	30
5	848 × 480	0.583	0.348	0.175	0.108
	424 × 240	0.587	0.347	0.177	0.108
7	848 × 480	0.580	0.344	0.176	0.108
	424 × 240	0.584	0.348	0.175	0.109
15	848 × 480	0.616	0.358	0.179	0.112
	424 × 240	0.620	0.360	0.179	0.114

Afin de déterminer les paramètres appropriés pour une exécution en temps réel, nous calculons le *facteur de vitesse*, qui compare le temps d'échantillonnage des images au temps d'exécution de la Ceiling-DSO.

Le facteur de vitesse, noté κ, est défini comme $\kappa = \frac{1}{t_F \cdot f}$, où t_F représente le temps d'exécution par image en secondes et f la fréquence d'images en hertz (les deux valeurs peuvent être obtenues à partir du tableau 4.1).

Un facteur de vitesse supérieur à un ($\kappa > 1$) indique que l'algorithme peut traiter les images plus rapidement que la fréquence d'images, tandis qu'un facteur de vitesse inférieur à un ($\kappa < 1$) implique que l'algorithme ne peut pas traiter les images à la vitesse de la fréquence d'images, ne respectant donc pas les contraintes pour un traitement en temps réel.

La figure 4.32 illustre le facteur de vitesse pour diverses combinaisons des paramètres testés. Comme indiqué dans le tableau 4.1, il n'y a pas de corrélation apparente entre la taille de l'image et le temps d'exécution. Nous avons alors marginalisé la taille de l'image dans le diagramme à barres de la figure. Les résultats démontrent une performance temporelle acceptable (dans les conditions de l'expérience), sauf dans le cas d'une fréquence d'images élevée de 30fps combinée à une taille de fenêtre maximale de 15. D'après nos tests, une taille de fenêtre de 7 semble offrir un bon compromis entre la précision et le temps d'exécution.

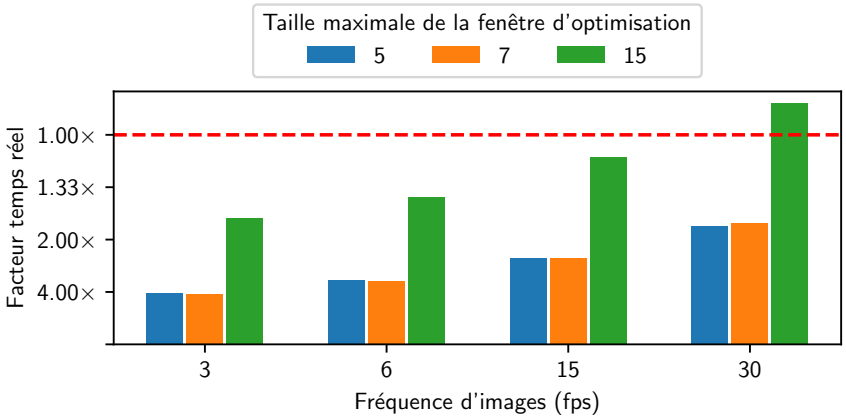


Figure 4.32 – *Facteur de vitesse* moyen sur les deux séquences. La ligne rouge en pointillés délimite la limite pour un traitement en temps réel.

4.10 Conclusion

Dans ce chapitre, nous avons présenté notre stratégie de localisation en environnements industriels dynamiques à base d'une caméra verticale. Nous avons montré que la séparation de l'espace de navigation à l'horizontal de l'espace de localisation à la verticale simplifie d'une façon significative le système conçu et nous épargne les problèmes de localisation liés à la présence d'objets mobiles dans la scène.

La méthode proposée se base sur la formulation de la *Direct Sparse Odometry (DSO)* qui, étant une approche directe, effectue les calculs directement sur les valeurs d'intensités des images brutes afin minimiser l'erreur photométrique. Ceci permet d'un côté de simplifier la conception du système (pas d'extraction de caractéristiques) et permet d'un autre côté de s'adapter aux environnements à faibles contrastes et à faibles textures.

L'un des principaux avantages de notre système est qu'il est adapté aux environnements dynamiques que l'on trouve couramment dans les milieux industriels. Nous avons montré que, contrairement à d'autres approches, notre système ne repose pas sur des hypothèses concernant la forme ou le contenu spécifique du plafond, ce qui en fait une solution polyvalente.

Pour valider notre approche, nous avons effectué une comparaison entre les trajectoires estimées avec celle-ci et une vérité terrain calculée à base des données du LiDAR. En outre, nous avons mené des expériences pour analyser l'impact de la variation des paramètres de la DSO (à savoir, la *taille de l'image* d'entrée, de la *fréquence des images* d'entrée et de la *taille de la fenêtre d'optimisation*) sur la précision et les performances temps réel du système, dans le but d'identifier une combinaison optimale des paramètres.

Les résultats expérimentaux indiquent que la modification de la taille de l'image d'entrée n'a pas eu d'impact signifiant sur les performances du système. Cependant, la modification de la fréquence d'images d'entrée a eu un léger effet sur la trajectoire estimée et le temps d'exécution.

D'après nos tests, une fréquence d'images de 15 images par seconde offre un bon équilibre entre la précision et l'efficacité en temps d'exécution. Et en ce qui concerne la taille de la fenêtre d'optimisation, sa variation n'a pas eu d'effet significatif sur la précision de la trajectoire, mais a eu un impact notable sur le temps d'exécution. Nos expériences ont démontré que l'utilisation d'une taille de fenêtre maximale de 7 permettait d'obtenir des résultats satisfaisants en termes de précision et de temps de calcul.

Ce travail sert de base aux futurs travaux qui se concentreront sur le développement d'une approche d'estimation de l'échelle métrique basée sur la fusion multicapteur. En outre, nous souhaitons proposer prochainement des stratégies de gestion des cartes et de fermeture des boucles afin d'offrir une solution SLAM complète pour les applications de vision au plafond. En plus, nous prévoyons de mettre notre jeu de données à la disposition de la communauté scientifique, dans le but de permettre à d'autres chercheurs d'évaluer et de valider leurs algorithmes de vision au plafond dans des scénarios réels.

Les travaux présentés dans ce chapitre ont été valorisés par une conférence internationale avec acte et comité de lecture, et ont été soumises à un journal international spécialisé.

- [114] Abdelhak Bougouffa, Emmanuel Seignez, Samir Bouaziz, and Florian Gardes. *« Evaluation of a Novel DSO-Based Indoor Ceiling-Vision Odometry System. »* In 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), 47–53, 2022. DOI : 10.1109/ICARCV57592.2022.10004272.
- [142] Abdelhak Bougouffa, Emmanuel Seignez, Samir Bouaziz, and Florian Gardes. *« Indoor Ceiling-Vision Odometry based on Direct Sparse Odometry. »* , Preprint submitted to an international journal.

CHAPITRE 5

Jeux de données multicapteurs pour la vision verticale

5.1 Introduction

Pour pouvoir valider un système de localisation destiné aux environnements industriels, nous avons besoin d'un jeu de données représentatif de ce type d'environnements. L'usage de la vision verticale pour la localisation indoor requiert des algorithmes prenant en compte la nature et la structure des plafonds à partir desquels on pourrait extraire de l'information utile pour se localiser. Nous avons présenté dans le chapitre 4 quelques exemples de la littérature scientifique de systèmes de localisation à base de vision verticale, ces systèmes supposent généralement des hypothèses sur la forme du plafond ainsi que sur les motifs observables sur ce dernier. La plupart de ces systèmes ont été conçus pour des environnements de travail de type bureaux, espaces ouverts (*open spaces*), ou garages.

Il existe de nombreux jeux de données pour évaluer des algorithmes de la localisation par vision, tels que le jeu de données de TUM monoVO [120], le KITTI Vision Benchmark Suite [143] et le nuScenes [144]. Néanmoins, nul de ces jeux de données n'inclut des images acquises à partir d'une caméra verticale, ni en environnements industriels, ni en d'autres environnements. La plupart des jeux de données ne comportent que les images acquises d'une (ou plusieurs) caméras orientées vers l'avant, l'arrière ou vers les côtés d'un véhicule/robot, ou des caméras portées en mains libres lors de l'acquisition.

Afin de valider, dans des conditions réelles, notre approche de localisation à base de vision verticale (présentée dans le chapitre 4) et en vue de tester d'éventuelles stratégies de fusion multicapteur, nous avons réalisé un état de l'art des jeux de données visuels existants. Ce travail nous a permis d'identifier l'absence d'un jeu de données préétabli pour un tel objectif, ce qui nous a conduits par la suite à concevoir et à réaliser une expérimentation dans le but d'acquérir un jeu de données multicapteur centré sur la vision verticale.

Ce jeu de données devrait permettre la validation de notre approche sur des environnements industriels représentatifs. Ceci devrait aussi permettre la comparaison des différentes méthodes de localisation à base de vision verticale les unes par rapport aux autres, ainsi que les comparer avec des méthodes de vision classiques (avec caméras frontales), des méthodes à base de LiDAR ou des méthodes à base de fusion multicapteur.

5.2 Travaux similaires

Il existe plusieurs jeux de données visuelles qui ont été proposés dans la littérature scientifique. Nous nous intéressons ici aux jeux contenant des données visuelles (images) et qui sont destinés à des applications en localisation robotique indoor. En effet, dans notre contexte, nous devons prendre en considération le cas d'utilisation ciblé. Ainsi, après une synthèse des jeux de données existants, nous pouvons distinguer quatre types principaux d'applications cibles :

- La reconstruction de scènes 3D ou la structure à partir de mouvements *(SfM - Structure from Motion)* [145, 146] ;
- La reconnaissance visuelle des lieux *(VPR - Visual Place Recognition)* [147] ;
- La localisation basée sur l'image *(IBL - Image-based Localization)* [147] ;
- L'estimation de trajectoires (Odométrie visuelle, vSLAM, *etc.*) [148-150].

Pour des algorithmes de structure à partir de mouvements (SfM), le but est de reconstruire une représentation 3D de l'environnement à partir d'un ensemble d'images prises d'angles différents. Les poses des caméras seront aussi produites pendant ce processus, en revanche, celles-ci ne représentent pas forcément une trajectoire réelle. Les images d'un jeu de données destiné aux applications en SfM ne sont pas forcément ordonnées spatialement, *c.-à-d.*, elles ne sont pas acquises d'une manière séquentielle suivant une trajectoire continue. Parmi les jeux de données en environnements indoor destinés à des applications en SfM, nous citons le ETH3D Stereo [145] et le DTU [146].

Dans les applications en localisation basée sur l'image (IBL), les données sont présentées sous forme d'un ensemble d'images avec une représentation 3D préétablie de l'environnement. Le but des systèmes IBL est de se localiser dans l'environnement en utilisant une image à la fois et en ayant une représentation tridimensionnelle préalable de la structure de l'environnement. Baidu-IBL [147] est un exemple de jeux de données, avec des scènes en indoor, destiné à des applications en IBL.

Les applications en reconnaissance visuelle des lieux (VPR) ont pour but de permettre le stockage et l'indexation d'un ensemble d'images dans une base de données adéquate. Ensuite, en prenant une image en entrée, l'algorithme de VPR permet de chercher des images similaires dans cette base de données. Dans le contexte des systèmes vSLAM, la VPR est communément utilisée pour détecter la fermeture de boucle *(Loop closure)* [64, 67, 68]. Ainsi, l'algorithme du SLAM indexe des images tout au long de sa trajectoire et il cherche à chaque fois à détecter si l'endroit courant a été déjà visité (présent dans la base de données). L'approche dominante pour résoudre ce problème en vSLAM est connue sous le nom du *sac de mots binaires/visuels (BBoW - Bag of Binary/Visual Words)* [151].

Les jeux de données destinés exclusivement à des applications en SfM, VPR ou IBL ne sont pas adaptés à des applications en odométrie visuelle ou en vSLAM, ils sont donc écartés de notre étude.

Plusieurs des jeux de données disponibles sont destinés à des applications sur des véhicules autonomes. Ces jeux sont acquis en environnements urbains (outdoor), avec la présence, dans la plupart des cas, de modalités spécifiques aux environnements outdoors tel que le GPS. Un exemple d'un tel jeu de données est le fameux KITTI [143] proposé en 2012, qui est devenu le standard de facto pour l'évaluation des systèmes destinés aux environnements urbains (odométrie/SLAM à base de vision monoculaire, stéréoscopique ou de LiDAR).

Le jeu de données nuScenes [144], présenté en 2020, vise à couvrir une combinaison plus large de capteurs utilisés dans les véhicules autonomes (voir la figure 5.1), des séquences plus longues et plus d'annotations pour les applications en détection et classification d'objets sur la scène.

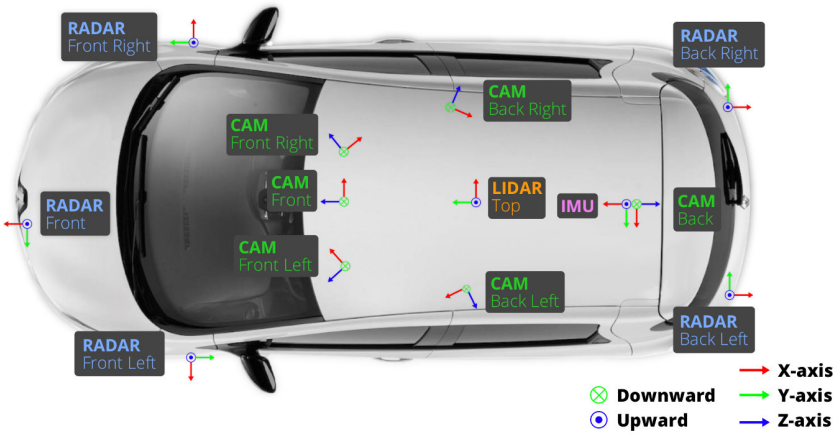


Figure 5.1 – La configuration des capteurs employés dans le jeu de données nuScenes.

D'autres jeux de données similaires sont disponibles dont Freiburg AS [152], NewCollege [153], CityScapes [154], et ApolloScape [155]. En revanche, dans ce chapitre, nous allons concentrer sur les jeux de données destinés à des applications en localisation indoor. Par conséquent, nous n'allons pas prendre en compte les jeux de données qui sont exclusivement destinés à des environnements outdoor.

5.2.1 Jeux de données pour VO et vSLAM en indoor

Pour se positionner par rapport aux travaux précédents, nous nous intéressons aux jeux de données qui sont destinés à des applications en odométrie visuelle ou vSLAM, et qui sont acquis dans des environnements indoor. Il existe plusieurs jeux de données qui répondent à ce critère, parmi eux, nous avons sélectionné : Rawseeds [148], TUM RGB-D [156], TUM monoVO [120], FusionPortable [149] et Hilti-Oxford [150].

5.2.1.1 Rawseeds

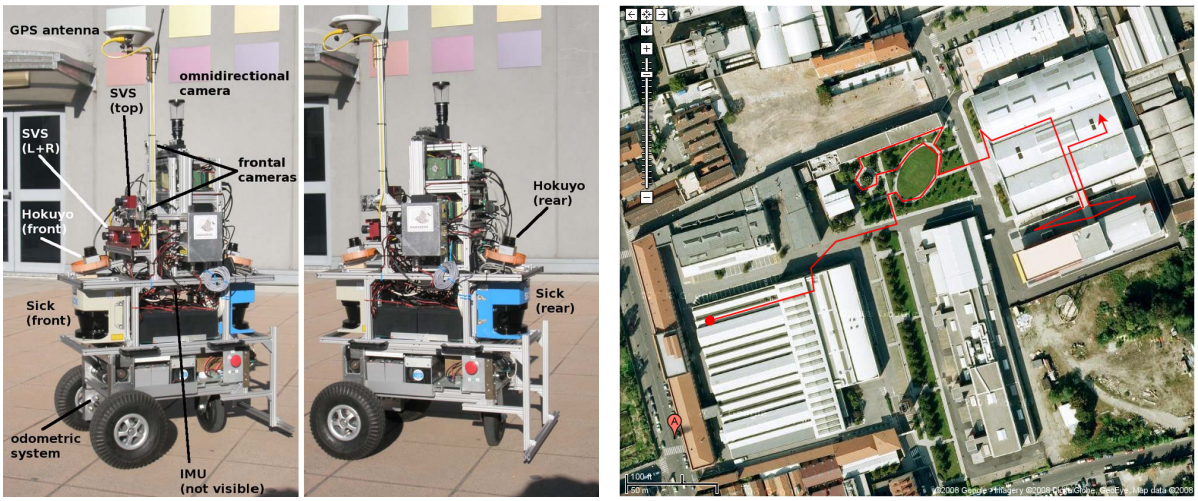
Rawseeds [148] est un projet porté par l'École polytechnique de Milan et financé par la Commission européenne dans le cadre du programme *« Technologies de la société de l'information »* (FP6-045144). L'objectif du projet est de stimuler et de soutenir les progrès de la robotique autonome en fournissant une boîte à outils complète de benchmarking de haute qualité [157].

Le but du projet est de définir un ensemble de référentiels et de méthodologies de haute qualité pour l'évaluation des algorithmes en robotique autonome. Le jeu de données est axé sur les problématiques d'analyse de données sensorielles et de fusion multicapteur appliquées à la localisation, à la cartographie et au SLAM.

Ce jeu de données offre 5 séquences en environnement indoor et 6 séquences en environnement mixtes (outdoor, indoor et outdoor). Ces séquences ont été enregistrées respectivement sur les campus de l'Université de Milan-Bicocca et de l'École polytechnique de Milan. La figure 5.2b illustre la trajectoire d'une séquence mixte du jeu de données.

La figure 5.2a montre l'ensemble des capteurs utilisés, montés sur le robot Robocom. Le jeu de données offre multiple types de données, incluent :

- Des images (3 caméras orientées vers l'avant + une caméra omnidirectionnelle).
- Des données inertielles (IMU 6-DoF).
- Des données des scrutateurs LiDAR.
- Des données ultrasoniques (ceinture de 12 capteurs à ultrasons).



(a) **(b)**

Figure 5.2 – (a) : La plateforme Robocom avec les capteurs utilisés dans Rawseeds ; (b) : La trajectoire d'une séquence mixte (indoor et outdoor) (tirés de [148]).

Ce jeu de données offre une vérité terrain à base de deux systèmes qui sont indépendants des capteurs installés sur le robot, désignés *GTvision* et *GTlaser*. Le *GTvision* utilise les données d'un réseau externe de caméras fixes et calibrées ; la pose du robot est reconstituée en se basant sur l'observation, par le réseau de caméras, d'un ensemble de marqueurs visuels

placés sur le robot. Le système *GTlaser* utilise les données d'un réseau de scanners laser fixes ; dans ce cas, la pose du robot est reconstituée en se basant sur la détection d'une coque rectangulaire réfléchissante fixée autour du robot [148].

5.2.1.2 TUM RGB-D

Le jeu de données TUM RGB-D [156] a été développé en 2012 par le groupe de vision par ordinateur de l'Université technique de Munich (TUM). Il vise à évaluer les systèmes SLAM à base de RGB-D, *c.-à-d.*, des images RGB avec des images de profondeurs.

Les images de ce jeu de données ont été capturées par une caméra du type Microsoft Kinect équipée d'un capteur RGB-D et d'un IMU (*fig.* 5.3a).

Le TUM RGB-D est composé de 39 séquences enregistrées dans un environnement de bureau (*fig.* 5.3d) et dans un hall industriel (*fig.* 5.3c). La plupart des séquences ont été enregistrées à partir d'une Kinect *portée à main* avec des mouvements 6-DoF non contraints. Seulement une sous partie des séquences a été enregistrée à partir d'une Kinect montée sur un robot différentiel de type Pioneer 3 (*fig.* 5.3b) piloté manuellement dans un environnement intérieur [156].

La vérité terrain fournit avec TUM RGB-D vient d'un système de *capture de mouvement (MoCap)*, qui suit les positions des marqueurs réfléchissants placés sur la Kinect (*fig.* 5.3a).

(a) Kinect avec marqueurs réfléchissants **(b)** Le robot utilisé **(c)** Environnement industriel **(d)** Environnement de bureaux

Figure 5.3 – Le jeu de données TUM RGB-D [156].

5.2.1.3 TUM monoVO

Ce jeu de données, comme son nom l'indique, est destiné à évaluer des systèmes d'odométrie visuelle à base de caméra monoculaire. Il est constitué de 50 séquences acquises par deux caméras portées à main ; une caméra d'un champ de vision étroit de $98° \times 79°$ (première ligne de la figure 5.4) et une autre d'un large champ de vision de $148° \times 122°$ (deuxième ligne de la figure 5.4) [120].

Figure 5.4 – Exemples d'images du jeu de données TUM monoVO. Sur la première ligne les images de la caméra à champ de vision étroit ; sur la deuxième ligne les images de la caméra de large champ de vision [120].

Les séquences contiennent principalement des mouvements d'exploration de la scène, commençant et se terminant à la même position. Cela permet d'évaluer la précision du suivi via la dérive accumulée du début à la fin, sans exiger la présence d'une vérité terrain pour toute la trajectoire [120].

Ce jeu de données a la particularité d'être accompagné des données de calibrage photométrique. Ainsi, chaque image est accompagnée de son temps d'exposition tel que rapporté par le capteur. Le jeu de données inclus aussi le calibrage de la fonction de réponse du capteur ainsi que les facteurs denses (masque de la même taille que l'image) d'atténuation de l'objectif (appelée aussi vignettage). Pour plus d'informations, voir la section 4.8.1 du chapitre 4.

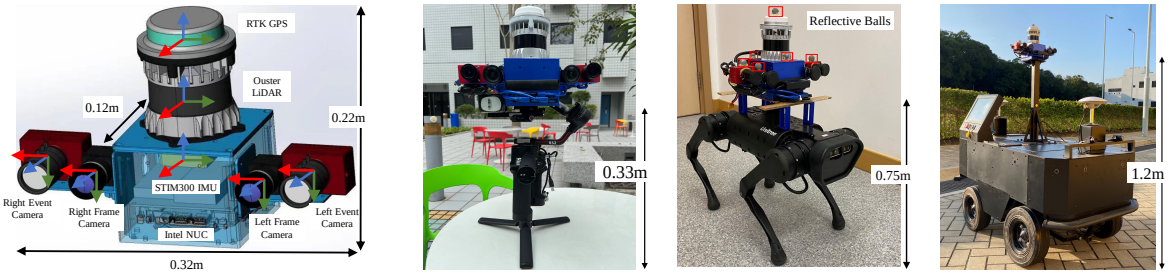
5.2.1.4 FusionPortable

FusionPortable est un jeu de données multicapteur destiné à des applications en SLAM. Les séquences de ce jeu de données ont été réalisées dans le campus de l'Université des sciences et technologies d'Hong Kong, principalement en milieux indoor [149].

Ce jeu contient des données en provenance de deux caméras, deux caméras d'événements, un IMU, un GPS-RTK et un LiDAR 3D. Les capteurs sont placés sur une monture (figure 5.5a) qui a été utilisée lors de la collecte des données.

Sur la plupart des séquences enregistrées en indoor, la monture a été placée sur un cardan stabilisateur *(Gimbal stabilizer)* porté à main lors de l'acquisition (figure 5.5b). Sur le reste des séquences en indoor, la monture des capteurs a été placée sur un robot quadrupède (figure 5.5c). Et dans la séquence enregistrée en environnement outdoor, la monture a été placée sur un robot mobile terrestre (figure 5.5d).

Selon les séquences, trois types de réalité terrain ont été utilisées dans FusionPortable. La séquence enregistrée en environnement extérieur (outdoor) utilise une référence à base de d'un GPS-RTK. Tandis que les séquences indoor utilisent soit un système de *capture de mouvement (MoCap - Motion Capture)* soit une mise en correspondances des données LiDAR 3D à base de la méthode NDT à 6 degrés de libertés (NDT 6DoF).



(a) **(b)** **(c)** **(d)**

Figure 5.5 – Plateformes et capteurs utilisés pour l'acquisition des données de FusionPortable (tirés de [149]).

5.2.1.5 Hilti-Oxford

Hilti-Oxford est un jeu de données destiné à l'évaluation des algorithmes SLAM. Il contient des données multimodales, acquises par une plateforme de collecte de données (figure 5.6a) comprenant un LiDAR 3D, cinq caméras (dont une orientée vers le haut) et une unité de mesures inertielles (IMU) [150]. L'équipe Hilti et le groupe robotique et perception (RPG) de l'université de Zürich ont lancé le Hilti Challange à base de ce jeu de données afin de permettre aux acteurs (académiques ou industriels) d'évaluer les performances de leurs systèmes SLAM.

(a) **(b)** **(c)** **(d)**

Figure 5.6 – (a) : Capteurs utilisés dans Hilti-Oxford. (b-d) : exemples d'environnements du jeu de données (tirés de [150]).

Les séquences de Hilti-Oxford sont fournies avec une vérité terrain d'une précision millimétrique en utilisant un scanner topographique ultra-précis de type *Z+F Imager 5016*. Les nuages de points de référence sont acquis par le scanner, puis, ils sont utilisés pour l'enregistrement de ceux collectés par le robot pendant la réalisation de sa trajectoire [150].

Ce jeu de données est le seul contenant une caméra orientée vers le plafond. Néanmoins, sur les séquences disponibles, il n'y a qu'une seule séquence acquise dans un sous-sol qui peut ressembler à un environnement industriel de type entrepôt de petite taille.

5.2.1.6 Synthèse et discussion

Nous considérons que les jeux de données présentés ici ne sont pas adaptés à notre application. D'un côté, les séquences sont enregistrées, dans la plupart des cas, dans des environnements non assimilables à des environnements industriels (de type : bureaux, campus d'université, *etc.*). D'un autre côté, la plupart de ces jeux de données ne contiennent pas de caméra verticale, les deux exceptions étaient Rawseeds et Hilti-Oxford.

Rawseeds offre des images issues d'une caméra omnidirectionnelle qui couvre partiellement le plafond. Néanmoins, les images de cette caméra sont très distordues et elles ne sont pas calibrées photométriquement. Ainsi, elles ne sont pas utilisables avec notre approche (voir la section 4.8 du chapitre 4).

Hilti-Oxford est très récent et il n'était pas disponible lors de la construction de notre jeu de données. Hilti-Oxford offre des images d'une caméra verticale, ce qui peut permettre l'évaluation de notre approche. Néanmoins, seulement une séquence peut être utilisée dans le cadre de notre travail, à savoir, la séquence enregistrée sur le robot dans un sous-sol qui ressemble à un environnement de type entrepôt de petite taille.

En conclusion, le besoin d'un jeu de données centré sur la vision verticale reste toujours d'actualité. En plus, nous cherchons un jeu de données acquis sur une plateforme qui reproduit les mêmes dynamiques de mouvement que nos plateformes ciblées. Ainsi, les jeux de données acquis à partir de caméras portées à main libre ne sont pas pertinents dans notre cas. Par conséquent, nous avons choisi de concevoir et réaliser un jeu de données qui permettra l'évaluation des approches d'odométrie visuelle ou de vSLAM à base de vision au plafond et/ou de vision classique.

Le tableau 5.1 positionne notre travail par rapport aux autres travaux en présentant une comparaison entre les jeux de données présentés précédemment et le nôtre. À notre que le symbole $(*)$ dans le tableau signifie la vue du plafond est couverte partiellement par une caméra omnidirectionnelle (distorsions majeures) et le $(**)$ dans le cas du Hilti-Oxford signifie que seulement deux séquences assimilables à un environnement industriel, dont une acquise sur un robot.

À noter aussi que la ligne *Plateforme* du tableau désigne le type de la plateforme utilisée pour l'acquisition des données, avec la codification suivante :

- (M) Capteurs portés à main *(handheld)* ;
- (D) Robot à entrainement différentiel *(Differential Drive)* ;
- (Q) Robot quadrupède.

5.3 Plateforme robotique et capteurs

Afin de récolter les différentes séquences de notre jeu de données, nous avons exploité la plateforme *ez-Wheel SWD® Starter Kit*, présentée auparavant dans le chapitre 4. Nous avons instrumenté cette plateforme en intégrant plusieurs capteurs, notamment :

- Une caméra Intel® RealSense™ D455 orientée vers le plafond ;
- Une caméra Intel® RealSense™ D435i orientée vers l'avant ;
- Un scrutateur laser (LiDAR) de sécurité, d'une nappe bidimensionnelle (2D) de type IDEC SE2L ;

Table 5.1 – Table comparative des jeux de données contenant des scènes en environnements indoor.

Jeu de données	Rawseeds	TUM RGB-D	TUM monoVO	Fusion-Portable	Hilti-Oxford	**Le nôtre**
Env. industriel		×			**	×
Plateforme	M/D	M/D	M	M/Q	M/D/Q	D
Caméra frontale	×	×	×	×	×	×
Stéréo frontale	×	×		×	×	×
Caméra verticale	*				×	×
Stéréo verticale						×
Vérité de terrain	×	×		×	×	×
IMU	×			×	×	×
LiDAR 2D	×			×		×
LiDAR 3D				×	×	
Odométrie	×					×

- Des codeurs incrémentaux à effet Hall (intégrés dans la roue *ez-Wheel SWD® Core*) ;
- Deux capteurs de flux optique PMW3901, un de chaque côté du robot, orientés vers le sol.

La figure 5.7 montre une photo annotée de notre plateforme expérimentale *SWD® Starter Kit*, en mettant en évidence l'ensemble des capteurs montés sur celle-ci. La figure 5.8 illustre la structure de la plateforme, les emplacements des capteurs et les dimensions associées.

5.3.1 Caméras Intel® RealSense™

Nous avons choisi d'utiliser des caméras Intel® RealSense™. Ces dernières sont concues spécifiquement pour des applications en vision par ordinateur.

Les modèles utilisés, à savoir le D435i et le D455 (*fig.* 5.9), sont très similaires avec quelques différences mineurs entre les deux. Les deux caméras sont dotées chacune d'un capteur RGB, de deux capteurs proches infrarouges pour la vision stéréoscopique, d'un projecteur infrarouge pour l'augmentation des scènes de faibles textures, et d'une unité de mesures inertielles *(IMU - Inertial Measurement Unit)* de 6DoF (accéléromètre et gyromètre en 3D).

Les caractéristiques techniques des deux caméras sont décrites dans les tableaux suivants. Les spécifications techniques des capteurs proches infrarouges, couleurs et des IMUs sont décrites dans les tableaux 5.2, 5.3 et 5.4, respectivement. Tandis que le tableau 5.5 décrit les caractéristiques des images de profondeurs des deux modèles.

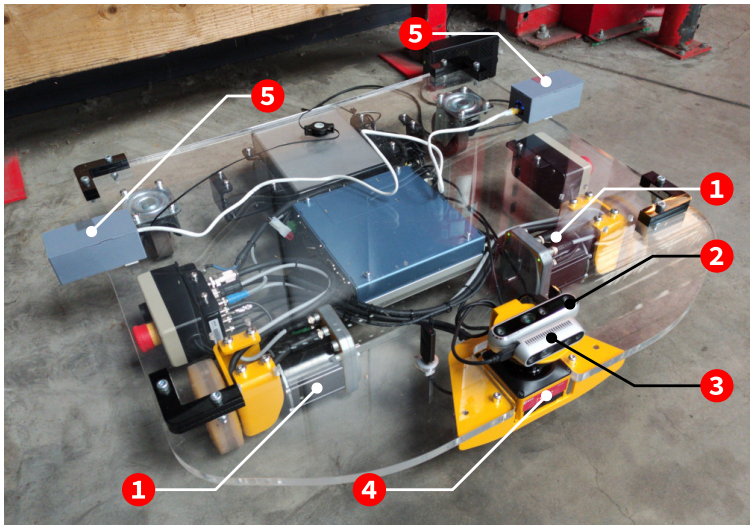


Figure 5.7 – La plateforme expérimentale SWD® Starter Kit avec l'ensemble des capteurs montés sur celle-ci. (1) Deux roues SWD® Core avec des codeurs incrémentaux intégrés ; (2) Caméra RealSense™ D455 orientée vers le plafond ; (3) Caméra RealSense™ D435i orientée vers l'avant ; (4) Le LiDAR de sécurité IDEC SE2L ; (5) Les deux capteurs de flux optique PMW3901 orientés vers le sol.

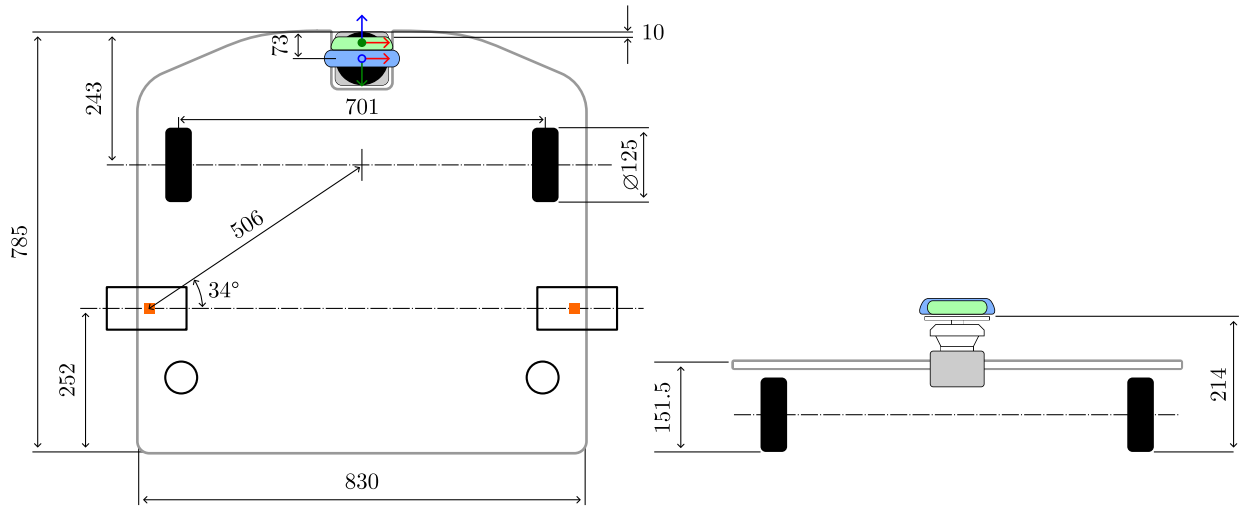


Figure 5.8 – Schéma de la plateforme expérimentale SWD Starter Kit avec les dimensions (en millimètres) et les emplacements des capteurs. En bleu, la caméra Intel® RealSense™ D455 orientée vers le plafond, et en vert, la caméra Intel® RealSense™ D435i orientée vers l'avant, les deux sont montés au dessus du LiDAR IDEC SE2L. En orange, les capteurs de flux optique PMW3901.

5.3.2 Capteurs de flux optique PMW3901

Les capteurs de flux optique *(Optical Flow Sensors)* sont des capteurs de vision utilisés généralement dans les souris optiques. Ils permettent de mesurer des déplacements linéaires sur un plan bidimensionnel (2D). Ces capteurs ne coûtent en moyenne que quelques euros, d'où l'intérêt de les intégrer sur le robot dans le but de les qualifier et d'étudier leurs fiabilités pour une application à l'estimation incrémentale et la prédiction des mouvements.

Figure 5.9 – Les caméras Intel® RealSense™ utilisées, à gacuhe la D435i et à droite la D455.

Table 5.2 – Les caractéristiques des caméras gauche et droite (proches infrarouges).

Paramètre	**D435i**	**D455**
Capteur d'images	OV9282	OV9782
Pixels actifs	1280×800	1280×800
Rapport d'aspect du capteur	8 : 5	8 : 5
Format d'acquisition	10-bit RAW	10-bit RAW
Ouverture *(f-number)*	f/2.0	f/2.0
Distance focale [mm]	1.93	1.93
Type de filtre	N/A	N/A
Mise au point *(Focus)*	Fixe	Fixe
Type d'obturateur *(Shutter)*	Global	Global
Champ de vision horizontal [deg]	91.2	90 ± 1
Champ de vision vertical [deg]	65.5	65 ± 1
Champ de vision diagonal [deg]	100.6	95 ± 1
Distorsion	$\leq 1.5\%$	$\leq 1.5\%$

Table 5.3 – Les caractéristiques des caméras utilsées (capteurs RGB).

Paramètre	**D435i**	**D455**
Capteur d'images	OV2740	OV9782
Pixels actifs	1920×1080	1280×800
Rapport d'aspect du capteur	16 : 9	16 : 10
Format d'acquisition	10-bit RAW RGB	10-bit RAW RGB
Ouverture *(f-number)*	f/2.0	f/2.0
Distance focale [mm]	1.88	1.93
Type de filtre	IR Cut Filter	IR Cut Filter
Mise au point *(Focus)*	Fixe	Fixe
Type d'obturateur *(Shutter)*	Roulant *(Rolling)*	Global
Champ de vision horizontal [deg]	69.4	90
Champ de vision vertical [deg]	42.5	65
Champ de vision diagonal [deg]	77	98
Distorsion	$\leq 1.5\%$	$\leq 1.5\%$

Table 5.4 – Les caractéristiques des IMUs intégrées dans les deux caméras.

Paramètre	**D435i**	**D455**
Capteur	BMI055	BMI085
Degrés de liberté	6	6
Porté de l'accéléromètre [g]	± 4 (± 5%)	± 4 (± 5%)
Porté du gyromètre [deg/s]	± 1000 (± 0.3%)	± 1000 (± 0.3%)
Fréquence d'échantillonnage (accéléro) [Hz]	62.5, 250	100, 200
Fréquence d'échantillonnage (gyro) [Hz]	200, 400	200, 400

Table 5.5 – Les caractéristiques des images de profondeurs des deux caméras utilisées.

Paramètre	**D435i**	**D455**
Entraxe (baseline) [mm]	50	95
FoV stéréo HD [deg] (± 3)	H : 87 / V : 58 / D : 95	H : 87 / V : 58 / D : 95
FoV stéréo VGA [deg] (± 3)	H : 75 / V : 62 / D : 89	H : 75 / V : 62 / D : 89
FoV projecteur IR [deg]	H : 90 / V : 63 / D : 99	H : 90 / V : 63 / D : 99
FoV caméra RGB	H : 69 / V : 42 / D : 77	H : 90 / V : 65 / D : 98

Un capteur de flux optique est constitué d'une caméra de basse résolution (moins de 100px × 100px) avec un circuit intégré qui implémente un algorithme d'estimation de flux optique à partir des images acquis. Pour des capteurs commerciaux de ce type, les détails des algorithmes utilisés en interne sont souvent tenus secrets. Néanmoins, d'une manière générale, ces algorithmes sont basés sur des méthodes classiques de détection de flux optique telles que la méthode Lucas-Kanade [158].

5.3.2.1 Système d'odométrie à base de capteurs de flux optique

Dans cette section, nous proposons un système d'estimation prédictive des déplacements à l'aide des capteurs de flux optique. Les données issues de ces capteurs peuvent être utilisées pour implémenter une odométrie basée sur les mesures du flux optique et d'étudier sa pertinence par rapport à l'odométrie des roues.

La plupart des capteurs de flux optique disponibles dans le commerce fonctionnent à des distances maximales du sol de 2mm à 50mm. Ces contraintes sont acceptables pour des applications sur des souris optiques. Cependant, pour un robot destiné à évoluer dans des environnements industriels, un capteur placé à quelques millimètres du sol peut être facilement endommagé. Par conséquent, nous avons opté pour le capteur PMW3901 (figure 5.10b) équipé d'une optique lui permettant de fonctionner à une distance de $[80\text{mm}, \infty]$.

Un capteur de flux optique peut être considéré comme une caméra de basse résolution, un *modèle sténopé (Pinhole camera model)* peut ainsi être utilisé pour modéliser les projections à l'intérieur du capteur (figure 5.10a).

(a) Modèle sténopé appliqué au capteur **(b)** Le capteur PMW3901

Figure 5.10 – Modélisation du capteur de flux optique PMW3901.

Contrairement à une caméra, le capteur de flux optique ne fournit pas des images, mais les vecteurs dominants du flux optique (Δx_S Δy_S). Le capteur effectue en interne un calcul du flux optique (suivi des déplacements de chaque pixel) sur les images successives à une haute fréquence d'échantillonnage (de 100fps à plus de 2000fps), puis, retourne le vecteur de déplacement dominant sur toute l'image.

5.3.2.2 Conception du système

Dans un premier temps, nous avons intégré les capteurs sur le robot comme illustré sur la figure 5.11. Les capteurs PMW3901 sont pilotés via un bus SPI *(Serial Peripheral Interface)* par un microcontrôleur 32-bit ARM® Cortex™-M3 de type *mbed LPC1768*.

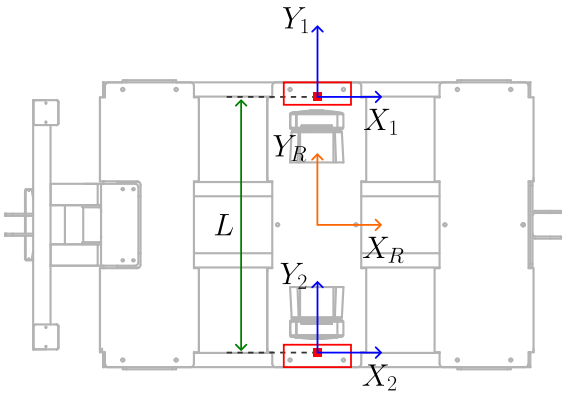


Figure 5.11 – L'emplacement des capteurs de flux optique (en rouge) sur robot.

Les deux capteurs doivent être placés sur chaque côté du robot, le capteur du côté gauche est liée directement via le bus SPI au microcontrôleur. L'autre capteur doit être placé à une distance du premier (au côté droite du robot). Or le bus SPI est destiné aux communications inter-circuits, il n'est donc pas adapté pour des communications sur des longues distances, surtout à proximité des moteurs où il y a de fortes interférences électromagnétiques générées par ces derniers.

Pour connecter le deuxième capteur, nous avons choisi d'utiliser des circuits intégrés de type LTC4332 d'Analog Devices. Ce périphérique est un prolongateur robuste point-à-point du bus SPI. Il a été conçu pour fonctionner dans les environnements industriels à bruit élevé et sur de longues distances (jusqu'à 10m pour une horloge de 2MHz).

Ainsi, nous avons placé un périphérique LTC4332 à chaque extrémité du bus SPI qui lie le capteur PMW3901 et le microcontrôleur distant (figure 5.12). Les deux périphériques LTC4332 utilisés sont équipés d'une interface RJ45, permettant d'utiliser un câble de réseau croisé pour lier les deux bouts. Le câble de réseau étant blindé et en paires torsadées, il permet d'avoir une bonne résilience aux problèmes de compatibilité électromagnétique (CEM).

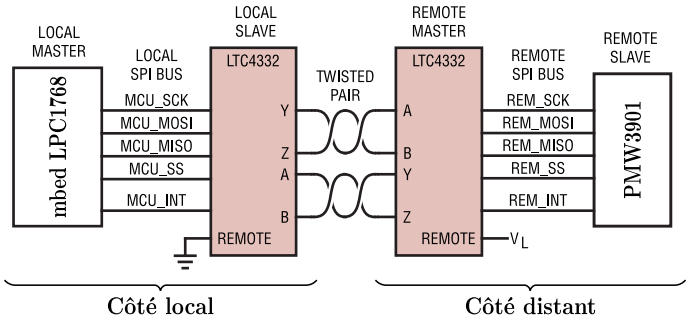


Figure 5.12 – Liaison du capteur PMW3901 distant (côté droite du robot) au microcontrôleur mbed LPC1768 via les deux périphériques LTC4332.

L'acquisition de données des capteurs est gérée par le microcontrôleur mbed LPC1768 à une fréquence d'échantillonnage de 100Hz, les données sont ensuite encodées et envoyées au PC embarqué via un port série (émulé sur USB).

5.3.2.3 Modélisation du capteur

Afin de construire un odomètre alternatif à base de capteurs de flux optique, la disposition la plus simple est de monter deux capteurs sur chaque côté du robot, sur le même axe que les roues motrices (figure 5.11).

Les valeurs lues d'un capteur (Δx_S Δy_S) peuvent être converties en déplacements en mètres (Δx_W Δy_W) en connaissant la distance capteur-sol h, la résolution du capteur r, et l'angle du champ de vision du capteur α. Ainsi, nous proposons le modèle décrit par l'équation (5.1) que nous avons dérivé géométriquement à partir de la figure 5.10a, avec w une constante interne du capteur.

$$\begin{bmatrix} \Delta x_W \\ \Delta y_W \end{bmatrix} = \beta \begin{bmatrix} \Delta x_S \\ \Delta y_S \end{bmatrix} \tag{5.1}$$

$$\text{Avec : } \beta = \frac{2h}{rw} \tan\left(\frac{\alpha}{2}\right) \tag{5.2}$$

Les déplacements par rapport au sol $[\Delta x_W \ \Delta y_W]^\mathsf{T}$ sont donc proportionnellement liés aux déplacements bruts lus des deux capteurs $[\Delta x_S \ \Delta y_S]^\mathsf{T}_{l|r}$.

Une autre configuration de capteurs est possible en plaçant les deux capteurs en avant et en arrière comme illustré sur la figure 5.13. Cette configuration, contrairement à celle de la figure 5.11, exploite d'une manière simple les mesures sur les deux axes X et Y des capteurs.

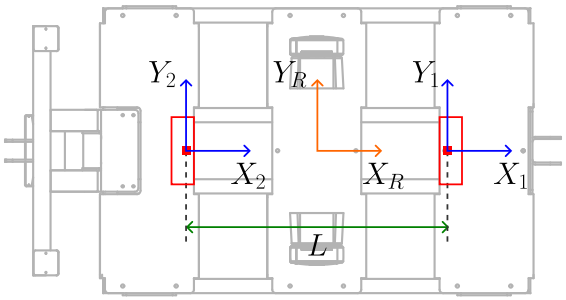


Figure 5.13 – Montage alternatif des capteurs de flux optique (carrés rouges), en avant (1) et arrière (2), avec les axes associés aux capteurs et au robot (R).

5.3.2.4 Calibrage du système

Le robot peut être contrôlé en vitesse linéaire et vitesse angulaire, v et ω respectivement. En supposant une configuration comme celle de la figure 5.13, l'axe X_1 du capteur monté en avant mesure les déplacements selon l'axe X_R du robot. Or, la contrainte de non-glissement sur l'axe Y_R empêche le robot de se déplacer selon l'axe Y_R, les mouvements mesurés par le capteur monté en avant sur l'axe Y_1 sont donc liées aux changements de l'orientation du robot *(vitesses angulaires)*. De la même manière, l'axe X_1 du capteur mesure les changements en déplacement linéaire du robot *(vitesses linéaires)*.

Notons qu'avec un montage en avant ou en arrière, un seul capteur de flux optique suffirait pour la mesure des déplacements du robot. Néanmoins, l'utilisation de deux capteurs peut être intéressante pour assurer une redondance des données et pour diminuer l'incertitude des mesures.

Les déplacements $[\Delta x\ \Delta y\ \Delta\phi]^\mathsf{T}$ estimés avec les données des capteurs avant (1) et arrière (2) sont donnés par l'équation (5.3).

$$\begin{bmatrix}\Delta x_R\\ \Delta y_R\\ \Delta\phi_R\end{bmatrix} = \begin{bmatrix}\dfrac{\beta_1\Delta x_1 - \beta_2\Delta x_2}{2}\cos\phi_R\\ \dfrac{\beta_1\Delta x_1 - \beta_2\Delta x_2}{2}\sin\phi_R\\ \left(\dfrac{\beta_2\Delta y_2 - \beta_1\Delta y_1}{L}\right)\end{bmatrix} \tag{5.3}$$

Avec β_1 et β_2 les coefficients du calibrage des capteurs avant et arrière respectivement, donnés par l'équation (5.2).

Le calibrage dépend de plusieurs facteurs, à savoir le champ de vision du capteur α, la résolution r, la constante interne w, et la hauteur du capteur h (voir la figure 5.10a). Les deux premiers sont connus du datasheet du capteur, par contre la valeur de la constante interne w n'est pas spécifiée dans la documentation. Nous avons donc défini une procédure de calibration qui permet de calculer directement le coefficient β qui regroupe toutes les constantes ainsi que la hauteur (la distance capteur-sol).

Le calibrage est réalisé en exécutant un mouvement linéaire sur une distance connue D. Ce mouvement linéaire est répété N fois, et à chaque fois, une mesure de cette distance est calculée par le capteur avant et le capteur arrière $X = [x_1\ x_2]^\mathsf{T}$. Les coefficients de calibration dans ce cas peuvent être calculés de la manière suivante :

$$\begin{bmatrix}\beta_1\\ \beta_2\end{bmatrix} = \frac{1}{D \cdot N} \sum_{i=1}^{N} \begin{bmatrix}x_1\\ x_2\end{bmatrix}_i \tag{5.4}$$

5.3.2.5 Données fournies

En plus des données brutes des vecteurs de déplacement selon les axes X et Y $[\Delta x_S\ \Delta y_S]^\mathsf{T}$, le capteur de flux optique fournie également indicateur booléen *(Flag)* précisant si un mouvement a été détecté entre deux acquisitions successives. Le capteur fournie par ailleurs un indicateur sur la qualité de la surface lors de l'acquisition des images. Cet indicateur décrit la qualité de texture de la surface observée par le capteur (dans notre cas, le sol), et permet donc de quantifier le niveau de fiabilité des données utilisées dans le calcul du flux optique.

L'indicateur de la qualité de surface est un entier sur 8-bits, il peut donc être utilisé pour calculer une sorte d'incertitude sur les mesures d'une façon dynamique [1].

5.3.3 Le LiDAR IDEC SE2L

La plateforme intègre aussi un LiDAR de sécurité de type *IDEC SE2L* (figure 5.14a) dont les caractéristiques principales sont décrites dans le tableau 5.6. Le IDEC SE2L est un LiDAR *à base d'impulsions (PB-LiDAR)* qui se base sur le calcul du temps de vol (ToF) de courtes impulsions de lumière infrarouge. La section 3.3 du chapitre 3 détaille les différentes technologies des LiDARs.

Ce LiDAR est connecté directement aux roues *SWD® Core* gauche et droite, via des entrées/sorties sécurisées OSSD. Ceci permet d'assurer les fonctionnalités de sûreté du bas niveau (SLS et SDI) décrites dans le chapitre 3.

Le LiDAR peut par ailleurs délivrer les mesures de distances des points perçus. Ces données peuvent être utilisées par la suite pour alimenter les algorithmes de la localisation et de la navigation. Les données du LiDAR sont représentées sous le format standard `sensor_msgs/LaserScan` de ROS [2].

5.3.4 L'odométrie des roues

Les roues *SWD® Core* (figure 5.14b) utilisées dans la plateforme expérimentale sont dotées de codeurs à effet Hall, chacun fourni 420 impulsions par révolution de roue.

1. Nous avons représenté les données acquises des deux capteurs de flux optiques par la structure `OpticalFlowDuo` qui regroupe toutes les données délivrées par ces capteurs. La section B.1.2 de l'annexe B décrit le format détaillé de cette structure.
2. Voir la section B.1.1 de l'annexe B pour le format détaillé.

Table 5.6 – Les caractéristiques du LiDAR IDEC SE2L utilisé.

Paramètre	**Spécification**
Champ de protection [m]	5
Champ d'avertissement [m]	20
Portée de mesure [m]	40
Champ angulaire [deg]	270
Résolution angulaire [deg]	0.125
Temps de balayage [ms]	30
Tolérance de détection [mm]	100
Champ de détection	Du noir-réflecteur de 1.8% au rétro-réflecteur
Type d'optique	PLD *(Pulsed Laser Diode)*
Classe du laser	Class 1 (IEC60825-1)
Longueur d'onde [nm]	905

Les données remontées par les codeurs sont utilisées dans un modèle d'entrainement différentiel pour calculer l'odométrie au centre de robot comme précisé dans la section 3.5.1.2 du chapitre 3.

(a) Le LiDAR de sécurité IDEC SE2L **(b)** SWD® Core

Figure 5.14 – Le moteur et le LiDAR utilisés dans le jeu de données.

Les données de l'odométrie sont publiées dans ROS à une cadence de 20Hz. Sous ROS, le message de l'odométrie est publiée sous un format standardisé spécifié par la structure `nav_msgs/Odometry`[3]. Ce message contient la pose et sa covariance, calculées incrémentalement à l'aide des codeurs, ainsi que les vitesses linéaires et angulaires et leurs covariances calculées à partir des mêmes données.

3. Voir la section B.1.3 de l'annexe B pour le format détaillé des données de l'odométrie.

5.4 Le jeu de données

Pour avoir un jeu de données représentatif des environnements industriels, nous nous sommes rendus sur le site l'entreprise Provost située à Lille. Provost est le numéro 1 français de la fabrication de rayonnages et d'équipement pour la manutention industrielle depuis 1963. Dans le cadre d'une collaboration entre ez-Wheel et Provost, nous avons organisé une visite aux entrepôts de l'entreprise pour conduire notre expérience.

Nous avons choisi deux types d'environnements pour la collecte des données, le premier est un nouvel entrepôt construit principalement en charpente métallique, que nous désignons ici par « entrepôt I ». Le deuxième est un ancien entrepôt avec un plafond construit essentiellement en bois et en métal, que nous désignons ici par « entrepôt II ». La figure 5.15 montre quelques photos des deux entrepôts.

Le jeu de données construit consiste en six (6) séquences, quatre dans l'entrepôt I et deux dans l'entrepôt II. La figure 5.16 illustre quelques exemples d'images acquises à partir des différentes caméras. La ligne (A) représente des images de la caméra RGB orientée vers le plafond ; (B) les images de la caméra RGB frontale ; (C) et (D) les images des caméras verticales gauches et droites, respectivement ; (E) et (F) les images des caméras frontales, gauche et droite, respectivement. Les colonnes (1) et (2) représentent deux instants d'une séquence enregistrée dans l'entrepôt I ; (3) et (4) représentent deux instants d'une séquence enregistrée dans l'entrepôt II.

Table 5.7 – Statistiques globales des données récoltées sur les six (6) séquences de notre jeu de données.

Séquence	**1**	**2**	**3**	**4**	**5**	**6**
Entrepôt	I	I	I	I	II	II
Durée [s]	347	274	330	244	557	236
N° d'images frontales (G/D)	10418	8216	9907	7324	16722	7076
N° d'images frontales (RGB)	10418	8217	9907	7323	16722	N/A
N° d'images plafond (G/D)	10421	8219	9908	7325	16726	7079
N° d'images plafond (RGB)	10420	8218	9908	7324	16726	7078
N° de mesures IMU (cam. frontale)	N/A	109604	132135	97693	223057	N/A
N° de mesures IMU (cam. verticale)	139148	109741	132299	97811	223351	94520
N° de mesures du flux optique	34752	27409	33046	24429	55782	23606
N° de mesures de l'odométrie	6950	5481	6608	4885	11156	4721
N° de scans du LiDAR	11904	9386	11309	8342	19082	8075

Le tableau 5.7 récapitule les statistiques globales des données contenues dans chacune des six séquences. L'acronyme *(G/D)* dans le tableau signifie (gauche/droit), et représente les images des capteurs proches-infrarouge des modules stéréoscopiques de la caméra frontale et de la caméra orientée vers plafond (caméra verticale).

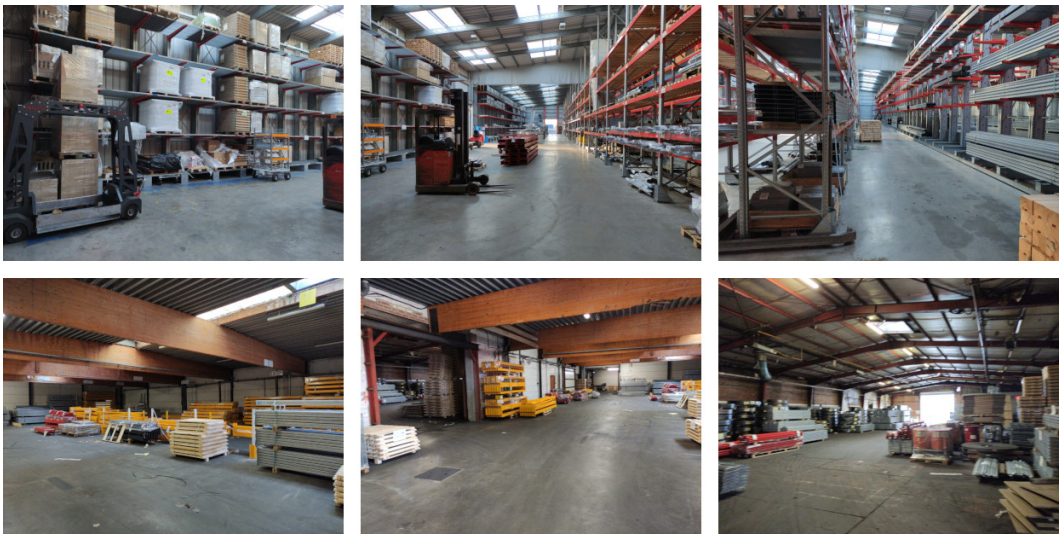


Figure 5.15 – Photos du site Provost. Première ligne, nouvel entrepôt en charpente métallique *« entrepôt I »*. Deuxième ligne, ancien entrepôt *« entrepôt II »*.

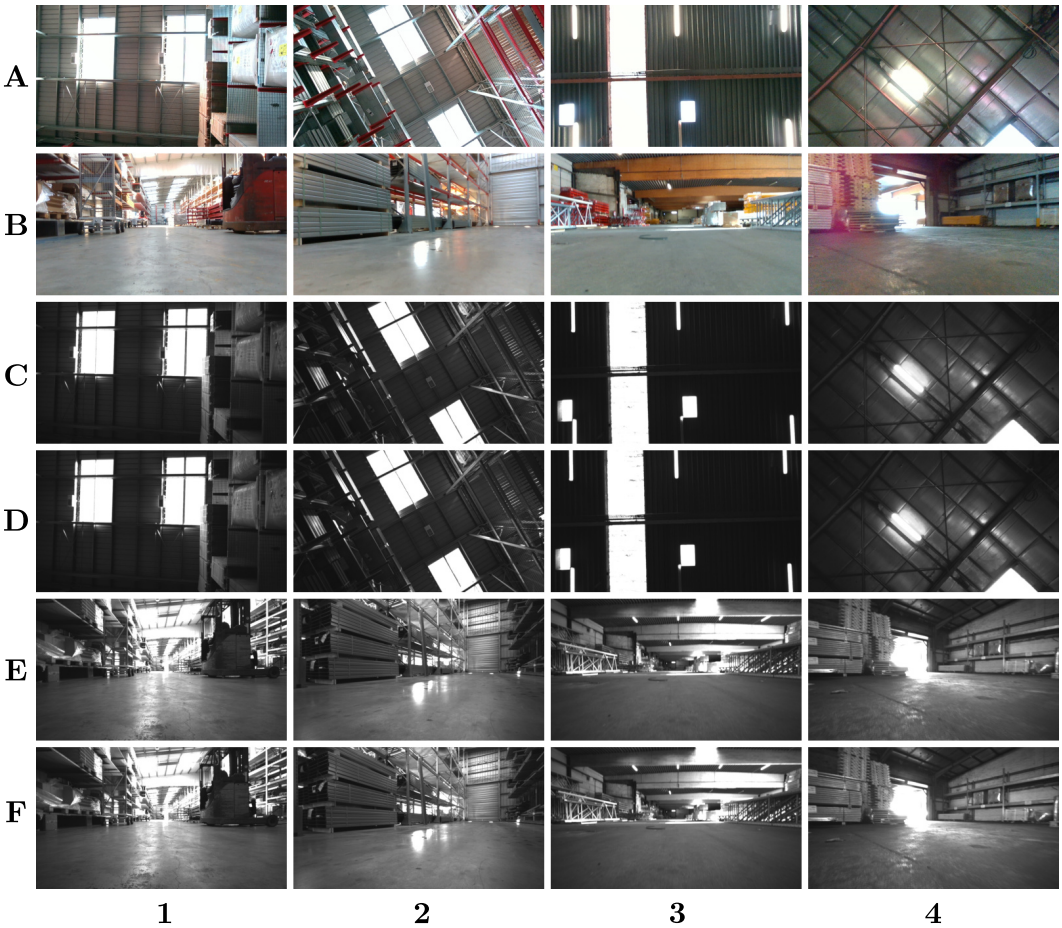


Figure 5.16 – Exemples d'images acquises par les différentes caméras, tirées de notre jeu de données.

Les images acquises par toutes les caméras sont de taille 480×848. Les informations de calibration des caméras sont données par la matrice de projection $\mathbf{P}$ et les paramètres de distorsion $\mathbf{D}$ du *modèle radial-tangentiel*[4] tel que :

$$\mathbf{P} = \begin{bmatrix} f_x & 0 & c_x & T_x \\ 0 & f_y & c_y & T_y \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.5}$$

$$\mathbf{D} = \begin{bmatrix} k_1 & k_2 & t_1 & t_2 & k_3 \end{bmatrix} \tag{5.6}$$

Plus d'informations sur l'interprétation de ces paramètres peuvent être trouvées dans la section 4.2 du chapitre 4.

Il convient de noter que les paramètres T_x et T_y ici sont particulièrement utiles dans le cas des caméras stéréoscopiques, ils spécifient la position du centre optique de la première caméra par rapport au centre optique de la deuxième caméra.

Les valeurs des paramètres de calibration des caméras utilisées sont données dans le tableau 5.8.

Table 5.8 – Valeurs de calibration des caméras utilisées, G/D pour gauche/droite.

Paramètre	D435i RGB	D435i G	D435i D	D455 RGB	D455 G	D455 D
f_x	611.296	418.564	418.564	419.757	425.541	425.541
f_y	610.802	418.564	418.564	419.451	425.541	425.541
c_x	427.756	424.311	424.311	422.232	425.496	425.496
c_y	238.753	244.403	244.403	243.969	237.351	237.351
T_x	0	0	−20.915	0	0	−40.444
T_y	0	0	0	0	0	0
k_1	0	0	0	−0.05572	0	0
k_2	0	0	0	0.06603	0	0
t_1	0	0	0	−0.00055	0	0
t_2	0	0	0	0.00088	0	0
k_3	0	0	0	−0.02108	0	0

5.5 Génération d'une vérité terrain

La vérité terrain est un sujet crucial dans chaque jeu de données. En effet, pour bien exploiter un jeu de données, nous avons besoin d'une référence établissant la réalité de l'expérimentation. Ainsi, pour un jeu de données d'images destiné à la localisation indoor, nous devons être capables de fournir une trajectoire de référence pour chaque séquence de données.

Après une synthèse de la littérature scientifique, nous pouvons classer les approches d'obtention d'une vérité terrain en trois familles :

4. Le modèle radial-tangentiel est désigné dans les données ROS par le nom `"plumb_bob"`.

- Systèmes de localisation de haute précision ;
- Traçage au sol ;
- Reconstruction par calcul exhaustif.

5.5.1 Systèmes de localisation de haute précision

Cette stratégie consiste en l'utilisation d'un système de localisation suffisamment précis pour être considéré comme une référence. Pour la localisation outdoor, le *GPS RTK (Real-time Kinematic)* est le standard de facto [143, 148, 149].

Le GPS RTK permet de corriger les mesures du GPS pour surmonter les limitations relatives à cette technologie. Les données de correction sont obtenues à partir d'un ou plusieurs balises GPS fixes (station de référence) placés sur des positions dont les coordonnées sont parfaitement connues. Il est ainsi possible de quantifier les différentes erreurs et de les corriger pour pouvoir ensuite les partager avec d'autres récepteurs GPS situés à proximité.

Pour les systèmes de localisation indoor, il n'y a pas d'équivalent fiable au GPS RTK. D'une manière générale, les solutions similaires, à savoir, des solutions à base de radiofréquences centralisées (de type *UWB - Ultra-wide Band*) peuvent être facilement perturbées par la structure de l'environnement (ligne de mire, multichemins, structure métallique, *etc.*) [159].

Pour produire une vérité de terrain dans un environnement indoor, un système de *capture de mouvement (MoCap - Motion Capture)* peut être la solution la plus adéquate [149, 156]. Il s'agit d'un réseau de caméras qui monitorent le robot qui est équipé de marqueurs (passifs ou actifs) pour faciliter la détection et le suivi de ces derniers dans les images. Ces systèmes font une triangulation des marqueurs observés par toutes les caméras afin de reconstruire les mouvements de ceux-ci. En revanche, ce type de systèmes est destiné, d'une manière générale, à des environnements ouverts *(Open Spaces)* et ils ne sont pas adaptés aux grands environnements industriels. En plus, le coût d'un tel système peut poser une contrainte forte, surtout qu'il accroît rapidement avec la taille de l'environnement à couvrir.

D'autres jeux de données utilisent un ensemble de LiDARs 2D ou 3D fixés sur l'environnement pour estimer les mouvements de la plateforme robotique d'une manière similaire aux systèmes de capture de mouvement [148]. D'autres jeux de données utilisent des LiDARs 3D montés sur le robot pour estimer, à postériori, la structure de l'environnement et la trajectoire réalisée [149].

Le choix de cette stratégie nécessite donc la disponibilité d'un système de localisation indoor d'une haute précision. L'absence d'un système de localisation précis nécessitera un grand investissement en temps et en ressources pour acquérir ou concevoir un tel système. Il n'est alors pas pertinent d'aller dans cette direction pour construire un outil qui ne servira qu'à la validation de notre travail à des phases très particulières.

5.5.2 Traçage au sol

À l'aide d'un traçage au sol, le robot peut récolter les données de ses capteurs tout en suivant ce traçage en utilisant un système de suivi de ligne. Ainsi, la trajectoire réalisée dans ce cas serait préalablement connue, les mesures de la trajectoire tracée au sol doivent être prélevées le plus fidèlement possible pour construire la trajectoire de référence.

Le problème avec ce type de vérité terrain est la complexité de réalisation d'une telle référence [160]. En effet, cela impose des difficultés liées au suivi par le robot du traçage préétabli, à la prise de mesures sur place, à la datation des points de la trajectoire de référence par rapport aux données collectées, et à l'exécution de trajectoires complexes.

Cette solution a donc été éliminée, car trop contraignante et pas forcément réaliste sur le terrain, notamment lorsque les expériences sont réalisées sur des sites industriels.

5.5.3 Reconstruction par calcul exhaustif

Dans cette stratégie, les données du jeu de données sont utilisées pour le calcul de la trajectoire de référence à l'aide d'un algorithme de calcul exhaustif qui permet d'avoir un résultat proche de la réalité [161].

Dans le contexte de la localisation à base de vision, ce type d'algorithmes est généralement utilisé pour la reconstruction de scène 3D. Un tel algorithme met en correspondance les caractéristiques détectées dans les images, d'une manière exhaustive, pour reconstruire la structure 3D de l'environnement à partir des différences d'apparence des caractéristiques causées par le mouvement de la caméra. C'est ce qu'on désigne par la technique de la structure à partir du mouvement *(SfM - Structure from Motion)*.

Pour valider notre jeu de données, nous avons opté pour ce type de vérité terrain. En effet, cette technique nous permet d'avoir des trajectoires de référence à bas coût (*c.-à-d.*, sans investir dans un système de localisation plus précis qui serait trop coûteux) en plus d'être assez flexible pour la réalisation de trajectoires complexes et plus réalistes.

5.6 L'outil de photogrammétrie COLMAP

Pour générer les trajectoires de référence, nous nous sommes basées sur le logiciel de photogrammétrie COLMAP. Il s'agit d'un outil libre et open source de reconstruction 3D proposé par Schönberger et al. [162].

COLMAP est un pipeline générique et polyvalent de *structure à partir de mouvement (SfM - Structure from Motion)* et de vision *stéréoscopique multivue (MVS - Multi-View Stereo)*. Il offre une interface graphique et une interface en ligne de commande. Nous avons utilisé cette dernière pour implémenter un script de traitement par lots *(Batch processing)* des différentes séquences d'images de notre jeu de données.

La figure 5.17 décrit les étapes du pipeline incrémental de la *structure à partir de mouvement (SfM)* implémenté dans COLMAP.

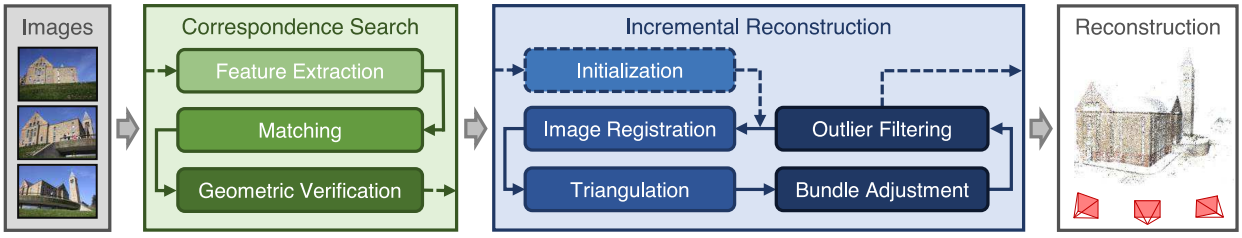


Figure 5.17 – Le pipeline incrémental *structure à partir de mouvement (SfM - Structure from Motion)* implémenté dans COLMAP (tiré de [162]).

Le pipeline de traitement consiste en deux étapes principales, la recherche des correspondances *(Correspondance Search)* et la reconstruction incrémentale *(Incremental Reconstruction)*.

5.6.1 La recherche des correspondances

Dans cette phase, l'algorithme cherche les caractéristiques communes entre les images d'entrée, qui peuvent être acquises dans un ordre séquentiel ou pas. L'heuristique utilisée pour la recherche des correspondances peut différer, comme nous allons l'expliquer, selon le mode d'acquisition des images.

La première étape consiste en l'extraction des caractéristiques à partir des images d'entrée *(Feature Extraction)*. COLMAP utilise l'algorithme SIFT *(Scale-Invariant Feature Transform)* [163] pour la détection et la description des caractéristiques.

L'étape d'après cherche les potentiels chevauchements de scènes entre les images. Ceci est réalisé en mettant en correspondances les caractéristiques détectées dans les différentes images *(Feature Matching)*, à l'aide du même algorithme SIFT.

La stratégie naïve pour la mise en correspondance teste d'une manière exhaustive chaque paire d'images pour chercher des correspondances *(Exhaustive Matcher)* ; ce qui est particulièrement utile dans le cas d'une reconstruction à partir d'un ensemble d'images non ordonnées.

Dans le cas où les images sont acquises d'une manière séquentielle avec des contraintes de covisibilité entre les images successives, une autre stratégie peut être utilisée. Cette stratégie consiste à tester les images d'une façon séquentielle *(Sequential Matcher)*. Ceci permet d'économiser le temps de calcul, car nous ne cherchons qu'à tester les correspondances de chaque image avec un nombre réduit d'images qui la suivent/précedent au lieu de tester toutes les images.

COLMAP implémente aussi d'autres stratégies de mise en correspondances, à savoir, la stratégie transitive, spatiale et à base de vocabulaire visuel [162].

La troisième étape de la première phase vérifie la géométrie des paires d'images qui se chevauchent potentiellement. Étant donné que l'appariement (la mise en correspondance) est basé uniquement sur l'apparence, il n'est pas garanti que les correspondances détectées représentent réellement le même point dans la scène. Pour remédier à cela, la SfM vérifie les correspondances en essayant d'estimer une transformation qui permet de lier les caractéristiques entre les images à l'aide de la géométrie projective. Si l'algorithme trouve

une transformation qui décrit d'une façon unique la relation géométrique entre un nombre suffisant de points dans les deux images testées, ces correspondances sont considérées comme géométriquement vérifiées.

À ce stage, et étant donné que les correspondances issues de l'appariement sont souvent contaminées par des valeurs aberrantes, des techniques d'estimation robustes, telles que RANSAC *(Random Sample Consensus)*, sont appliquées.

À la fin de cette phase, un graphe de la scène est produit, décrivant les relations de covisibilité entre les images et entre les caractéristiques détectées dans chacune de ces images.

5.6.2 La reconstruction incrémentale

Cette phase utilise le graphe de la scène produit lors de la phase précédente pour estimer les poses de la caméra ainsi que pour reconstruire la scène à partir des images d'entrée [118].

L'initialisation de ce processus est une étape cruciale. Au début, l'algorithme choisie une paire d'images à utiliser. Ces images peuvent également être sélectionnées manuellement afin d'assurer un bon démarrage du processus de la SfM [162].

Pendant la reconstruction incrémentale, les nouvelles images peuvent être enregistrées dans le modèle actuel de l'environnement. Cela est effectué en résolvant le problème *Perspective-n-Point (PnP)*, à l'aide des correspondances des caractéristiques avec les points triangulés à partir des images déjà enregistrées [118]. Dans le problème PnP, nous cherchons à estimer la pose de la caméra à partir des images bidimensionnelles (2D) observées.

Le chevauchement des images permet d'observer un certain nombre de points dans plusieurs images sur les régions déjà observées, et de trouver de nouveaux points dans les régions nouvellement observées. Ces nouveaux points seront ajoutés au modèle lorsque ces mêmes points sont retrouvés dans les nouvelles images prises sous un angle différent. Cela permet de trianguler ces points dans l'espace tridimensionnel afin de reconstruire leur géométrie.

En l'absence d'une stratégie de raffinement des estimations lors de l'enregistrement et de la triangulation, la SfM peut cumuler une dérive très rapidement. L'étape de l'*ajustement de faisceaux (BA - Bundle Adjustment)* traite ce problème en réalisant une minimisation de l'erreur de rétro-projection sur toutes les images, effectuée à l'aide du système suivant [162] :

$$E = \sum_j \rho_j \left(\left\| \Pi \left({}^o\mathbf{X}_\mathcal{P}, \left({}^c\mathbf{T}_o \right)_t \right) - \left({}^i\mathbf{X}_\mathcal{P} \right)_j \right\|_2^2 \right) \tag{5.7}$$

Avec E l'erreur à minimiser, ρ_j une fonction de coût qui peut être utilisée pour pénaliser les valeurs aberrantes, et $\Pi(\cdot)$ est la fonction qui permet de projeter un point tridimensionnel (3D) de la scène à un point bidimensionnel (2D) sur le plan de l'image. L'équation 5.7 quantifie l'écart en pixels entre un point mesuré dans le plan de l'image ${}^i\mathcal{P}$ de coordonnées ${}^i\mathbf{X}_\mathcal{P}$ et un point 3D de la scène ${}^o\mathcal{P}$ de coordonnées ${}^o\mathbf{X}_\mathcal{P}$ projeté dans le plan de l'image en utilisant la fonction de projection Π et la pose actuelle de la caméra $\left({}^c\mathbf{T}_o \right)_t$.

Ainsi, l'ajustement de faisceaux minimise l'erreur de l'équation 5.7 en utilisant une méthode d'optimisation telle que Gauss-Newton ou Levenberg-Marquardt [164], la dernière étant la plus utilisée pour résoudre ce type de problèmes. Ceci permet de raffiner les paramètres de la pose de la caméra ainsi que les positions des points dans le modèle tridimensionnel reconstitué.

5.7 Validation du jeu de données

Pour valider notre jeu de données, nous avons exploité l'outil de photogrammétrie COLMAP [5]. Les images étant acquises à basses vitesses (un maximum de $0.5\text{m} \cdot \text{s}^{-1}$), nous avons sous-échantillonné les images des caméras d'un facteur de $\frac{1}{4}$ avant de les utiliser dans COLMAP. En effet, cela permet de réduire considérablement le temps de calcul de COLMAP tout en gardant une bonne précision.

La reconstruction conjointe de la trajectoire et de la scène tridimensionnelle a été réalisée sur une machine munie d'un processeur Intel® Core™ i7-11850H de 2.50GHz et un processeur graphique (GPU) de type NVIDIA T1200 (TU117GLM). Les statistiques de la reconstruction des six séquences sont résumées dans le tableau 5.9.

Table 5.9 – Les statistiques de la reconstruction des six séquences par COLMAP.

Séquence	**1**	**2**	**3**	**4**	**5**	**6**
N° d'images plafond	10421	8219	9908	7325	16726	7079
N° d'images traitées	2606	2055	2477	1832	4174	1770
N° de points 3D projetés	256K	133K	227K	61K	120K	51K
N° d'observations	3.05M	1.82M	2.62M	1.23M	2.13M	805K
N° moyen d'obs./image	1169.93	884.66	1058.05	668.97	510.95	454.90
N° moyen d'obs./point	11.93	13.66	11.53	19.97	17.84	15.90
Err. moyenne de reprojection [px]	0.696	0.655	0.695	0.457	0.611	0.513

La partie haute du tableau 5.9 montre le nombre d'images par séquence, le nombre d'images traitées par COLMAP (après le sous-échantillonnage), le nombre de points 3D reconstruits et le nombre total d'observations (caractéristiques observées sur toutes les images). La partie basse du tableau montre la moyenne d'observations par image, la moyenne des observations utilisées pour trianguler un point 3D de la scène et l'erreur moyenne de rétro-projection (en pixels) sur toute la trajectoire.

Sur toutes les séquences enregistrées, le robot part d'une position marquée au sol (figure 5.18), ensuite, il fait le tour en réalisant une trajectoire complexe puis retourne au point de départ.

5. Voir la section B.2 de l'annexe B pour plus de détails sur la configuration utilisée dans COLMAP.

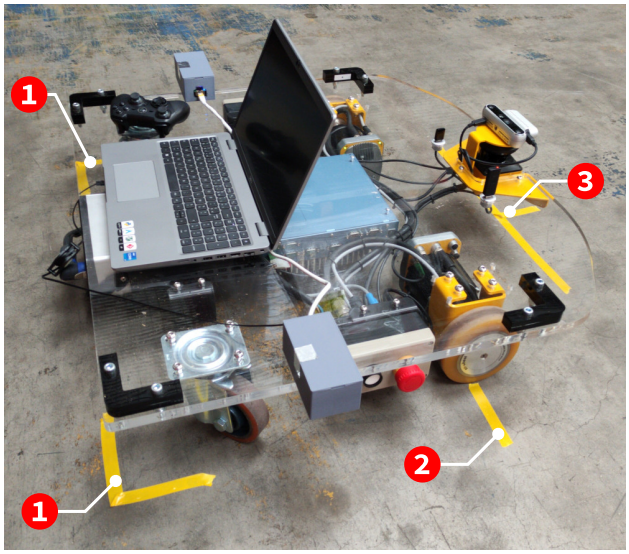


Figure 5.18 – Le marquage au sol du point de démarrage. (1) empreinte du robot (derrière) ; (2) l'axe des deux roues ; (3) traçage de l'empreinte (devant) du robot avec la position du LiDAR.

À la sortie de COLMAP, les trajectoires estimées, étant calculées seulement à partir d'images monoculaires, ne sont pas à l'échelle métrique. Pour remonter cela, nous avons calculé l'échelle métrique à partir des trajectoires estimées en appliquant l'algorithme LaMa SLAM sur les données du LiDAR (voir la figure 5.19). La méthode utilisée pour le calcul de l'échelle métrique a été précédemment détaillée dans la section 4.9.3 du chapitre 4. Le tableau 5.10 montre les distances parcourues sur chaque séquence, estimée à partir de la trajectoire COLMAP après la mise à l'échelle.

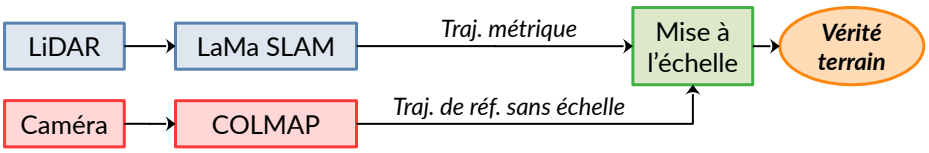


Figure 5.19 – La stratégie utilisée pour mettre la trajectoire estimée par COLMAP à l'échelle métrique.

Table 5.10 – Les distances parcourues au long des séquences, estimées à partir des trajectoires COLMAP après la mise en échelle.

Séquence	**1**	**2**	**3**	**4**	**5**	**6**
Distance parcourue [m]	142.4	114.4	143.0	88.1	238.0	101.7

Les trajectoires de référence estimées à l'aide de COLMAP et mises à l'échelle sont tracées dans les figures de 5.20 à 5.25. Le point rouge sur les trajectoires marque la position de départ, la croix verte dénote la position estimée à l'arrivée.

La figure 5.26 visualise les nuages de points épars générés par COLMAP pour les 6 séquences. Sur la figure sont représentées en rouge, les positions de la caméra lors de la réalisation de ces trajectoires.

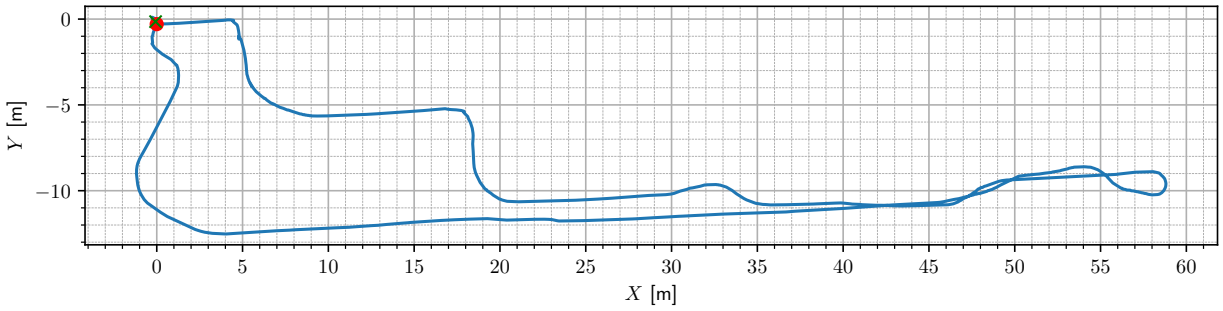


Figure 5.20 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 1).

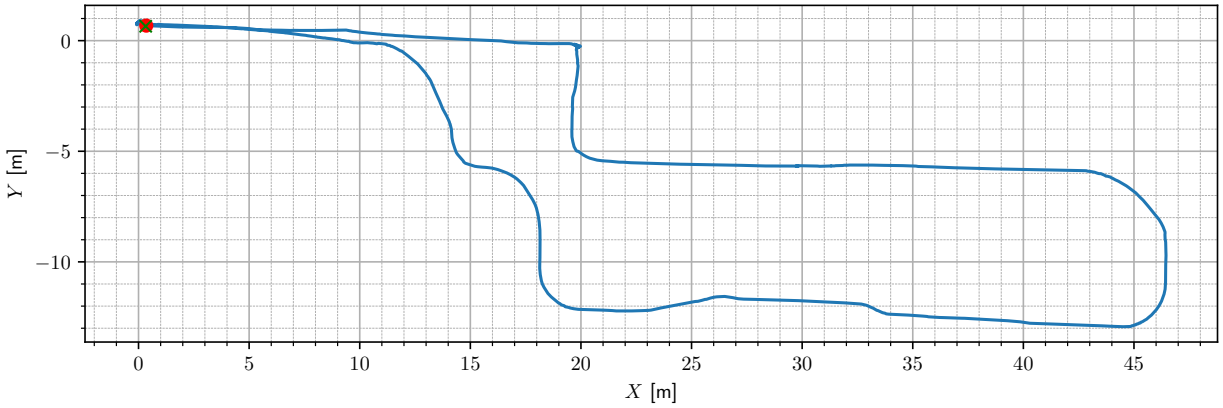


Figure 5.21 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 2).

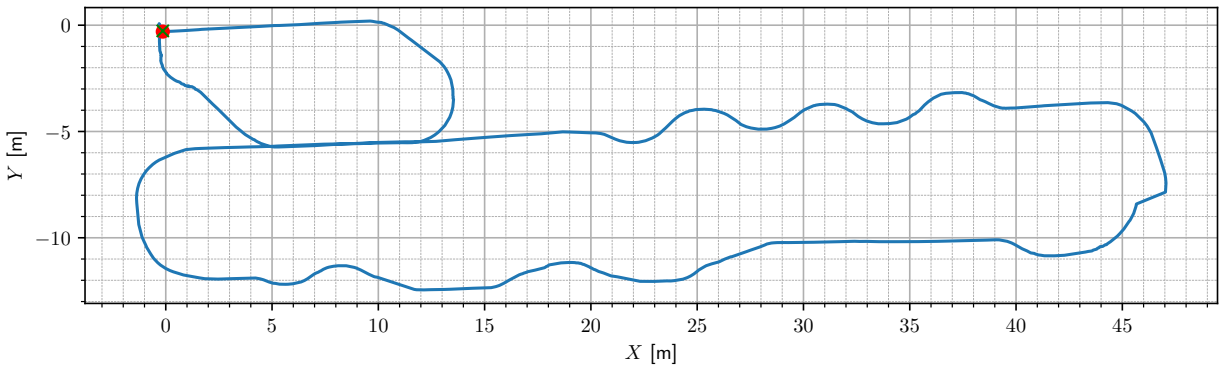


Figure 5.22 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 3).

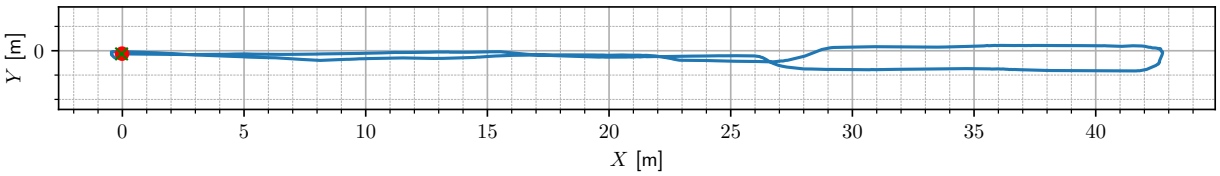


Figure 5.23 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 4).

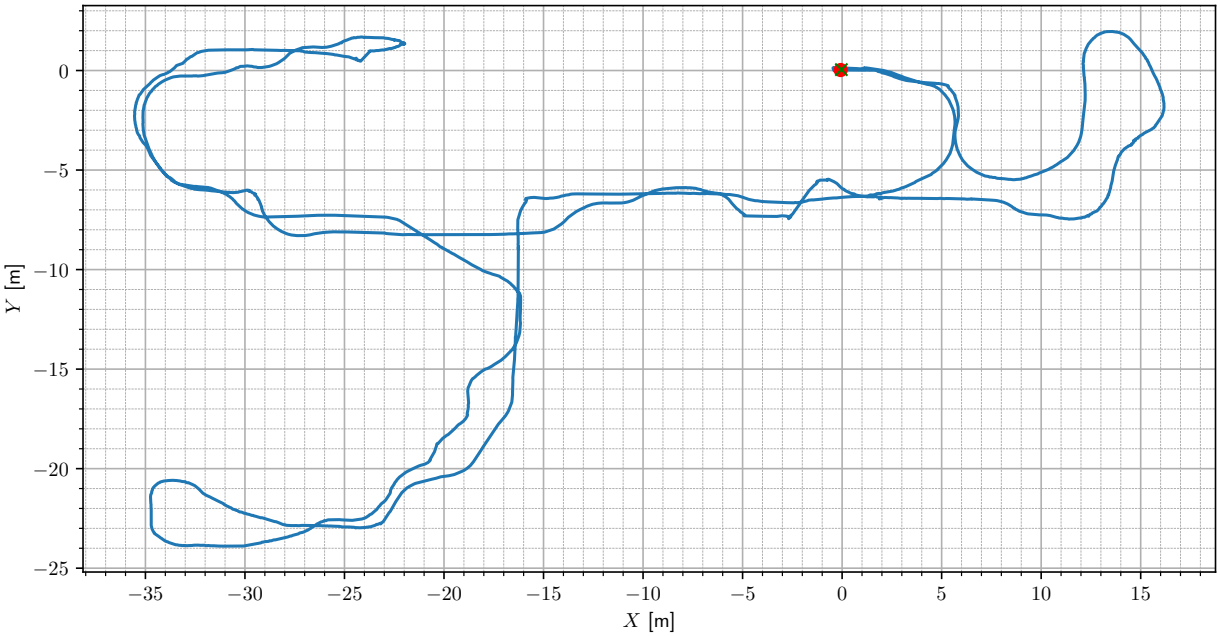


Figure 5.24 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 5).

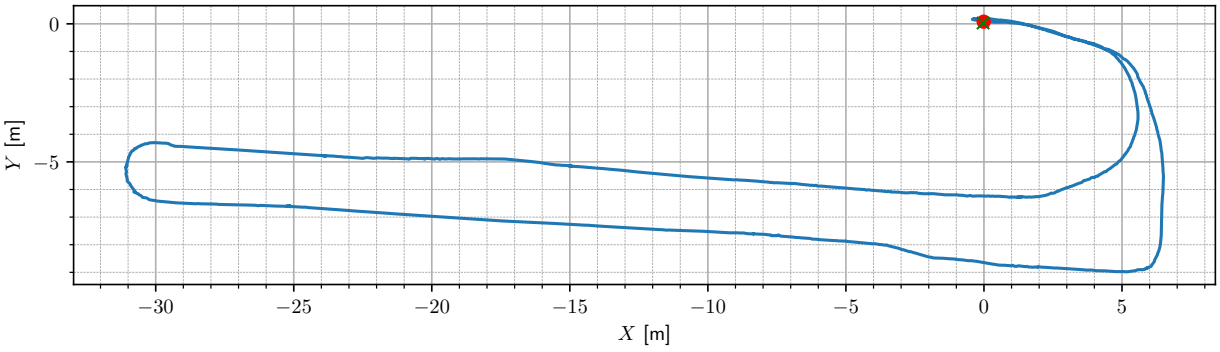


Figure 5.25 – Trajectoire estimée en utilisant COLMAP et mise à l'échelle (séquence 6).

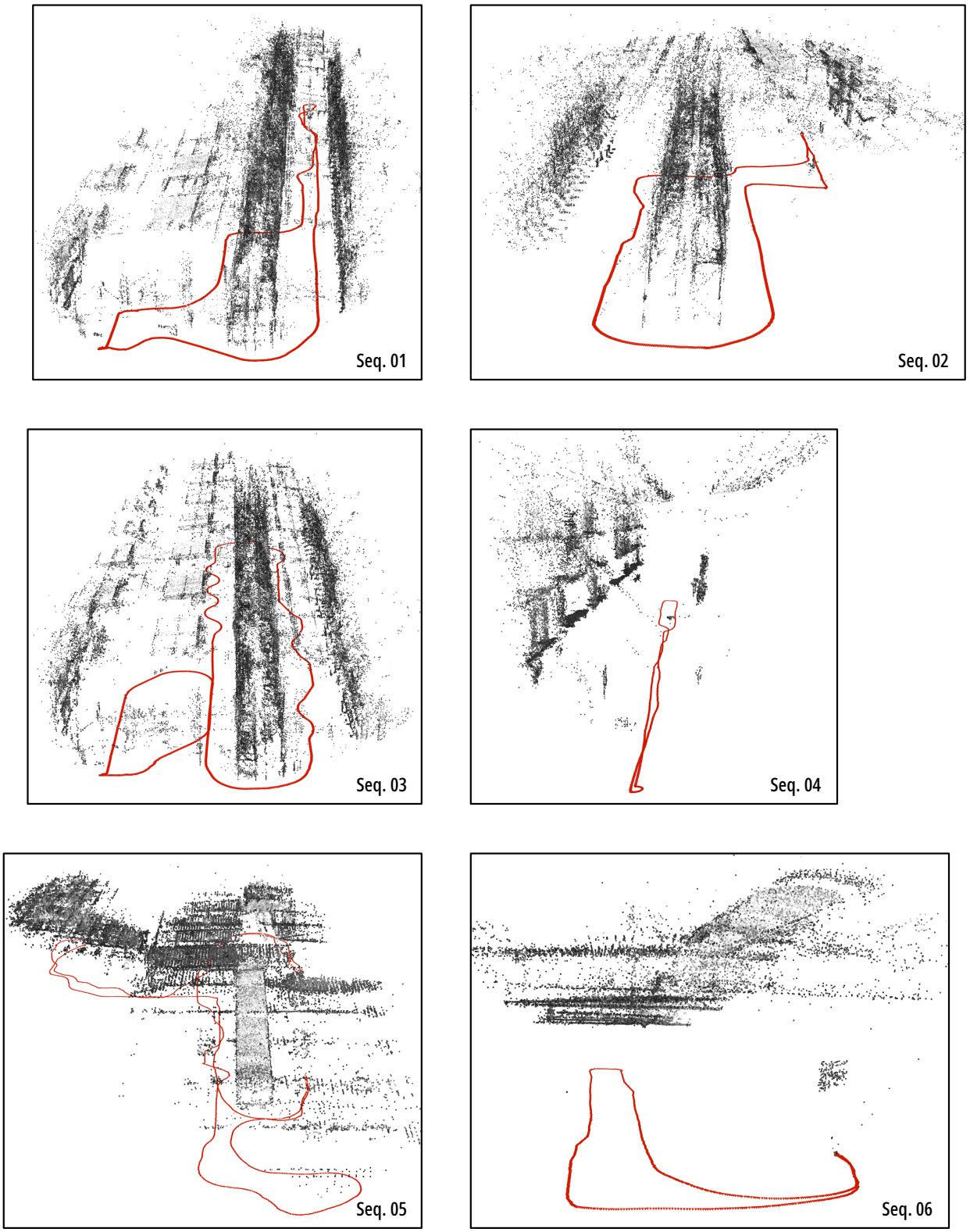


Figure 5.26 – Les trajectoires réalisées avec les nuages de points épars générés par COLMAP.

5.8 Validation de la Ceiling-DSO sur le jeu de données

Après la validation du jeu de données et le calcul des trajectoires de référence, nous avons exploité ce dernier pour procéder à la validation, en environnements réels, de la méthode Ceiling-DSO présentée dans le chapitre précédent.

Nous avons suivi la même méthodologie que celle décrite dans la section 4.9.3 du chapitre 4. Les différentes trajectoires estimées avec Ceiling-DSO ont été alignées par rapport à la trajectoire de référence, ensuite, nous avons calculé les erreurs relatives entre les deux pour évaluer la qualité de la trajectoire estimée.

Nous tenons à préciser que notre approche a échoué à estimer les trajectoires sur les séquences 5 et 6. Ces deux séquences ont été enregistrées, comme nous l'avons mentionné précédemment, dans un ancien entrepôt.

Les conditions d'éclairage dans ce dernier n'étaient pas optimales et la composition du plafond dans cet entrepôt était différente, ce qui a rendu moins robuste l'estimation des mouvements de points entre les images successives. Dans les deux séquences, l'algorithme Ceiling-DSO arrive au début à estimer une partie de la trajectoire, puis diverge rapidement. Nous allons discuter les causes de cet échec d'estimation sur les séquences 5 et 6 prochainement dans ce chapitre.

Les figures de 5.27 à 5.30 montrent les trajectoires estimées par Ceiling-DSO sur les quatre premières séquences. Ces trajectoires ont été calculées en prenant une taille d'images de 848×480, une fréquence d'images de 30fps, et une taille maximale de la fenêtre d'optimisation de 7.

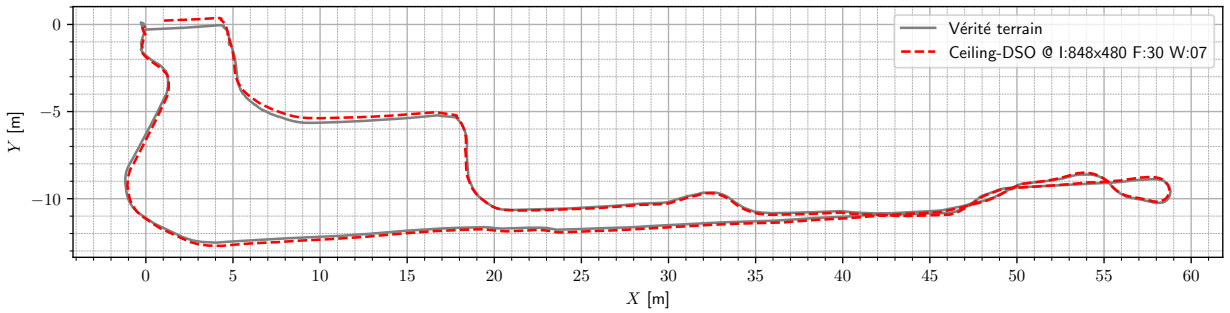


Figure 5.27 – Trajectoire de la Ceiling-DSO estimée sur la séquence 1 comparée à la trajectoire de référence.

Les résultats présentés dans la figure 5.31 montrent les erreurs relatives sur les trajectoires des séquences 1 à 4 sous forme de boîtes à moustaches. Ces erreurs sont calculées par rapport aux différentes combinaisons des paramètres de la résolution des images (I), la fréquence des images (F) et la taille de la fenêtre d'optimisation (W). Les résultats sont triés par les médianes des erreurs relatives de la séquence 1. La boîte délimite les premier et troisième quartiles et la ligne orange marque la médiane. Les lignes *(moustaches)* s'étendent de la boîte indiquant la variabilité à moins de $1.5 \times$ IQR.

Les résultats obtenus sur les séquences de 1 à 4 viennent confirmer la fiabilité de l'algorithme de l'odométrie visuelle Ceiling-DSO, en comparaison avec la vérité terrain calculée à l'aide du calcul photogrammétrique intensif de l'outil COLMAP.

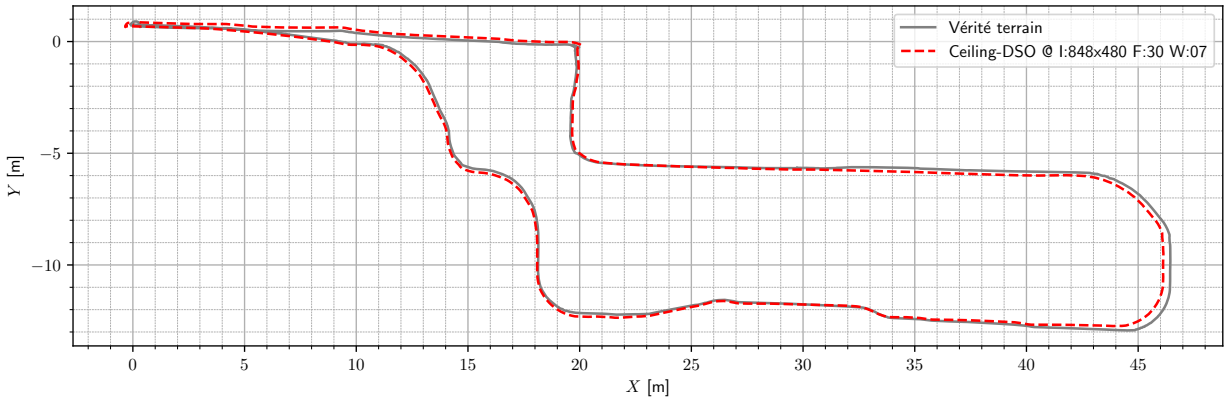


Figure 5.28 – Trajectoire de la Ceiling-DSO estimée sur la séquence 2 comparée à la trajectoire de référence.

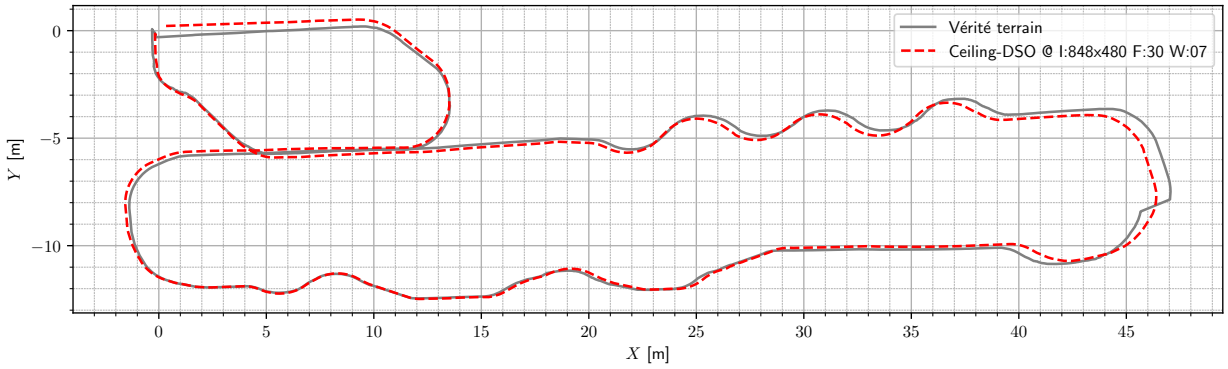


Figure 5.29 – Trajectoire de la Ceiling-DSO estimée sur la séquence 3 comparée à la trajectoire de référence.

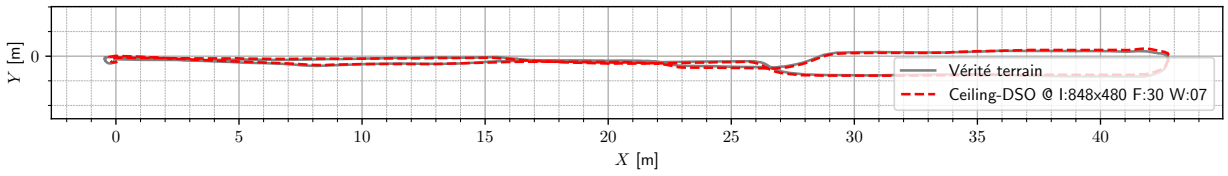


Figure 5.30 – Trajectoire de la Ceiling-DSO estimée sur la séquence 4 comparée à la trajectoire de référence.

Les résultats obtenus pour les séquences 5 et 6 ne sont pas concluants. Étant donné que l'entrepôt II est mal éclairé, nos premiers soupçons se sont portés sur le fait que la divergence observée dans les séquences 5 et 6 pourrait être due à d'importantes variations de luminosité.

En effet, l'algorithme DSO, étant une méthode directe (à base de minimisation de l'erreur photométrique), est intrinsèquement sensible aux changements brusques de luminosité. Dans la formulation de la DSO, les variations de luminosité entre les images consécutives sont

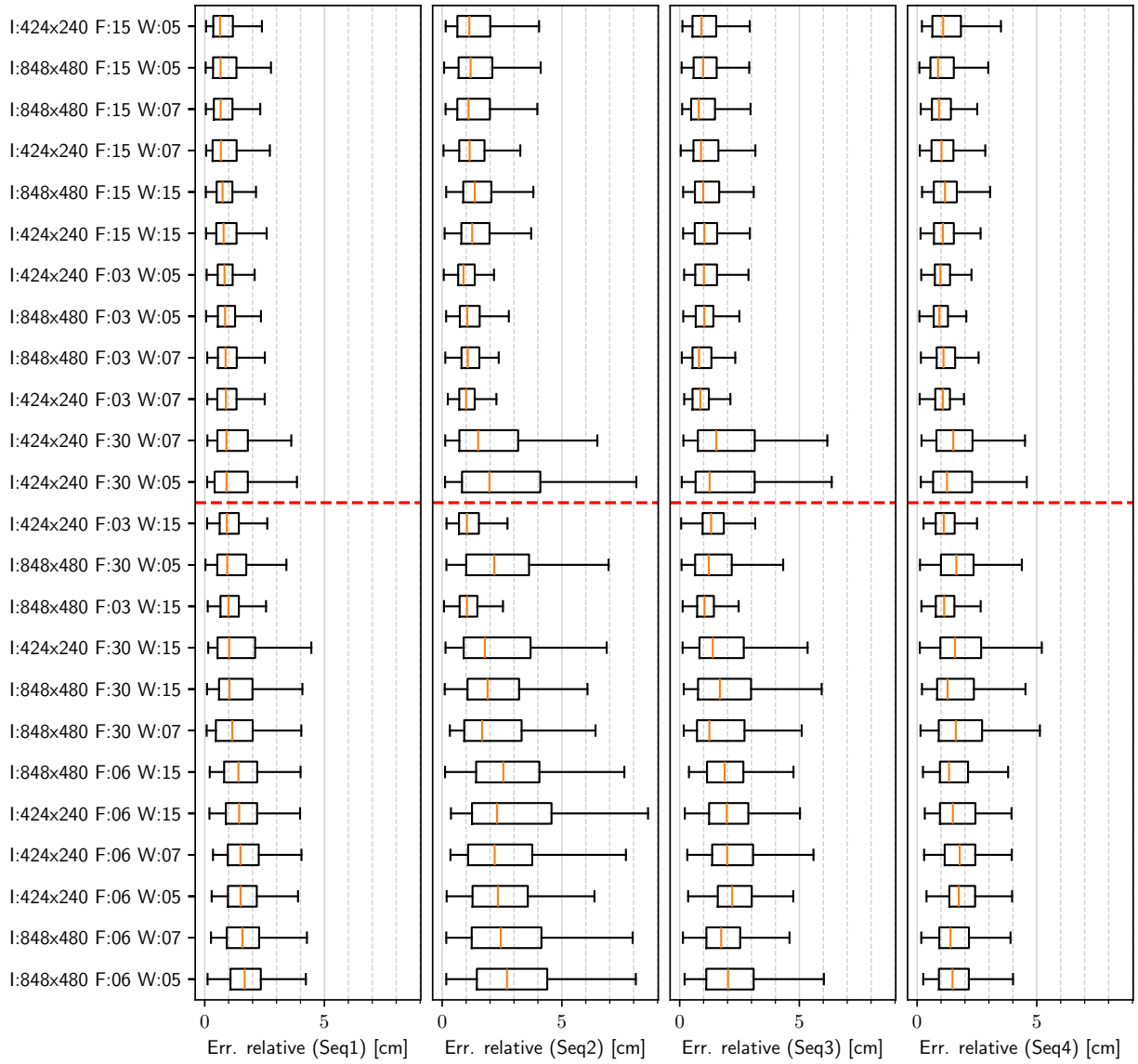


Figure 5.31 – Graphes en boîtes à moustache pour chaque séquence, représentant les *erreurs relatives* pour les combinaisons de paramètres testées (I : *résolution de l'image*, F : *fréquence d'images*, W : *taille de la fenêtre d'optimisation*). Les boîtes sont triées par les médianes des erreurs relatives de la *Seq1*.

modélisés dans le système optimisé et ils sont ainsi prises en considération (voir la section 4.8.1.1 du chapitre 4). Néanmoins, dans les cas extrêmes, cette prise en charge peut ne pas être suffisante, et ainsi, les estimations de l'algorithme peuvent diverger [165].

Pour les divergences des estimations constatées sur les séquences 5 et 6, la cause ne nous semble pas être liée à la luminosité, mais plutôt à une autre limitation bien connue dans le contexte du *flux optique* ou des *détecteurs de caractéristiques*, à savoir le *problème d'ouverture (Aperture problem)*.

D'une manière générale, dans une image, les régions avec des changements de contraste importants sont les plus faciles à détecter et à suivre. En revanche, même s'ils sont bien contrastés, les segments de ligne droite avec une seule orientation souffrent du problème d'ouverture [118]. La figure 5.32 montre une illustration du problème d'ouverture, les images perçues à travers les carrés (l'ouverture de la caméra) sont identiques bien que les mouvements (indiqués par les flèches) sont différents.

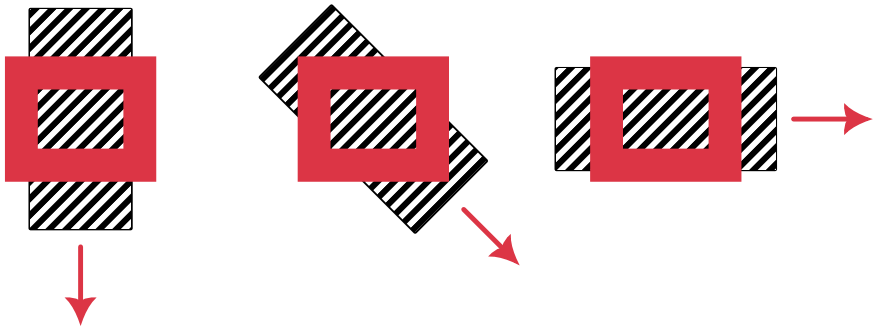


Figure 5.32 – Illustration du problème d'ouverture.

En examinant les données de la séquence 5 au moment de l'échec de l'algorithme Ceiling-DSO, nous avons observé que cela était effectivement lié à une zone présentant un alignement majoritaire des textures dans la direction du mouvement. Cependant, le problème d'ouverture rend difficile l'estimation du mouvement dans ces conditions. La figure 5.33 montre quelques images capturées, avec les points détectés et suivis par Ceiling-DSO, au moment où les estimations de mouvement ont commencé à se dégrader.

Ce problème est intrinsèque aux approches visuelles, et surtout celles basées sur des méthodes directes. Nous pouvons aborder ce problème, par exemple, en réinitialisant la Ceiling-DSO dès qu'on détecte une importante divergence par rapport aux autres modalités. La réinitialisation peut s'avérer utile pour redémarrer le processus d'estimation sans prendre en compte les derniers points. En revanche, si le robot reste toujours sous une zone de plafond avec des textures alignées sur une seule direction (problème d'ouverture), la nouvelle estimation risque de diverger elle aussi.

Il serait difficile donc de traiter ce problème sans utiliser d'autres modalités dans une stratégie de fusion multicapteur (odométrie des roues, IMU, *etc.*). Ainsi, l'intégration des données d'autres capteurs peut être utilisée pour réaliser une pondération des points détectés, donnant des poids importants aux points qui suivent le sens de mouvement du robot tel qu'estimé par l'autre modalité. Cela pourrait permettre d'améliorer le suivi des points dans les zones à faible nombre de points utiles. Par ailleurs, l'utilisation d'une autre modalité permettrait de continuer à estimer les mouvements du robot dans les zones problématiques.

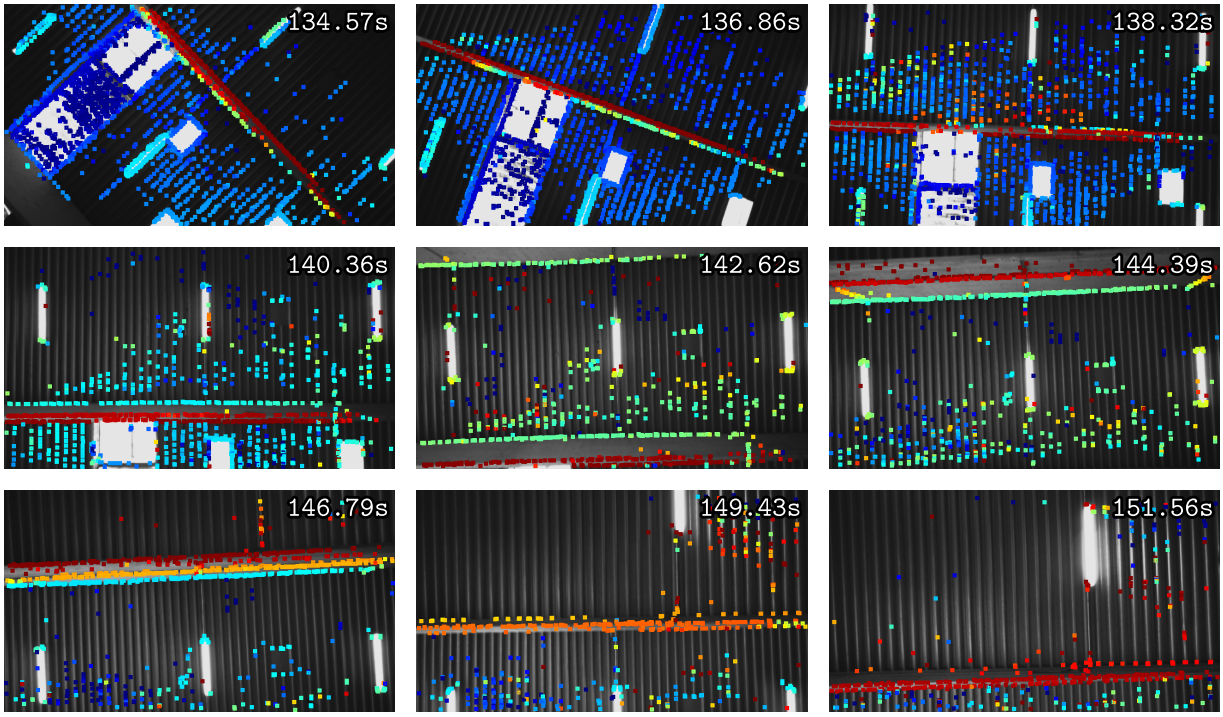


Figure 5.33 – Des images clés utilisées lors de l'estimation de la trajectoire sur la séquence 5, sur la zone problèmatique. Les points suivis sont visualisés sur les images. L'alignement de la trajectoire avec la majorité des motifs du plafond a causé l'échec de l'estimation à cause du problème de l'ouverture.

5.9 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les jeux de données destinés à l'évaluation des systèmes vSLAM et VO en environnements indoor. Ensuite, nous avons présenté nos travaux centrés sur la conception et la collecte d'un jeu de données multicapteur centré sur la vision verticale (du plafond). En effet, ce travail est motivé par le manque d'un jeu de données incluent des images acquises par une caméra orientée vers le plafond, notamment dans des environnements industriels réels.

Nous avons donc instrumenté une plateforme robotique en l'équipant de plusieurs capteurs. Cela nous a permis de réaliser, dans un environnement industriel réel, des séquences de données multicapteurs synchronisées temporellement dans le but de proposer un outil de validation, d'évaluation et de comparaison pour nos futurs travaux.

Dans ce chapitre, nous avons présenté la plateforme expérimentale utilisée, ainsi que l'ensemble des capteurs avec leurs caractéristiques et leurs informations de calibrage. Nous avons également présenté des trajectoires de référence estimées à l'aide de l'outil de photogrammétrie de calcul exhaustif COLMAP et mises à l'échelle grâce aux trajectoires estimées à partir du LiDAR. Enfin, nous avons exploité ce jeu de données pour valider le système Ceiling-DSO présenté précédemment dans le chapitre 4.

Ce jeu de données nous a permis d'identifier des cas extrêmes où le système Ceiling-DSO a échoué dans l'estimation des mouvements. Ainsi, nous avons discuté dans ce chapitre la cause de cet échec, à savoir, le problème d'ouverture *(apperture problem)*. En effet, le

deuxième entrepôt comportait des zones du plafond qui présentent un alignement majoritaire des motifs dans le sens du mouvement, ce qui a rendu difficile l'estimation des mouvements du robot. Ceci stimule la réflexion sur des futurs travaux, en particulier, ceux en relation avec l'intégration d'autres modalités dans le système proposé afin d'améliorer sa robustesse.

Ce travail fera prochainement sujet d'une publication dans un journal international avec la mise à disposition du jeu de données pour la communauté scientifique. Ainsi, ce travail peut contribuer à faciliter l'évaluation, le benchmark, et la comparaison des approches de localisation à base de vision verticale, horizontale, LiDAR ou à base de fusion multicapteur.

CHAPITRE 6

Conclusion générale

6.1 Bilan

Ce travail a été mené dans le cadre d'une *Convention industrielle de formation par la recherche (Cifre)* en partenariat entre le laboratoire des *Systèmes et applications des technologies de l'information et de l'énergie (SATIE)* et l'entreprise *ez-Wheel*. Cette dernière se spécialise dans le développement des solutions compactes et intégrées pour la motorisation des plateformes mobiles destinées au déplacement des lourdes charges, telles que des chariots, des plateformes mobiles d'usines, des dispositifs médicaux, et bien d'autres. Ce travail s'inscrit dans la perspective de la transformation robotique de l'entreprise ez-Wheel, qui a récemment développé une nouvelle gamme de produits baptisée *SWD®* pour *Safety Wheel Drive*, conçue spécifiquement pour des applications en robotique mobile sûre.

Dans cette optique, l'objectif de cette thèse est d'étudier, d'expérimenter et de proposer une solution de localisation adaptée aux environnements industriels indoor, atteignant une maturité technologique de niveau 6 (TRL 6) qui correspond à *un système prototype validé en conditions réelles*.

Le contexte industriel de la thèse exige des contraintes importantes en termes de coût et de caractéristiques du système visé. En effet, le cahier des charges initial stipule que la solution proposée doit être *non intrusive* dans le sens où elle *ne doit nécessiter ni une infrastructure dédiée (infrastructure-less) ni une modification de l'environnement*. De plus, cette solution doit être d'un *faible coût* et viable économiquement, permettant ainsi de rendre *plus abordable* l'automatisation des usines et des entrepôts *pour les petites et moyennes entreprises*.

Dans un premier temps, une étude bibliographique et une veille industrielle nous a permis de recenser les méthodes, les techniques et les technologies utilisées dans le cadre des solutions de localisation en environnements indoor. Les résultats de cette étude ont été analysés en

tenant compte du contexte industriel spécifique de la thèse, ainsi qu'aux contraintes imposées par l'entreprise partenaire. Ces éléments ont joué un rôle essentiel dans l'orientation de nos travaux et dans la sélection des choix technologiques adoptés.

Nos premières expérimentations ont porté sur la conception et la validation d'une plateforme robotique à base de roues motorisées de la gamme SWD® de l'entreprise ez-Wheel. Cette plateforme expérimentale, dénommée « SmartTrolley », a été développée pour démontrer la faisabilité d'un système robotisé modulaire pouvant être intégré dans des plateformes standards de type chariots de lourdes charges. La plateforme est équipée de codeurs de roues et de deux LiDARs couvrant ensemble un champ de vision de 360°.

Nous avons proposé une approche de localisation relative en utilisant les données des deux LiDARs, combinées avec l'odométrie à l'aide d'un processus de fusion à base du filtre de Kalman étendu (EKF). Pour cela, nous avons utilisé l'algorithme d'appariement des données de LiDAR *(Scan Matching)* dénommé NDT-PSO.

Étant probabiliste, l'approche proposée peut permettre de gérer, sans traitement supplémentaire, les petites perturbations engendrées par les objets mobiles. Malgré que les résultats préliminaires de cette solution étaient prometteurs, nous avons très vite identifié les limites de notre approche. En effet, cette première expérimentation a ouvert la voie à de nouvelles questions concernant les environnements dynamiques. D'autant plus que notre travail vise une application en environnements industriels, il est inévitable de tenir compte de la présence d'autres robots, d'humains ou d'engins mobiles dans cet environnement.

La littérature scientifique aborde la question des objets dynamiques principalement par le biais de la détection et du filtrage d'objets mobiles. Dans cette approche, un sous-système de perception est chargé de détecter les objets mobiles présents dans la scène, et puis de les exclure des données utilisées pour la localisation (filtrage des amers et des points repères). Cependant, la mise en œuvre de telles approches s'avère très compliquée, car elles ajoutent des couches de complexité supplémentaires au système de localisation, tout en exigeant généralement une importante puissance de calcul.

Pour résoudre cette problématique, nous avons proposé une approche alternative qui consiste à séparer l'espace de navigation (le plan horizontal utilisé pour la détection des obstacles) de l'espace de localisation (le plan vertical, *c.-à-d.*, le plafond, utilisé pour la détection des amers et le suivi des mouvements du robot).

Ainsi, notre approche repose sur l'utilisation d'une caméra orientée vers le plafond (que nous appelons *caméra verticale* ou *Ceiling-Camera*), tel que nous l'illustrons dans la figure 6.1. Le système de localisation utilise les images capturées par cette caméra pour effectuer une localisation de manière incrémentale ou absolue.

Dans le chapitre 4 dédié au concept de l'odométrie visuelle verticale, nous avons présenté d'abord les outils mathématiques nécessaires pour la compréhension de la vision par ordinateur et les principales approches de conception des systèmes VO. Par la suite, nous avons dressé un état de l'art des méthodes de localisation par vision verticale (caméra orientée vers le plafond). Suite à cette étude de la littérature scientifique, nous avons constaté que les méthodes existantes font souvent des hypothèses fortes concernant la forme du plafond et les types d'objets observables sur ce dernier. Or, les hypothèses de ces méthodes ne sont pas satisfaisables dans les plafonds complexes comme ceux des environnements industriels que nous ciblons.

Nous avons ensuite discuté de notre choix d'utiliser la méthode DSO *(Direct Sparse Odometry)* pour mettre en œuvre une stratégie de localisation incrémentale basée sur la vision au plafond. L'utilisation d'une méthode générique, telle que la DSO, nous permet de proposer une solution polyvalente qui s'adapte aux différents types de plafonds, sans être limitée par des hypothèses préalables sur leurs formes et/ou sur les objets présents sur ces derniers.

Figure 6.1 – Illustration du concepte de la séparation de l'espace de navigation (détection d'obstacles via un LiDAR sur le plan horizontal jaune) de l'espace de localisation (détection d'amers via une caméra sur le plan vertical rouge).

Afin de valider notre approche dénommée Ceiling-DSO, nous avons mené plusieurs expériences dans une grande salle au sein de l'entreprise ez-Wheel. Ces essais ont permis valider le système dans un environnement de laboratoire, atteignant ainsi un niveau 5 de maturité technologique (TRL 5). Les résultats que nous avons présentés dans le chapitre 4 ont porté aussi sur l'identification de l'influence des paramètres de la DSO sur les sorties de l'estimation, dans le but de trouver la combinaison optimale des paramètres qui répond à la fois aux contraintes du temps réel et de la précision. Ainsi, les erreurs relatives atteintes dans les cas optimaux étaient en dessous des 20cm.

Pour pouvoir valider le système dans des conditions opérationnelles et dans un véritable environnement industriel (TRL 6), il était nécessaire d'avoir accès à un jeu de données visuelles acquis dans ce type d'environnements, et qui comprend les données acquises à l'aide d'une caméra orientée vers le plafond. Bien qu'il existe plusieurs jeux de données pour la localisation visuelle, seul un jeu, à savoir le Hilti-Oxford, contenait des images prises par une caméra verticale, bien que ce ne soit pas son objectif principal. En revanche, les données de ce jeu n'ont pas été acquises dans des environnements industriels, elles ne sont donc pas très pertinentes pour notre cas d'étude.

Après avoir examiné les travaux les plus récents pour étudier les différents jeux de données disponibles, nous avons constaté qu'un jeu de données multicapteur qui se concentre sur la vision verticale n'était pas encore disponible.

Par conséquent, nous avons conçu et réalisé une expérience pour l'acquisition d'un jeu de données multicapteur centré sur la vision verticale. Pour ce faire, nous avons conçu une plateforme à base des roues SWD® d'ez-Wheel, que nous avons instrumenté en intégrant plusieurs capteurs, comprenant :

- Une caméra stéréoscopique orientée vers l'avant ;
- Une caméra stéréoscopique orientée vers le plafond ;
- Un LiDAR bidimensionnel (2D) ;
- Deux codeurs de roues ;
- Deux capteurs de flux optique orientés vers le sol.

En utilisant notre plateforme expérimentale, nous avons réalisé six séquences représentant des trajectoires complexes. Ces données ont été collectées dans deux entrepôts réels de l'entreprise française Provost, offrant ainsi un jeu de données récolté dans un environnement industriel authentique.

En plus de présenter notre plateforme expérimentale et de réaliser un état de l'art des jeux de données visuelles, le chapitre 5 présente également un bref état de l'art sur les méthodes de génération des vérités de terrain.

En effet, pour fournir une vérité terrain avec notre jeu de données, nous avons opté pour la génération de trajectoires de référence par un calcul exhaustif basé sur *l'ajustement de faisceaux (BA - Bundle Adjustment)*. Ce choix s'explique par l'absence dans notre cas d'un système de localisation précis en indoor, tel qu'un système de *capture de mouvement (MoCap)*. Ainsi, pour générer la vérité terrain, nous avons utilisé l'outil de photogrammétrie COLMAP, qui permet de reconstruire une représentation tridimensionnelle de l'environnement à partir d'un flux d'images en utilisant la technique de la *structure à partir du mouvement (SfM - Structure from Motion)*. Cette approche nous a permis d'estimer des trajectoires de référence pour les six séquences enregistrées. Par la suite, ces trajectoires ont été mises à l'échelle métrique en utilisant une carte générée à partir des données du LiDAR à l'aide de la méthode LaMa SLAM.

À la fin du chapitre 5, nous avons utilisé les données récoltées valider la Ceiling-DSO dans des scénarios réels. Ainsi, une partie des résultats venait confirmer les conclusions du chapitre 4. En effet, les erreurs relatives enregistrées sur les séquences de 1 à 4 ont été minimes, restant sous la barre des 10cm sur des trajectoires allant de 88m à 142m. En revanche, sur les séquences 5 et 6 enregistrées dans un autre entrepôt, les résultats n'ont pas été à cette hauteur. Dans un premier temps, nous avons soupçonné que cela venait des conditions d'éclairage dans cet entrepôt. Cependant, nous nous sommes rendus compte que l'échec de l'estimation sur ces deux séquences était plutôt causé par le *problème d'ouverture*, une autre limitation classique des méthodes de vision, notamment celles basées sur des formulations directes de type *flux optique*. En effet, l'analyse de la séquence d'images au moment de l'échec de l'estimation montrait clairement des zones avec des textures alignées principalement sur la direction du mouvement. Ce cas de figure marginal stimule des réflexions sur les futurs travaux et sur les améliorations que nous devons apporter à notre système, en particulier, celles en relation avec l'intégration d'autres modalités dans le système pour améliorer sa robustesse. D'autant plus que la présence de cas extrêmes comme celui-ci dans un jeu de données pourrait construire un scénario de test réaliste et intéressant pour la communauté scientifique.

En conclusion, ce travail a permis d'établir une étude approfondie sur les systèmes de localisation en environnements indoor, en s'intéressant particulièrement aux environnements industriels. De plus, il a permis de proposer une approche innovante de localisation adaptée aux environnements industriels dynamiques, en utilisant la vision verticale. Les expérimentations réalisées ont démontré la faisabilité de cette approche, avec des résultats prometteurs atteignant un niveau 6 de maturité technologique (TRL 6).

6.2 Perspectives

Les travaux effectués dans le cadre de cette thèse ouvrent la voie à de nombreuses perspectives de recherche et de développement. En effet, la méthode proposée, étant à base d'une caméra monoculaire, permet d'estimer les mouvements à une échelle près. Des travaux sont actuellement en cours pour permettre l'estimation des mouvements à l'échelle métrique à base d'une fusion avec d'autres modalités telles que les données inertielles. L'annexe A présente brièvement le problème de l'estimation de l'échelle métrique ainsi qu'un état de l'art préliminaire sur ce sujet. En plus, les mêmes modalités supplémentaires qui seront utilisées pour l'estimation de l'échelle pourront être exploitées aussi pour proposer des solutions au problème d'ouverture que nous avons identifié sur quelques séquences.

Une autre possibilité de développement consisterait à intégrer des stratégies de fermeture de boucle et de gestion de carte au système d'odométrie visuelle conçu, dans le but de concevoir un système SLAM complet. Cela permettrait d'améliorer la cohérence globale des trajectoires estimées, en particulier dans des environnements étendus. De plus, une gestion efficace de la carte permettrait une réutilisation aisée de celle-ci à des fins de localisation.

Un autre axe futur porterait sur l'amélioration de notre implémentation, notamment en portant les logiciels réalisés sur des plateformes à base de processeur ARM Cortex-A avec GPU. Ainsi, des travaux d'adéquation algorithme-architecture (AAA) doivent être réalisés afin de permettre la meilleure exploitation des ressources matérielles des plateformes ciblées, tout en optimisant l'implémentation pour minimiser le temps de calcul et la consommation de ces ressources.

Enfin, il serait essentiel de valoriser nos travaux en intégrant le système proposé dans des produits finaux.

ANNEXE A

Estimation de l'échelle métrique pour la VO : un état de l'art

A.1 Introduction

Comme nous l'avons évoqué dans le chapitre 4, l'usage d'une caméra monoculaire pour l'estimation des mouvements souffre intrinsèquement du *problème de l'ambiguïté de l'échelle métrique*. Et même s'il existe des stratégies permettant d'initialiser l'échelle métrique et de continuer à l'estimer au fil du temps, les méthodes monoculaires qui font cela sont souvent sujettes au problème de la *dérive de l'échelle*.

Dans les algorithmes de localisation monoculaire, la pose est généralement exprimée sous forme d'une similarité tridimensionnelle $\mathbf{x} \in \mathrm{Sim}(3)$. Cela veut dire, qu'en plus des 6 degrés de liberté (3 rotations et 3 translations), la pose estimée par une méthode monoculaire possède un septième degré de liberté, à savoir, l'échelle métrique (figure A.1). Ainsi, afin de pouvoir estimer l'échelle, cette dernière doit être ajoutée aux paramètres à estimer. En revanche, la caméra ne peut pas observer l'échelle métrique directement, cela impose donc l'utilisation d'une autre stratégie pour mettre à jour cette mesure.

Dans le cadre de la poursuite de nos travaux, nous présentons dans cette annexe, le problème de l'estimation de l'échelle métrique dans les systèmes d'odométrie visuelle monoculaire. Ainsi, cette annexe comportera un état de l'art préliminaire présentant les différentes approches proposées dans la littérature scientifique pour résoudre ce problème.

A.2 Vision monoculaire vs. vision stéréoscopique

Dans les systèmes de VO stéréoscopique, l'échelle des mouvements peut être récupérée à partir de l'observation de l'effet de la parallaxe entre les images acquises des deux caméras. Ainsi, en connaissant *l'entraxe (baseline)* qui sépare les deux capteurs, les profondeurs des points pourront être estimées. Cela se fait par le calcul de la carte de *disparité* entre les

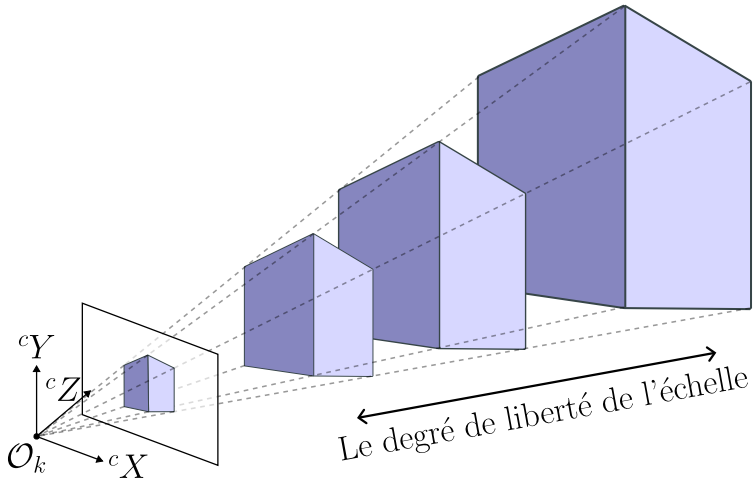


Figure A.1 – Illustration du problème de l'ambiguïté sur l'échelle dans la vision monoculaire. Nous ne pouvons pas déduire la distance caméra-objet d'une manière directe à partir d'une image monoculaire. Il y a une infinité de valeurs possibles, et donc un degré de liberté additionnel.

deux images. Les objets proches de la caméra produisent des larges disparités, tandis que les objets lointains produisent des disparités plus serrées. Par conséquent, une telle caméra peut être utilisée pour l'estimation des mouvements à l'échelle métrique [55].

Néanmoins, comme la disparité est inversement proportionnelle à la profondeur des objets observés, la vision stéréoscopique devient inefficace lorsque la distance entre la scène et la caméra est beaucoup plus grande que l'entraxe des deux caméras. Dans ce cas, les mesures d'une caméra stéréoscopique se dégradent vers des mesures monoculaires [126].

Ainsi, dans le cadre de notre travail, nous considérons que les caméras stéréoscopiques ne sont pas appropriées à notre problématique. En effet, pour une méthode de vision verticale, les plafonds des environnements industriels sont souvent très hauts, généralement entre 6 mètres jusqu'à plus de 15 mètres dans certains cas. Or pour les caméras stéréoscopiques du commerce, les deux capteurs ne sont séparés que d'une distance de 5 à 10cm, donnant une estimation des profondeurs entre 20cm à 5m, avec une fiabilité qui se dégrade en s'éloignant de la caméra.

A.3 Estimation de l'échelle métrique en vision monoculaire

Pour résoudre le problème de l'échelle métrique dans les méthodes visuelles à caméras monoculaires, plusieurs approches ont été proposées dans la littérature scientifique. En effet, l'ambiguïté de l'échelle est insoluble lorsque le mouvement de la caméra n'est pas contraint [118]. Par conséquent, l'échelle absolue peut être déterminée à partir de *mesures directes* (par exemple, la taille d'un objet dans la scène), *des contraintes de mouvement* ou du *couplage avec d'autres capteurs*, tels que l'unité de mesure inertielle (IMU) et les capteurs de distance.

Ainsi, ces approches suivent généralement deux stratégies, soit ils incorporent des *connaissances préalables* dans l'algorithme, soit ils exploitent le *couplage d'autres modalités* (autres capteurs) qui permettent l'observation de l'échelle métrique.

A.3.1 Intégration de connaissances préalables

Une approche possible est d'estimer l'échelle au démarrage à l'aide d'un objet connu dans l'environnement, puis d'utiliser cette valeur tout au long de l'estimation des mouvements de la caméra.

Cette technique a été utilisée dans MonoSLAM [66] où l'algorithme est aidé au démarrage avec une petite quantité d'informations préalables sur la scène en observant une cible de forme et dimensions connues placée devant la caméra (figure A.2). Ceci permet l'estimation de l'échelle métrique au démarrage et aussi d'être sûr d'avoir des caractéristiques observables dès le démarrage du système.

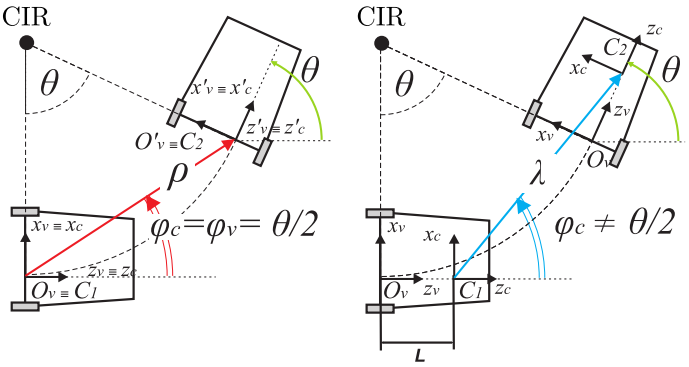
Figure A.2 – Dans la MonoSLAM, une mise en correspondance de quatre caractéristiques connues est réalisée à l'initialisation. Les carrés représentent les caractéristiques détectées et les cercles reflètent l'incertitude attribuée à l'estimation de la position de départ de la caméra (tirée de [66])

D'autres approches utilisent une information préalable sur la structure du robot qui est indirectement observable par la caméra. Zhou et al. [166] ont exploité la connaissance de la hauteur de la caméra pour estimer l'échelle métrique. En utilisant une caméra pointée vers l'avant d'un véhicule, l'approche de Zhou et al. consiste à extraire le plan du sol (figure A.3b), calculer sa normale et puis utiliser la hauteur connue de la caméra suivant la normale du plan du sol pour mettre toutes les mesures à l'échelle (figure A.3a). D'autres méthodes de ce type ont été proposées dans la littérature scientifique [167, 168].

Une autre approche similaire a été proposée par Scaramuzza et al. [169], qui ont exploité le fait que, pour les caméras montées sur des robots mobiles de type véhicule, la construction de ces derniers impose des contraintes intéressantes sur le mouvement de la caméra, appelées *contraintes non holonomes*. Ainsi, ils ont placé la caméra décalée d'une distance connue du centre de rotation du véhicule (figure A.4), et ils ont montré que l'échelle métrique pourrait être estimée en connaissant cette distance et le modèle cinématique du véhicule, en l'occurrence, un modèle d'Ackermann.

(a) Géométrie du plan au sol **(b)** La zone d'intérêt avec les caractéristiques extraites

Figure A.3 – (a) Géométrie du plan au sol, où $\vec{n}$ est la normale du plan de la route et h est la hauteur de la caméra. (b) Caractéristiques extraites d'une zone d'intérêt prédéfinie, les points verts et les rouges représentent les *inliers* et les *outliers* avec ajustement d'homographie (adaptée de [166])



(a) $L = 0$ (b) $L \neq 0$

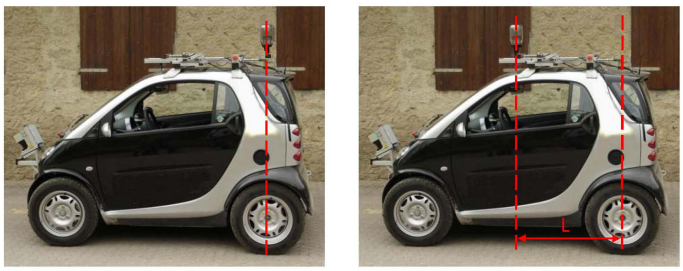


Figure A.4 – Le principe utilisé pour estimer l'échelle à partir des contraintes non holonomes dans un robot de type véhicule (modèle d'ACKERMANN). La figure (a) à gauche illustre le cas où la caméra est placée sur l'axe de rotation ($L = 0$), la direction de la translation du véhicule et de la caméra sont égaux $\varphi_v = \varphi_c = \frac{\theta}{2}$. À droite (b) la configuration utilisée par SCARAMUZZA et al., la caméra est écartée de l'axe de rotation d'une distance $L \neq 0$ connue, ce qui fait que la direction de translation de la caméra est différente de celle du véhicule ($\varphi_c \neq \varphi_v$). Cette contrainte est utilisée pour estimer l'échelle d'un mouvement mesuré par la caméra à l'aide de la distance connue L (tirée de [169])

A.3.2 Fusion multicapteur

Les mouvements de la caméra et la géométrie de la scène estimés dans l'odométrie visuelle monoculaire sont valables à une échelle métrique près, laissant une ambiguïté sur ce degré de liberté due à la non-observabilité de l'échelle dans les caméras monoculaires. Cette ambiguïté peut être enlevée via le couplage des données de la caméra monoculaire avec des données issues d'une autre modalité (autre capteur) qui permet l'observation directe ou indirecte de l'échelle métrique.

Ce problème a été abordé dans la littérature scientifique via différentes approches, nous explorons ici quelques-unes.

A.3.2.1 Fusion avec les codeurs des roues

Les codeurs des roues permettent de mesurer la vitesse de rotation des roues du robot. Les mesures des codeurs peuvent être transformées en données métriques en connaissant le modèle cinématique du robot et ses paramètres (modèle, distance entre les roues, diamètre des roues, etc.).

Ainsi, la vitesse peut être intégrée au fil du temps pour estimer localement les mouvements du robot d'une manière incrémentale. Ensuite, cette mesure peut être utilisée dans le cadre d'une stratégie de fusion pour permettre l'estimation de l'échelle métrique [170].

A.3.2.2 Fusion avec les données du LiDAR

Les méthodes d'odométrie visuelle couplées avec des LiDARs peuvent être vu sous plusieurs axes selon l'approche étudiée. Ainsi, certaines méthodes se reposent sur l'augmentation des données visuelles par les données du LiDAR en calibrant le système caméra-LiDAR pour associer les caractéristiques présentes dans l'image à des profondeurs métriques estimées par le LiDAR.

Inversement, d'autres méthodes reposent sur l'augmentation des données du LiDAR par les données visuelles, ou les données du LiDAR, souvent tridimensionnelles, sont augmentées par les observations visuelles afin d'identifier, classifier et annoter les objets présents dans la scène.

D'autres méthodes utilisent des stratégies d'estimation concurrente, où la caméra et le LiDARs sont utilisés dans une configuration de fusion complémentaire. Dans ce cas, la trajectoire estimée par le LiDAR peut être utilisée, par exemple, pour corriger l'échelle de la trajectoire estimée par la caméra.

Un exemple de cette famille d'approches peut être trouvé dans la V-LOAM, une méthode de localisation qui combine les estimations de l'odométrie visuelle et celles du LiDAR. Dans la V-LOAM, les estimations du mouvement obtenues par odométrie visuelle sont raffinées à une basse fréquence (1Hz) en utilisant une mise en correspondances sur les données du LiDAR. Cependant, la configuration utilisée pour évaluer la V-LOAM comprend un LiDAR 2D monté sur une unité d'inclinaison *(Pan-tilt unit)* afin d'obtenir des informations 3D. Ceci demande donc que les points de chaque balayage du LiDAR soient rectifiés à l'aide d'une hypothèse sur le mouvement du robot (un modèle cinématique et/ou dynamique) [171].

A.3.2.3 Fusion avec les mesures inertielles

L'*unité de mesures inertielles (IMU - Inertial Measurement Unit)* est un composant électronique qui contient un accéléromètre qui mesure l'accélération, un gyromètre qui mesure les vitesses angulaires, et parfois un magnétomètre qui mesure l'orientation par rapport au champ magnétique terrestre (nord magnétique).

Bien que l'accéléromètre et le gyromètre sont des capteurs proprioceptifs, le magnétomètre est un capteur extéroceptif, car il mesure une grandeur externe. En plus, pour les robots qui évoluent dans des environnements indoor, le magnétomètre est inutilisable dans la plupart des cas, notamment dans les environnements industriels où le bruit électromagnétique généré par les machines cause des perturbations permanentes des mesures du magnétomètre. À cause de ça, le magnétomètre est généralement exclu des capteurs utilisés pour la perception en environnement indoor.

L'IMU est un capteur intéressant pour l'odométrie visuelle, car il rend l'échelle métrique et la gravité observables [172].

Les approches de l'odométrie visuelle-inertielle suivent trois paradigmes principaux, à savoir :

- Les méthodes de filtrage (*Filtering methods*) ;
- Les lisseurs à retard fixe (*Fixed-lag smoothers*) ;
- Les lisseurs complets (*Full smoothers*).

Les méthodes de filtrage Dans ces méthodes, l'estimation est réalisée en limitant le processus d'inférence au dernier état du système. Les approches classiques de filtrage estiment à la fois les poses de la caméra et les positions d'un ensemble de caractéristiques prises comme des repères dans l'environnement.

Dans ce cas, le filtre (par exemple, le filtre de Kalman étendu) est indépendant de la quantité de données du capteur traitées. En revanche, la complexité du filtre augmente d'une façon quadratique avec nombre de repères estimés. Par conséquent, un petit nombre de repères sont généralement suivis pour permettre un fonctionnement efficace en temps réel [172].

Les lisseurs à retard fixe Appelés aussi *estimateurs à fenêtre glissante*, elles sont des méthodes qui estiment les états du système dans une fenêtre temporelle donnée (les N derniers états), tout en marginalisant les états les plus anciens. La stratégie de sélection de nombre d'états à traiter varie d'une méthode à une autre, certaines méthodes utilisent un nombre fixe d'images, d'autres utilisent des critères plus sophistiqués pour la sélection (par exemple, la marginalisation des états liés à des caractéristiques dès que ces dernières ne sont plus observables).

L'odométrie visuelle est un problème hautement non-linéaire, les approches de lissage à retard fixe sont souvent plus précises que le filtrage dans ce cas, car elles utilisent plusieurs mesures passées lors de la mise à jour de l'estimation actuelle. En plus, ces approches peuvent être rendues plus robustes aux valeurs aberrantes en rejetant explicitement les valeurs aberrantes après l'optimisation ou en utilisant des fonctions de coût robustes. Cependant, comme les lisseurs à décalage fixe ont toujours recours à la marginalisation, ils peuvent souffrir, comme les filtres, d'incohérences et d'erreurs de linéarisation [172].

Les lisseurs complets Appelés aussi méthodes basées sur des algorithmes des *moindres carrés non-linéaires* sont des méthodes qui estiment l'historique complet des états en les intégrant dans un grand système d'optimisation non-linéaire.

Le lissage complet garantit une plus grande précision, car il permet de mettre à jour le point de linéarisation de tous les états au fur et à mesure de l'évolution de l'estimation. Toutefois, comme la complexité du problème d'optimisation est approximativement cubique par rapport à la dimension des états, le fonctionnement d'une telle approche en temps réel devient rapidement irréalisable à mesure que la trajectoire et la carte grandissent avec le temps.

La pratique courante (utilisée également dans les lisseurs à décalage fixe) est de sélectionner et conserver seulement un sous ensemble des images, appelées *images clés (keyframes)*, ce qui permet de réduire le nombre de paramètres à optimiser, et par conséquent, réduire la complexité du problème d'optimisation.

Un autre problème qui peut affecter à la fois les lisseurs complets et les lisseurs à retard fixe est les différences du temps d'échantillonnage entre la caméra et l'IMU. Les méthodes de filtrage utilisent généralement l'IMU pour le modèle de prédiction et la caméra pour le modèle de mesure (mise à jour), et, par conséquent, elles gèrent naturellement les différents taux d'échantillonnage. En revanche, dans les approches de lissage, il est impossible pour les systèmes temps-réel d'ajouter un état à chaque mesure de l'IMU, car la complexité d'un tel problème augmente avec la dimensionnalité des états. Ainsi, les mesures de l'IMU sont habituellement préintégrées entre les images clés successives pour former des contraintes sur le mouvement relatif [172].

A.3.2.4 Autres approches

Le problème d'estimation de l'échelle métrique a été abordé via d'autres approches, notamment des approches requirent une infrastructure externe, tel que l'utilisation d'une infrastructure à base de *Ultrawide-Band (UWB)* pour apporter des corrections à l'échelle [173]. Cependant, nous n'avons pas exploré cette famille d'approches vu qu'elles sont intrusives.

A.4 L'odométrie visuelle-inertielle (VIO)

Selon la stratégie utilisée pour fusionner les mesures visuelles et inertielles, les approches de VIO peuvent être classées en deux catégories : les approches *faiblement couplées (Loosely Coupled)* et les approches *étroitement couplées (Tightly Coupled)*.

La figure A.5 illustre ces deux catégories. Conceptuellement, les méthodes faiblement couplées traitent les mesures visuelles et inertielles séparément en calculant deux estimations indépendantes du mouvement qui sont ensuite fusionnées pour obtenir l'estimation finale.

En revanche, les méthodes étroitement couplées calculent l'estimation finale directement à partir des mesures brutes de la caméra et de l'IMU, par exemple, les caractéristiques bidimensionnelles suivies, les vitesses angulaires et les accélérations linéaires. D'une manière générale, les approches étroitement couplées sont plus complexes, mais plus précises que celles faiblement couplées [172].

D'un côté, l'utilisation de la pré-intégration des mesures inertielles pour prédire les emplacements des caractéristiques bidimensionnelles dans l'image suivante peut faciliter le suivi des caractéristiques. En effet, cela permet de diminuer l'espace de recherche et donc de réduire le temps de calcul. En plus, le couplage étroit maintient les corrélations entre toutes les variables d'état et permet ainsi de lier directement des mesures visuelles à des mesures inertielles [174].

D'un autre côté, les approches faiblement couplées ne tiennent pas compte du couplage des informations visuelles et inertielles, ce qui les rend incapables de corriger la dérive dans l'estimateur à base de vision uniquement.

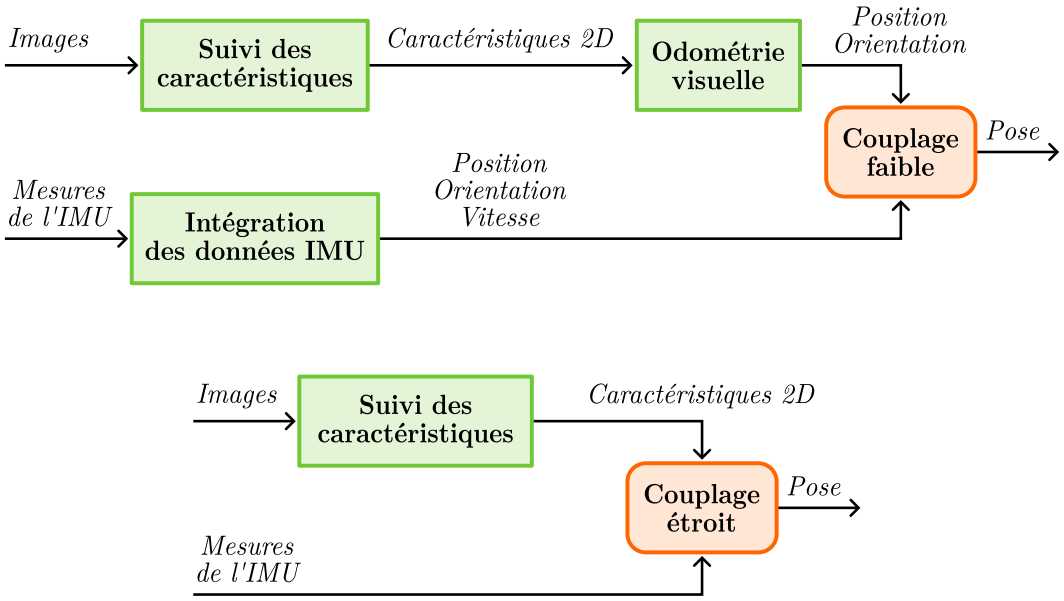


Figure A.5 – Types des systèmes d'odométrie visuelle-inertielle. En haut, un système faiblement couplé *(Loosely Coupled)*, en bas un système étroitement couplé *(Tightly Coupled)* (inspirée de [172]).

A.5 Conclusion

Dans cette annexe, nous avons exposé le problème de l'estimation de l'échelle métrique dans le contexte des approches monoculaires de l'odométrie visuelle. En effet, les trajectoires estimées à l'aide de données provenant d'une caméra monoculaire ne sont valables qu'à une échelle près. Il est ainsi essentiel d'estimer en temps réel l'échelle métrique de ces trajectoires afin de les utiliser de manière efficace dans le cadre du contrôle de mouvement et de la navigation autonome. L'estimation de l'échelle peut également être intéressante pour faciliter la fusion avec d'autres modalités.

Ainsi, nous avons effectué un état de l'art préliminaire sur les différentes approches proposées dans la littérature scientifique pour résoudre ce problème.

Notre analyse s'est concentrée sur la vision verticale *(Ceiling-Vision)* dans le but d'apporter une mise à l'échelle à notre approche Ceiling-DSO, présentée dans le chapitre 4.

Nous avons démontré que l'utilisation de la vision stéréoscopique n'est pas appropriée dans notre contexte. En effet, pour une caméra orientée vers un plafond très haut d'un environnement industriel, les estimations des profondeurs issues d'une caméra stéréoscopique ne seront pas fiables.

Par conséquent, il est plus pertinent dans notre cas de s'appuyer sur la fusion avec une autre modalité telle que l'IMU, les codeurs de roues, *etc.*, afin d'estimer l'échelle métrique de manière plus précise. Les modalités supplémentaires peuvent également être utilisées pour améliorer la robustesse et la fiabilité de l'estimation.

ANNEXE B

Jeu de données : détails techniques

B.1 Structures de données détaillées

B.1.1 La structure de données `sensor_msgs/LaserScan` de ROS

```
std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

B.1.2 La structure `OpticalFlowDuo`

Nous avons défini une nouvelle structure `OpticalFlowDuo`, groupant les données acquises des capteurs de flux optiques, sous la forme suivante :

```
std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
OpticalFlow left
    bool motion_detected
    uint8 surface_quality
```

```
    int16 dx
    int16 dy
    int64 x
    int64 y
OpticalFlow right
    bool motion_detected
    uint8 surface_quality
    int16 dx
    int16 dy
    int64 x
    int64 y
```

B.1.3 La structure `nav_msgs/Odometry` de ROS

```
std_msgs/Header header
    uint32 seq
    time stamp
    string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
    geometry_msgs/Pose pose
        geometry_msgs/Point position
            float64 x
            float64 y
            float64 z
        geometry_msgs/Quaternion orientation
            float64 x
            float64 y
            float64 z
            float64 w
    float64[36] covariance
geometry_msgs/TwistWithCovariance twist
    geometry_msgs/Twist twist
        geometry_msgs/Vector3 linear
            float64 x
            float64 y
            float64 z
        geometry_msgs/Vector3 angular
            float64 x
            float64 y
            float64 z
    float64[36] covariance
```

B.2 Configuration de COLMAP

Lors de la génération des trajectoires de référence présentées dans le chapitre 5, nous avons configuré l'outil de photogrammétrie COLMAP de la manière décrite ci-dessous. À noter que ce format de configuration, n'étant pas standardisé, peut différer d'une version de l'outil à une autre.

```
random_seed=0
[ImageReader]
single_camera=true
single_camera_per_folder=false
single_camera_per_image=false
default_focal_length_factor=1.2
camera_model=PINHOLE
camera_params=425.5406188964844, 425.5406188964844, 425.49615478515625,
↪  237.3509063720703
[SiftExtraction]
use_gpu=true
estimate_affine_shape=false
upright=false
domain_size_pooling=false
num_threads=-1
max_image_size=3200
max_num_features=8192
first_octave=-1
num_octaves=4
octave_resolution=3
max_num_orientations=2
dsp_num_scales=10
peak_threshold=0.0066699999999999997
edge_threshold=10
dsp_min_scale=0.16667000000000001
dsp_max_scale=3
gpu_index=-1
[SiftMatching]
use_gpu=true
cross_check=true
multiple_models=false
guided_matching=false
planar_scene=false
compute_relative_pose=false
num_threads=-1
max_num_matches=32768
max_num_trials=10000
min_num_inliers=15
max_ratio=0.80000000000000004
max_distance=0.69999999999999996
max_error=4
confidence=0.999
min_inlier_ratio=0.25
gpu_index=-1
[SequentialMatching]
quadratic_overlap=true
loop_detection=true
overlap=10
loop_detection_period=10
loop_detection_num_images=50
loop_detection_num_nearest_neighbors=1
loop_detection_num_checks=256
loop_detection_num_images_after_verification=0
```

```
loop_detection_max_num_features=-1
vocab_tree_path=/usr/share/colmap/vocabulary-tree-1M.bin
[BundleAdjustment]
refine_focal_length=true
refine_principal_point=false
refine_extra_params=true
refine_extrinsics=true
max_num_iterations=100
max_linear_solver_iterations=200
function_tolerance=0
gradient_tolerance=0
parameter_tolerance=0
[Mapper]
ignore_watermarks=false
multiple_models=true
extract_colors=true
ba_refine_focal_length=true
ba_refine_principal_point=false
ba_refine_extra_params=true
fix_existing_images=false
tri_ignore_two_view_tracks=true
min_num_matches=15
max_num_models=50
max_model_overlap=20
min_model_size=10
init_image_id1=-1
init_image_id2=-1
init_num_trials=200
num_threads=-1
ba_min_num_residuals_for_multi_threading=50000
ba_local_num_images=6
ba_local_max_num_iterations=25
ba_global_images_freq=500
ba_global_points_freq=250000
ba_global_max_num_iterations=50
ba_global_max_refinements=5
ba_local_max_refinements=2
snapshot_images_freq=0
init_min_num_inliers=100
init_max_reg_trials=2
abs_pose_min_num_inliers=30
max_reg_trials=3
tri_max_transitivity=1
tri_complete_max_transitivity=5
tri_re_max_trials=1
min_focal_length_ratio=0.10000000000000001
max_focal_length_ratio=10
max_extra_param=1
ba_local_function_tolerance=0
ba_global_images_ratio=1.1000000000000001
ba_global_points_ratio=1.1000000000000001
ba_global_function_tolerance=0
ba_global_max_refinement_change=0.00050000000000000001
```

```
ba_local_max_refinement_change=0.001
init_max_error=4
init_max_forward_motion=0.94999999999999996
init_min_tri_angle=16
abs_pose_max_error=12
abs_pose_min_inlier_ratio=0.25
filter_max_reproj_error=4
filter_min_tri_angle=1.5
local_ba_min_tri_angle=6
tri_create_max_angle_error=2
tri_continue_max_angle_error=2
tri_merge_max_reproj_error=4
tri_complete_max_reproj_error=4
tri_re_max_angle_error=5
tri_re_min_ratio=0.20000000000000001
tri_min_angle=1.5
```

Bibliographie

[1] Donald R. Hill. “Al-Jazarī”. In : *Encyclopaedia of the History of Science, Technology, and Medicine in Non-Western Cultures*. Sous la dir. d’Helaine Selin. Dordrecht : Springer Netherlands, 2016, p. 239-240. isbn : 978-94-007-7747-7. doi : 10.1007/978-94-007-7747-7_9612.

[2] Lotfi Romdhane et Saïd Zeghloul. “AL-JAZARI (1136–1206)”. In : *Distinguished Figures in Mechanism and Machine Science : Their Contributions and Legacies, Part 2*. Sous la dir. de Marco Ceccarelli. History of Mechanism and Machine Science. Dordrecht : Springer Netherlands, 2010, p. 1-21. isbn : 978-90-481-2346-9. doi : 10.1007/978-90-481-2346-9_1.

[3] Zekâi Şen. “Ancient Water Robotics and Abou-l Iz Al-Jazari”. In : *Water Supply* 13.3 (1er mai 2013), p. 699-709. issn : 1606-9749. doi : 10.2166/ws.2013.031.

[4] Ibn al-Razzāz Jazarī. “Folio from "Kitab fi ma’arifat al-hiyal al-handisaya", Automata by al-Jazari ; A musical toy in the form of a boat ;” 1206. url : https://asia.si.edu/object/F1930.73 (visité le 28/07/2022).

[5] Leonardo Torres Quevedo. “Ensayos Sobre Automática : Su Definición : Extensión Teórica de Sus Aplicaciones”. In : *Limbo : boletín internacional de estudios sobre Santayana* 17 (2003), p. 9-32.

[6] Eugene Kagan, Nir Shvalb et Irad Ben-Gal, éd. *Autonomous Mobile Robots and Multi-Robot Systems : Motion-Planning, Communication, and Swarming*. Hoboken, NJ : Wiley, 2019. 1 p. isbn : 978-1-119-21317-8.

[7] Karel Čapek. *RUR : Rossum’s universal robots drame collectif en un prologue de comédie en trois actes*. Trad. par Jan Rubeš et Brigitte Munier. Minos 81e. Paris : la Différence, 2011. isbn : 978-2-7291-2350-5.

[8] Saeed B. Niku. *Introduction to Robotics : Analysis, Control, Applications*. Third edition. Hoboken : Wiley, 2019. isbn : 978-1-119-52762-6.

[9] Francisco Rubio, Francisco Valero et Carlos Llopis-Albert. “A Review of Mobile Robots : Concepts, Methods, Theoretical Framework, and Applications”. In : *International Journal of Advanced Robotic Systems* 16.2 (1er mars 2019), p. 1729881419839596. issn : 1729-8806. doi : 10.1177/1729881419839596.

[10] Jr George C. Devol. “Programmed Article Transfer”. Brev. amér. 2988237A. Jr George C Devol. 13 juin 1961. url : https://patents.google.com/patent/US2988237/en (visité le 19/05/2022).

[11] Roland Siegwart, Illah R. Nourbakhsh et Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. 2e éd. The MIT Press, 2011. isbn : 978-0-262-01535-6.

[12] B.Y. Qi, Q.L. Yang et Y. Y. Zhou. “Application of AGV in Intelligent Logistics System”. In : *Fifth Asia International Symposium on Mechatronics (AISM 2015)*. Fifth Asia International Symposium on Mechatronics (AISM 2015). Oct. 2015, p. 1-5. doi : 10.1049/cp.2015.1527.

[13] Johann Borenstein. “The OmniMate : A Guidewire- and Beacon-Free AGV for Highly Reconfigurable Applications”. In : *International Journal of Production Research* 38.9 (1er juin 2000), p. 1993-2010. issn : 0020-7543. doi : 10.1080/002075400188456.

[14] Long Han et al. “System and Design of a Compact and Heavy-Payload AGV System for Flexible Production Line”. In : *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO). Déc. 2013, p. 2482-2488. doi : 10.1109/ROBIO.2013.6739844.

[15] Tim Bailey et Hugh Durrant-Whyte. “Simultaneous Localization and Mapping (SLAM) : Part II”. In : *IEEE Robotics Automation Magazine* 13.3 (sept. 2006), p. 108-117. issn : 1070-9932. doi : 10.1109/MRA.2006.1678144.

[16] Tim Bailey et Hugh Durrant-Whyte. “Simultaneous Localization and Mapping : Part I”. In : *IEEE Robotics Automation Magazine* 13.2 (juin 2006), p. 99-110. issn : 1070-9932. doi : 10.1109/MRA.2006.1638022.

[17] Sebastian Thrun et al. *Probabilistic Robotics*. MIT Press, 19 août 2005. 668 p. ISBN : 978-0-262-20162-9. Google Books : 2Zn6AQAAQBAJ.

[18] John J. Leonard et Hugh F. Durrant-Whyte. "Mobile Robot Localization by Tracking Geometric Beacons". In : *IEEE Transactions on Robotics and Automation* 7.3 (juin 1991), p. 376-382. ISSN : 1042-296X, 2374-958X. DOI : 10.1109/70.88147.

[19] Sebastian Thrun et al. "FastSLAM : An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data". In : *Journal of Machine Learning Research* 4 (19 mai 2004).

[20] Uwe D. Hanebeck et Gunther Schmidt. "Set Theoretic Localization of Fast Mobile Robots Using an Angle Measurement Technique". In : *Proceedings of IEEE International Conference on Robotics and Automation*. Proceedings of IEEE International Conference on Robotics and Automation. T. 2. Avr. 1996, 1387-1394 vol.2. DOI : 10.1109/ROBOT.1996.506900.

[21] Emmanuel Seignez et al. "Experimental Vehicle Localization by Bounded-Error State Estimation Using Interval Analysis". In : *In IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005.

[22] Dieter Fox, Wolfram Burgard et Sebastian Thrun. "Markov Localization for Mobile Robots in Dynamic Environments". In : *Journal of Artificial Intelligence Research* 11 (23 nov. 1999), p. 391-427. ISSN : 1076-9757. DOI : 10.1613/jair.616.

[23] Frank Dellaert et al. "Monte Carlo Localization for Mobile Robots". In : *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). T. 2. Mai 1999, 1322-1328 vol.2. DOI : 10.1109/ROBOT.1999.772544.

[24] Lei Zhang, René Zapata et Pascal Lépinay. "Self-Adaptive Monte Carlo Localization for Mobile Robots Using Range Finders". In : *Robotica* 30.2 (mars 2012), p. 229-244. ISSN : 1469-8668, 0263-5747. DOI : 10.1017/S0263574711000567.

[25] Dieter Fox et al. "Particle Filters for Mobile Robot Localization". In : *Sequential Monte Carlo Methods in Practice*. Sous la dir. d'Arnaud Doucet, Nando de Freitas et Neil Gordon. Statistics for Engineering and Information Science. New York, NY : Springer, 2001, p. 401-428. ISBN : 978-1-4757-3437-9.

[26] Benoît Desrochers, Simon Lacroix et Luc Jaulin. "Set-Membership Approach to the Kidnapped Robot Problem". In : *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Sept. 2015, p. 3715-3720. DOI : 10.1109/IROS.2015.7353897.

[27] Xiaohan Zhang et al. "Vision-Based Monte Carlo - Kalman Localization in a Known Dynamic Environment". In : *Robotics and Vision 2006 9th International Conference on Control, Automation*. Robotics and Vision 2006 9th International Conference on Control, Automation. Déc. 2006, p. 1-7. DOI : 10.1109/ICARCV.2006.345170.

[28] Linhui Xiao et al. "Dynamic-SLAM : Semantic Monocular Visual Localization and Mapping Based on Deep Learning in Dynamic Environment". In : *Robotics and Autonomous Systems* 117 (1er juill. 2019), p. 1-16. ISSN : 0921-8890. DOI : 10.1016/j.robot.2019.03.012.

[29] Chao Sheng et al. "Dynamic-DSO : Direct Sparse Odometry Using Objects Semantic Information for Dynamic Environments". In : *Applied Sciences* 10.4 (4 jan. 2020), p. 1467. DOI : 10.3390/app10041467.

[30] Wolfram Burgard, Dieter Fox et Sebastian Thrun. "Active Mobile Robot Localization". In : *IJCAI'97 : Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence*. Fifteenth International Joint Conference on Artifical Intelligence. T. 2. 1997, p. 1346-1352.

[31] Gian Luca Mariottini et Stergios I. Roumeliotis. "Active Vision-Based Robot Localization and Navigation in a Visual Memory". In : *2011 IEEE International Conference on Robotics and Automation*. 2011 IEEE International Conference on Robotics and Automation. Mai 2011, p. 6192-6198. DOI : 10.1109/ICRA.2011.5980340.

[32] Javier Correa et Alvaro Soto. "Active Visual Perception for Mobile Robot Localization". In : *Journal of Intelligent and Robotic Systems* 58.3 (1er juin 2010), p. 339-354. ISSN : 1573-0409. DOI : 10.1007/s10846-009-9348-4.

[33] Germán Martín Mendoza-Silva, Joaquín Torres-Sospedra et Joaquín Huerta. "A Meta-Review of Indoor Positioning Systems". In : *Sensors* 19.20 (20 jan. 2019), p. 4507. ISSN : 1424-8220. DOI : 10.3390/s19204507.

[34] A. Sneha et al. "QR Code Based Indoor Navigation System for Attender Robot". In : *EAI Endorsed Transactions on Internet of Things* 6.21 (17 août 2020), p. 165519. ISSN : 2414-1399. DOI : 10.4108/eai.13-7-2018.165519.

[35] Payam Nazemzadeh et al. "Indoor Localization of Mobile Robots Through QR Code Detection and Dead Reckoning Data Fusion". In : *IEEE/ASME Transactions on Mechatronics* 22.6 (déc. 2017), p. 2588-2599. ISSN : 1941-014X. DOI : 10.1109/TMECH.2017.2762598.

[36] Jae Hung Yoo et Ishwar K. Sethi. "Mobile Robot Localization with Multiple Stationary Cameras". In : *Mobile Robots VI*. Mobile Robots VI. T. 1613. SPIE, 14 fév. 1992, p. 155-170. DOI : 10.1117/12.135175.

[37] Konstantinos Delibasis, Vasilios Plagianakos et Ilias Maglogiannis. “Real Time Indoor Robot Localization Using a Stationary Fisheye Camera”. In : (30 sept. 2013). doi : 10.1007/978-3-642-41142-7_25.

[38] Wilson Sakpere, Michael Adeyeye Oshin et Nhlanhla BW Mlitwa. “A State-of-the-Art Survey of Indoor Positioning and Navigation Systems and Technologies”. In : *South African Computer Journal* 29.3 (8 déc. 2017). issn : 2313-7835. doi : 10.18489/sacj.v29i3.452.

[39] Swarun Kumar et al. “Accurate Indoor Localization with Zero Start-up Cost”. In : *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom ’14. New York, NY, USA : Association for Computing Machinery, 7 sept. 2014, p. 483-494. isbn : 978-1-4503-2783-1. doi : 10.1145/2639108.2639142.

[40] Dongchen Ni et al. “UWB Indoor Positioning Application Based on Kalman Filter and 3-D TOA Localization Algorithm”. In : *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. 2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE). Mars 2019, p. 1-6. doi : 10.1109/ATEE.2019.8724907.

[41] Annalisa Milella, Grazia Cicirelli et Arcangelo Distante. “RFID-assisted Mobile Robot System for Mapping and Surveillance of Indoor Environments”. In : *Industrial Robot : An International Journal* (7 mars 2008). issn : 0143-991X. doi : 10.1108/01439910810854638.

[42] Ryota Kimoto et al. “Evaluation of MultiZigLoc : Indoor ZigBee Localization System Using Inter-Channel Characteristics”. In : *2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. 2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU). Oct. 2018, p. 1-6. doi : 10.23919/ICMU.2018.8653263.

[43] Stefan Galler et al. “Combined AOA/TOA UWB Localization”. In : *2007 International Symposium on Communications and Information Technologies*. 2007 International Symposium on Communications and Information Technologies. Oct. 2007, p. 1049-1053. doi : 10.1109/ISCIT.2007.4392171.

[44] Yassen Dobrev et al. “Mobile Robot 6D Pose Estimation Using a Wireless Localization Network”. In : *2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. 2016 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM). Mai 2016, p. 1-4. doi : 10.1109/ICMIM.2016.7533923.

[45] Byoung-Suk Choi et Ju-Jang Lee. “Mobile Robot Localization Scheme Based on RFID and Sonar Fusion System”. In : *2009 IEEE International Symposium on Industrial Electronics*. 2009 IEEE International Symposium on Industrial Electronics. Juill. 2009, p. 1035-1040. doi : 10.1109/ISIE.2009.5221628.

[46] Edmundo Torres-Zapata et al. “Implementation of a VLC-based Indoor Localization System”. In : *Transactions on Emerging Telecommunications Technologies* 30.2 (2019), e3498. issn : 2161-3915. doi : 10.1002/ett.3498.

[47] Weipeng Guan et al. “Indoor Localization System of ROS Mobile Robot Based on Visible Light Communication”. 6 jan. 2020. doi : 10.48550/arXiv.2001.01888. arXiv : 2001.01888 [eess].

[48] Oliver J. Woodman et Robert K. Harle. “Concurrent Scheduling in the Active Bat Location System”. In : *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops). Mars 2010, p. 431-437. doi : 10.1109/PERCOMW.2010.5470631.

[49] Alessio De Angelis et al. “Design and Characterization of a Portable Ultrasonic Indoor 3-D Positioning System”. In : *IEEE Transactions on Instrumentation and Measurement* 64.10 (oct. 2015), p. 2616-2625. issn : 1557-9662. doi : 10.1109/TIM.2015.2427892.

[50] *Marvelmind Indoor Navigation System Operating Manual*. 9 août 2023. url : https://marvelmind.com/pics/marvelmind_navigation_system_manual.pdf (visité le 21/08/2023).

[51] Cagri Kilic et al. “Improved Planetary Rover Inertial Navigation and Wheel Odometry Performance through Periodic Use of Zero-Type Constraints”. In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (nov. 2019), p. 552-559. doi : 10.1109/IROS40897.2019.8967634. arXiv : 1906.08849.

[52] Martin Brossard, Axel Barrau et Silvere Bonnabel. “AI-IMU Dead-Reckoning”. In : *IEEE Transactions on Intelligent Vehicles* (2020), p. 1-1. issn : 2379-8904. doi : 10.1109/TIV.2020.2980758.

[53] Jian Lin et al. “ORB-SLAM, IMU and Wheel Odometry Fusion for Indoor Mobile Robot Localization and Navigation”. In : *Academic Journal of Computing & Information Science* 3.1 (27 avr. 2020). doi : 10.25236/AJCIS.030114.

[54] Michael Bloesch et al. “Iterated Extended Kalman Filter Based Visual-Inertial Odometry Using Direct Photometric Feedback :” in : *The International Journal of Robotics Research* (18 sept. 2017). doi : 10.1177/0278364917728574.

[55] Richard Hartley et Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge, UK ; New York : Cambridge University Press, 2003. 655 p. isbn : 978-0-521-54051-3.

[56] Georges YOUNES, Daniel ASMAR et John ZELEK. "A Unified Formulation for Visual Odometry". In : *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Nov. 2019, p. 6237-6244. DOI : 10.1109/IROS40897.2019.8968440.

[57] P. H. S. TORR et Andrew ZISSERMAN. "Feature Based Methods for Structure and Motion Estimation". In : *Vision Algorithms : Theory and Practice*. Sous la dir. de Bill TRIGGS, Andrew ZISSERMAN et Richard SZELISKI. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2000, p. 278-294. ISBN : 978-3-540-44480-0. DOI : 10.1007/3-540-44480-7_19.

[58] M. IRANI et P. ANANDAN. "About Direct Methods". In : *Vision Algorithms : Theory and Practice*. Sous la dir. de Bill TRIGGS, Andrew ZISSERMAN et Richard SZELISKI. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2000, p. 267-277. ISBN : 978-3-540-44480-0. DOI : 10.1007/3-540-44480-7_18.

[59] David NISTER, Oleg NARODITSKY et James BERGEN. "Visual Odometry". In : *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. T. 1. Juin 2004, p. I-I. DOI : 10.1109/CVPR.2004.1315094.

[60] I. PARRA et al. "Robust Visual Odometry for Vehicle Localization in Urban Environments". In : *Robotica* 28.3 (mai 2010), p. 441-452. ISSN : 1469-8668, 0263-5747. DOI : 10.1017/S026357470900575X.

[61] Houssem Eddine BENSEDDIK, Oualid DJEKOUNE et Mahmoud BELHOCINE. "SIFT and SURF Performance Evaluation for Mobile Robot-Monocular Visual Odometry". In : *Journal of Image and Graphics* (2014), p. 70-76. ISSN : 23013699. DOI : 10.12720/joig.2.1.70-76.

[62] Jakob ENGEL, Vladlen KOLTUN et Daniel CREMERS. "Direct Sparse Odometry". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (mars 2018), p. 611-625. ISSN : 1939-3539. DOI : 10.1109/TPAMI.2017.2658577.

[63] Christian FORSTER, Matia PIZZOLI et Davide SCARAMUZZA. "SVO : Fast Semi-Direct Monocular Visual Odometry". In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). Mai 2014, p. 15-22. DOI : 10.1109/ICRA.2014.6906584.

[64] Xiang GAO et al. "LDSO : Direct Sparse Odometry with Loop Closure". 3 août 2018. DOI : 10.48550/arXiv.1808.01111. arXiv : 1808.01111 [cs].

[65] Andrew J. DAVISON. "Real-Time Simultaneous Localisation and Mapping with a Single Camera". In : *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV '03. Washington, DC, USA : IEEE Computer Society, 2003, p. 1403-. ISBN : 978-0-7695-1950-0.

[66] Andrew J. DAVISON et al. "MonoSLAM : Real-Time Single Camera SLAM". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (juin 2007), p. 1052-1067. ISSN : 0162-8828, 2160-9292, 1939-3539. DOI : 10.1109/TPAMI.2007.1049.

[67] Raúl MUR-ARTAL, J. M. M. MONTIEL et Juan D. TARDÓS. "ORB-SLAM : A Versatile and Accurate Monocular SLAM System". In : *IEEE Transactions on Robotics* 31.5 (oct. 2015), p. 1147-1163. ISSN : 1552-3098, 1941-0468. DOI : 10.1109/TRO.2015.2463671.

[68] Raúl MUR-ARTAL et Juan D. TARDÓS. "ORB-SLAM2 : An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In : *IEEE Transactions on Robotics* 33.5 (oct. 2017), p. 1255-1262. ISSN : 1552-3098, 1941-0468. DOI : 10.1109/TRO.2017.2705103.

[69] Mathieu LABBÉ et François MICHAUD. "RTAB-Map as an Open-Source Lidar and Visual Simultaneous Localization and Mapping Library for Large-Scale and Long-Term Online Operation". In : *Journal of Field Robotics* 36.2 (2019), p. 416-446. ISSN : 1556-4967. DOI : 10.1002/rob.21831.

[70] Antoni ROSINOL et al. "Kimera : An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping". 18 déc. 2019. DOI : 10.48550/arXiv.1910.02490. arXiv : 1910.02490 [cs].

[71] Georg KLEIN et David MURRAY. "Parallel Tracking and Mapping for Small AR Workspaces". In : *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. Nov. 2007, p. 225-234. DOI : 10.1109/ISMAR.2007.4538852.

[72] Jakob ENGEL, Thomas SCHÖPS et Daniel CREMERS. "LSD-SLAM : Large-Scale Direct Monocular SLAM". In : *Computer Vision – ECCV 2014*. Sous la dir. de David FLEET et al. Lecture Notes in Computer Science. Cham : Springer International Publishing, 2014, p. 834-849. ISBN : 978-3-319-10605-2. DOI : 10.1007/978-3-319-10605-2_54.

[73] Guillermo GALLEGO et al. "Event-Based Vision : A Survey". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.1 (8 août 2020), p. 154-180. ISSN : 0162-8828, 2160-9292, 1939-3539. DOI : 10.1109/TPAMI.2020.3008413. arXiv : 1904.08405 [cs].

[74] Tobi Delbrück et al. “Activity-Driven, Event-Based Vision Sensors”. In : *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. Proceedings of 2010 IEEE International Symposium on Circuits and Systems. Mai 2010, p. 2426-2429. doi : 10.1109/ISCAS.2010.5537149.

[75] Andrea Censi et Davide Scaramuzza. “Low-Latency Event-Based Visual Odometry”. In : *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014 IEEE International Conference on Robotics and Automation (ICRA). Mai 2014, p. 703-710. doi : 10.1109/ICRA.2014.6906931.

[76] Jaebum Choi et Markus Maurer. “Hybrid Map-Based SLAM with Rao-Blackwellized Particle Filters”. In : *17th International Conference on Information Fusion (FUSION)*. 17th International Conference on Information Fusion (FUSION). Juill. 2014, p. 1-6.

[77] Huaxin Ye et Charles C Zhou. “A New EKF SLAM Algorithm of Lidar-Based AGV Fused with Bearing Information”. In : *TechConnect Briefs* 4.2018 (13 mai 2018), p. 32-39. issn : 9780998878218.

[78] Peter Biber et Wolfgang Strasser. “The Normal Distributions Transform : A New Approach to Laser Scan Matching”. In : Intelligent Robots and Systems 2003 (IROS 2003). T. 3. IEEE. 2003, p. 2743-2748. doi : 10.1109/IROS.2003.1249285.

[79] Yun-Ting Wang et al. “A Single LiDAR-Based Feature Fusion Indoor Localization Algorithm”. In : *Sensors* 18.4 (4 avr. 2018), p. 1294. doi : 10.3390/s18041294.

[80] Sara Bouraine, Abdelhak Bougouffa et Ouahiba Azouaoui. “Particle Swarm Optimization for Solving a Scan-Matching Problem Based on the Normal Distributions Transform”. In : *Evolutionary Intelligence* 15.1 (3 jan. 2021), p. 683-694. issn : 1864-5917. doi : 10.1007/s12065-020-00545-y.

[81] Alberto Elfes. “Sonar-Based Real-World Mapping and Navigation”. In : *IEEE Journal on Robotics and Automation* 3.3 (juin 1987), p. 249-265. issn : 0882-4967, 2374-8710. doi : 10.1109/JRA.1987.1087096.

[82] Sewan Kim et Younggie Kim. “Robot Localization Using Ultrasonic Sensors”. In : *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). T. 4. Sept. 2004, 3762-3766 vol.4. doi : 10.1109/IROS.2004.1390000.

[83] Emmanuel Seignez et al. “An Experimental Platform for Testing Localization Algorithms”. In : *2006 2nd International Conference on Information Communication Technologies*. 2006 2nd International Conference on Information Communication Technologies. T. 1. Avr. 2006, p. 748-753. doi : 10.1109/ICTTA.2006.1684466.

[84] Franklin White. *Data Fusion Lexicon*. USA : Data Fusion Subpanel of the Joint Directors of Laboratories (JDL) - Technical Panel for C3, 1er oct. 1991. url : https://apps.dtic.mil/sti/citations/ADA529661 (visité le 24/01/2023).

[85] Abdellah Lamallem. “Performance evaluation of a information fusion systems”. Thèse de doct. Université de Grenoble, 17 juill. 2012. url : https://theses.hal.science/tel-00768212 (visité le 22/01/2023).

[86] Henrik Boström et al. *On the Definition of Information Fusion as a Field of Research*. HS-IKI-TR-07-006. University of Skövde, School of Humanities and Informatics : Institutionen för kommunikation och information, 2007, p. 8.

[87] James Llinas et al. “Context and Fusion : Definitions, Terminology”. In : *Context-Enhanced Information Fusion : Boosting Real-World Performance with Domain Knowledge*. Sous la dir. de Lauro Snidaro et al. Advances in Computer Vision and Pattern Recognition. Cham : Springer International Publishing, 2016, p. 3-23. isbn : 978-3-319-28971-7. doi : 10.1007/978-3-319-28971-7_1.

[88] B.V. Dasarathy. “Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications”. In : *Proceedings of the IEEE* 85.1 (jan. 1997), p. 24-38. issn : 1558-2256. doi : 10.1109/5.554206.

[89] Hugh F. Durrant-Whyte. “Sensor Models and Multisensor Integration”. In : *The International Journal of Robotics Research* 7.6 (1er déc. 1988), p. 97-113. issn : 0278-3649. doi : 10.1177/027836498800700608.

[90] Wilfried Elmenreich. “Sensor Fusion in Time-Triggered Systems”. 1er jan. 2002.

[91] Dan Simon. *Optimal State Estimation : Kalman, H∞, and Nonlinear Approaches*. John Wiley & Sons, juin 2006. 552 p. isbn : 978-0-470-04533-6.

[92] Mengshen Yang et al. “Sensors and Sensor Fusion Methodologies for Indoor Odometry : A Review”. In : *Polymers* 14.10 (10 jan. 2022), p. 2019. issn : 2073-4360. doi : 10.3390/polym14102019.

[93] Chee-Yee Chong, Shozo Mori et Donald B. Reid. “Forty Years of Multiple Hypothesis Tracking - A Review of Key Developments”. In : *2018 21st International Conference on Information Fusion (FUSION)*. 2018 21st International Conference on Information Fusion (FUSION). Juill. 2018, p. 452-459. doi : 10.23919/ICIF.2018.8455386.

[94] Abdelhak Bougouffa et al. “SmartTrolley : An Experimental Mobile Platform for Indoor Localization in Warehouses”. In : *2020 3rd International Conference on Robotics, Control and Automation Engineering (RCAE)*. 2020 3rd International Conference on Robotics, Control and Automation Engineering (RCAE). Nov. 2020, p. 108-115. doi : 10.1109/RCAE51546.2020.9294484.

[95] Henrik Andreasson et al. *Sensors for Mobile Robots*. 7 juin 2022. doi : 10.48550/arXiv.2206.03223. arXiv : 2206.03223 [cs]. preprint.

[96] Jaromir Konecny et al. "Scan Matching by Cross-Correlation and Differential Evolution". In : *Electronics* 8.8 (8 août 2019), p. 856. issn : 2079-9292. doi : 10.3390/electronics8080856.

[97] P. J. Besl et N. D. McKay. "A Method for Registration of 3-D Shapes". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (fév. 1992), p. 239-256. issn : 1939-3539. doi : 10.1109/34.121791.

[98] Majd Alshawa. "lCL : Iterative Closest Line A Novel Point Cloud Registration Algorithm Based on Linear Features". In : *Ekscentar* 10 (2007), p. 53-59.

[99] Feng Lu et Evangelos Milios. "Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans". In : *Journal of Intelligent and Robotic Systems* 18.3 (1er mars 1997), p. 249-275. issn : 1573-0409. doi : 10.1023/A:1007957421070.

[100] Martin Lauer, Sascha Lange et Martin Riedmiller. "Calculating the Perfect Match : An Efficient and Accurate Approach for Robot Self-localization". In : *RoboCup 2005 : Robot Soccer World Cup IX*. Sous la dir. d'Ansgar Bredenfeld et al. Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2006, p. 142-153. isbn : 978-3-540-35438-3. doi : 10.1007/11780519_13.

[101] Héber Sobreira et al. "Map-Matching Algorithms for Robot Self-Localization : A Comparison Between Perfect Match, Iterative Closest Point and Normal Distributions Transform". In : *Journal of Intelligent & Robotic Systems* 93.3-4 (mars 2019), p. 533-546. issn : 0921-0296, 1573-0409. doi : 10.1007/s10846-017-0765-5.

[102] Edwin B. Olson. "Real-Time Correlative Scan Matching". In : *2009 IEEE International Conference on Robotics and Automation*. 2009 IEEE International Conference on Robotics and Automation. Mai 2009, p. 4387-4393. doi : 10.1109/ROBOT.2009.5152375.

[103] C.C. Tan, C. Hird et Y. Okada. "Processing of Sound Field Signal of a Constrained Panel by Cross-Correlation". In : *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing*. 1997 International Conference on Information, Communications and Signal Processing. T. 1. IEEE, sept. 1997, 316-320 vol.1. doi : 10.1109/ICICS.1997.647111.

[104] Philipp Vath et Benjamin Ummenhofer. "2D Multi-Resolution Correlative Scan-Matching Using a Polygon-Based Similarity Measurement". In : 2010.

[105] Martin Magnusson, Achim Lilienthal et Tom Duckett. "Scan Registration for Autonomous Mining Vehicles Using 3D-NDT". In : *Journal of Field Robotics* 24.10 (2007), p. 803-827. issn : 1556-4967. doi : 10.1002/rob.20204.

[106] Sara Bouraine, Abdelhak Bougouffa et Ouahiba Azouaoui. "NDT-PSO, a New NDT Based SLAM Approach Using Particle Swarm Optimization". In : *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV). Déc. 2020, p. 321-326. doi : 10.1109/ICARCV50220.2020.9305519.

[107] ez-Wheel. *Ez-Wheel SWD® ROS Controllers*. ez-Wheel, 12 mai 2022. url : https://github.com/ezWheelSAS/swd_ros_controllers (visité le 11/07/2022).

[108] ez-Wheel. *Ez-Wheel SWD® ROS2 Controllers*. ez-Wheel, 8 juill. 2022. url : https://github.com/ezWheelSAS/swd_ros2_controllers (visité le 11/07/2022).

[109] Lauro Ojeda et Johann Borenstein. "Methods for the Reduction of Odometry Errors in Over-Constrained Mobile Robots". In : *Autonomous Robots* 16.3 (mai 2004), p. 273-286. issn : 0929-5593. doi : 10.1023/B:AURO.0000025791.45313.01.

[110] Josep M. Mirats Tur et Carlos Albores Borja. "Outdoor Robot Navigation Based on a Probabilistic Data Fusion Scheme". In : *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct. 2007, p. 3733-3738. doi : 10.1109/IROS.2007.4399108.

[111] James Kennedy et Russell Eberhart. "Particle Swarm Optimization". In : *Proceedings of ICNN'95*. International Conference on Neural Networks. T. 4. Nov. 1995, 1942-1948 vol.4. doi : 10.1109/ICNN.1995.488968.

[112] Hans Moravec et Alberto Elfes. "High Resolution Maps from Wide Angle Sonar". In : *1985 IEEE International Conference on Robotics and Automation Proceedings*. 1985 IEEE International Conference on Robotics and Automation Proceedings. T. 2. Mars 1985, p. 116-121. doi : 10.1109/ROBOT.1985.1087316.

[113] Davide Scaramuzza et Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In : *IEEE Robotics Automation Magazine* 18.4 (déc. 2011), p. 80-92. issn : 1070-9932, 1558-223X. doi : 10.1109/MRA.2011.943233.

[114] Abdelhak Bougouffa et al. "Evaluation of a Novel DSO-based Indoor Ceiling-Vision Odometry System". In : *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV). Déc. 2022, p. 47-53. doi : 10.1109/ICARCV57592.2022.10004272.

[115] Eurico Pedrosa, Artur Pereira et Nuno Lau. “A Non-Linear Least Squares Approach to SLAM Using a Dynamic Likelihood Field”. In : *Journal of Intelligent & Robotic Systems* 93.3-4 (mars 2019), p. 519-532. issn : 0921-0296, 1573-0409. doi : 10.1007/s10846-017-0763-7.

[116] Eurico Pedrosa, Artur Pereira et Nuno Lau. “Fast Grid SLAM Based on Particle Filter with Scan Matching and Multithreading”. In : *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). Ponta Delgada, Portugal : IEEE, avr. 2020, p. 194-199. isbn : 978-1-72817-078-7. doi : 10.1109/ICARSC49921.2020.9096191.

[117] Jan Solem. *Programming Computer Vision with Python : Tools and Algorithms for Analyzing Images*. 1st edition. Sebastopol, CA : O’Reilly Media, 24 juill. 2012. 260 p. isbn : 978-1-4493-1654-9.

[118] Richard Szeliski. *Computer Vision : Algorithms and Applications*. Second edition. Texts in Computer Science. Cham : Springer, 2022. 925 p. isbn : 978-3-030-34372-9 978-3-030-34371-2.

[119] Wolfgang Förstner et Bernhard P. Wrobel. *Photogrammetric Computer Vision : Statistics, Geometry, Orientation and Reconstruction*. 1st ed. 2016. Geometry and Computing 11. Cham : Springer International Publishing : Imprint : Springer, 2016. 1 p. isbn : 978-3-319-11550-4. doi : 10.1007/978-3-319-11550-4.

[120] Jakob Engel, Vladyslav Usenko et Daniel Cremers. “A Photometrically Calibrated Benchmark For Monocular Visual Odometry”. 8 oct. 2016. doi : 10.48550/arXiv.1607.02555. arXiv : 1607.02555 [cs].

[121] Joan Solà, Jeremie Deray et Dinesh Atchuthan. “A Micro Lie Theory for State Estimation in Robotics”. 12 nov. 2020. doi : 10.48550/arXiv.1812.01537. arXiv : 1812.01537 [cs].

[122] Khalid Yousif, Alireza Bab-Hadiashar et Reza Hoseinnezhad. “An Overview to Visual Odometry and Visual SLAM : Applications to Mobile Robotics”. In : *Intelligent Industrial Systems* 1.4 (1er déc. 2015), p. 289-311. issn : 2199-854X. doi : 10.1007/s40903-015-0032-7.

[123] Dayang Nur Salmi Dharmiza Awang Salleh et Emmanuel Seignez. “Swift Path Planning : Vehicle Localization by Visual Odometry Trajectory Tracking and Mapping”. In : *Unmanned Systems* 06.04 (29 août 2018), p. 221-230. issn : 2301-3850. doi : 10.1142/S2301385018500085.

[124] Dayang Nur Salmi Dharmiza Awang Salleh et Emmanuel Seignez. “Longitudinal Error Improvement by Visual Odometry Trajectory Trail and Road Segment Matching”. In : *IET Intelligent Transport Systems* 13.2 (2019), p. 313-322. issn : 1751-9578. doi : 10.1049/iet-its.2018.5272.

[125] Tommi Tykkälä, Cédric Audras et Andrew I. Comport. “Direct Iterative Closest Point for Real-Time Visual Odometry”. In : *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). Nov. 2011, p. 2050-2056. doi : 10.1109/ICCVW.2011.6130500.

[126] Mohammad O. A. Aqel et al. “Review of Visual Odometry : Types, Approaches, Challenges, and Applications”. In : *SpringerPlus* 5.1 (28 oct. 2016), p. 1897. issn : 2193-1801. doi : 10.1186/s40064-016-3573-7.

[127] Huizhong Zhou et al. “StructSLAM : Visual SLAM With Building Structure Lines”. In : *IEEE Transactions on Vehicular Technology* 64.4 (avr. 2015), p. 1364-1375. issn : 1939-9359. doi : 10.1109/TVT.2015.2388780.

[128] Albert Pumarola et al. “PL-SLAM : Real-time Monocular Visual SLAM with Points and Lines”. In : *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA). Mai 2017, p. 4503-4508. doi : 10.1109/ICRA.2017.7989522.

[129] Jigang Liu et al. “Conditional Simultaneous Localization and Mapping : A Robust Visual SLAM System”. In : *Neurocomputing* 145 (5 déc. 2014), p. 269-284. issn : 0925-2312. doi : 10.1016/j.neucom.2014.05.034. (Visité le 29/07/2023).

[130] Andreas Geiger, Julius Ziegler et Christoph Stiller. “StereoScan : Dense 3d Reconstruction in Real-Time”. In : *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011 IEEE Intelligent Vehicles Symposium (IV). Juin 2011, p. 963-968. doi : 10.1109/IVS.2011.5940405.

[131] Christian Forster et al. “SVO : Semidirect Visual Odometry for Monocular and Multicamera Systems”. In : *IEEE Transactions on Robotics* 33.2 (avr. 2017), p. 249-265. issn : 1941-0468. doi : 10.1109/TRO.2016.2623335.

[132] Deok-Hwa Kim et Jong-Hwan Kim. “Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment”. In : *IEEE Transactions on Robotics* 32.6 (déc. 2016), p. 1565-1573. issn : 1941-0468. doi : 10.1109/TRO.2016.2609395.

[133] Jeong WooYeon et Mu Lee Kyoung. “CV-SLAM : A New Ceiling Vision-Based SLAM Technique”. In : *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. Août 2005, p. 3195-3200. doi : 10.1109/IROS.2005.1545443.

[134] Dong Yeop Kim et al. “A New cvSLAM Exploiting a Partially Known Landmark Association”. In : *Advanced Robotics* 27.14 (1er oct. 2013), p. 1073-1086. issn : 0169-1864. doi : 10.1080/01691864.2013.805470.

[135] S. Hwang et J. Song. "Monocular Vision-Based SLAM in Indoor Environment Using Corner, Lamp, and Door Features From Upward-Looking Camera". In : *IEEE Transactions on Industrial Electronics* 58.10 (oct. 2011), p. 4804-4812. ISSN : 1557-9948. DOI : 10.1109/TIE.2011.2109333.

[136] Hyukdoo Choi, Ryunseok Kim et Euntai Kim. "An Efficient Ceiling-view SLAM Using Relational Constraints Between Landmarks :" in : *International Journal of Advanced Robotic Systems* (1er jan. 2014). DOI : 10.5772/57225.

[137] Arthur Ribacki et al. "Vision-Based Global Localization Using Ceiling Space Density". In : *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018 IEEE International Conference on Robotics and Automation (ICRA). Mai 2018, p. 3502-3507. DOI : 10.1109/ICRA.2018.8460515.

[138] Yuehua Li et al. "An Improved Graph-Based Visual Localization System for Indoor Mobile Robot Using Newly Designed Markers". In : *International Journal of Advanced Robotic Systems* 15.2 (1er mars 2018), p. 1729881418769191. ISSN : 1729-8814. DOI : 10.1177/1729881418769191.

[139] David Schubert et al. "Direct Sparse Odometry with Rolling Shutter". In : *Computer Vision – ECCV 2018*. Sous la dir. de Vittorio Ferrari et al. Lecture Notes in Computer Science. Cham : Springer International Publishing, 2018, p. 699-714. ISBN : 978-3-030-01237-3. DOI : 10.1007/978-3-030-01237-3_42.

[140] Peter J. Huber. "Robust Estimation of a Location Parameter". In : *The Annals of Mathematical Statistics* 35.1 (mars 1964), p. 73-101. ISSN : 0003-4851, 2168-8990. DOI : 10.1214/aoms/1177703732.

[141] Zichao Zhang et Davide Scaramuzza. "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry". In : *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Oct. 2018, p. 7244-7251. DOI : 10.1109/IROS.2018.8593941.

[142] Abdelhak Bougouffa et al. *Indoor Ceiling Vision Odometry Based on Direct Sparse Odometry*. Juill. 2023. preprint.

[143] Andreas Geiger, Philip Lenz et Raquel Urtasun. "Are We Ready for Autonomous Driving ? The KITTI Vision Benchmark Suite". In : *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012 IEEE Conference on Computer Vision and Pattern Recognition. Juin 2012, p. 3354-3361. DOI : 10.1109/CVPR.2012.6248074.

[144] Holger Caesar et al. "nuScenes : A Multimodal Dataset for Autonomous Driving". 5 mai 2020. arXiv : 1903.11027 [cs, stat]. URL : http://arxiv.org/abs/1903.11027 (visité le 15/07/2020).

[145] Thomas Schöps et al. "A Multi-view Stereo Benchmark with High-Resolution Images and Multi-camera Videos". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Juill. 2017, p. 2538-2547. DOI : 10.1109/CVPR.2017.272.

[146] Rasmus Jensen et al. "Large Scale Multi-view Stereopsis Evaluation". In : *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014 IEEE Conference on Computer Vision and Pattern Recognition. Juin 2014, p. 406-413. DOI : 10.1109/CVPR.2014.59.

[147] Xun Sun et al. "A Dataset for Benchmarking Image-Based Localization". In : *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Juill. 2017, p. 5641-5649. DOI : 10.1109/CVPR.2017.598.

[148] Simone Ceriani et al. "Rawseeds Ground Truth Collection Systems for Indoor Self-Localization and Mapping". In : *Autonomous Robots* 27.4 (22 sept. 2009), p. 353. ISSN : 1573-7527. DOI : 10.1007/s10514-009-9156-5.

[149] Jianhao Jiao et al. *FusionPortable : A Multi-Sensor Campus-Scene Dataset for Evaluation of Localization and Mapping Accuracy on Diverse Platforms*. 25 août 2022. DOI : 10.48550/arXiv.2208.11865. arXiv : 2208.11865 [cs]. preprint.

[150] L. Zhang et al. "Hilti-Oxford Dataset : A Millimeter-Accurate Benchmark for Simultaneous Localization and Mapping". In : *IEEE Robotics and Automation Letters* 8.1 (2022). DOI : 10.1109/LRA.2022.3226077.

[151] Dorian Galvez-López et Juan D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In : *IEEE Transactions on Robotics* 28.5 (oct. 2012), p. 1188-1197. ISSN : 1941-0468. DOI : 10.1109/TRO.2012.2197158.

[152] Tayyab Naseer, Wolfram Burgard et Cyrill Stachniss. "Robust Visual Localization Across Seasons". In : *IEEE Transactions on Robotics* 34.2 (avr. 2018), p. 289-302. ISSN : 1941-0468. DOI : 10.1109/TRO.2017.2788045.

[153] Mike Smith et al. "The New College Vision and Laser Data Set". In : *The International Journal of Robotics Research* 28.5 (1er mai 2009), p. 595-599. ISSN : 0278-3649. DOI : 10.1177/0278364909103911.

[154] Marius Cordts et al. *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 7 avr. 2016. DOI : 10.48550/arXiv.1604.01685. arXiv : 1604.01685 [cs]. preprint.

[155] Xinyu Huang et al. “The ApolloScape Open Dataset for Autonomous Driving and Its Application”. In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.10 (1er oct. 2020), p. 2702-2719. ISSN : 0162-8828, 2160-9292, 1939-3539. DOI : 10.1109/TPAMI.2019.2926463. arXiv : 1803.06184 [cs].

[156] Jürgen Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In : *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Oct. 2012, p. 573-580. DOI : 10.1109/IROS.2012.6385773.

[157] *Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets, FP6-IST, Grant Agreement ID : 045144*. Community Research and Development Information Service (CORDIS). 30 avr. 2009. URL : https://cordis.europa.eu/project/id/045144 (visité le 25/04/2023).

[158] Bruce D Lucas et Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In : *Proceedings of Imaging Understanding Workshop* (1981), p. 10.

[159] Guangliang Cheng. “Accurate TOA-Based UWB Localization System in Coal Mine Based on WSN”. In : *Physics Procedia*. International Conference on Applied Physics and Industrial Engineering 2012 24 (1er jan. 2012), p. 534-540. ISSN : 1875-3892. DOI : 10.1016/j.phpro.2012.02.078.

[160] Maksim Filipenko et Ilya Afanasyev. “Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment”. In : *2018 International Conference on Intelligent Systems (IS)*. 2018 International Conference on Intelligent Systems (IS). Sept. 2018, p. 400-407. DOI : 10.1109/IS.2018.8710464.

[161] Eric Brachmann et al. “On the Limits of Pseudo Ground Truth in Visual Camera Re-Localisation”. In : Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021, p. 6218-6228. DOI : 10.48550/arXiv.2109.00524.

[162] Johannes L. Schönberger et Jan-Michael Frahm. “Structure-from-Motion Revisited”. In : *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Juin 2016, p. 4104-4113. DOI : 10.1109/CVPR.2016.445.

[163] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In : *International Journal of Computer Vision* 60.2 (1er nov. 2004), p. 91-110. ISSN : 1573-1405. DOI : 10.1023/B:VISI.0000029664.99615.94.

[164] Bill Triggs et al. “Bundle Adjustment – A Modern Synthesis”. In : International Workshop on Vision Algorithms. T. 1883. Springer-Verlag, 21 sept. 2000, p. 298. DOI : 10.1007/3-540-44480-7_21.

[165] Lukas von Stumberg, Vladyslav Usenko et Daniel Cremers. “Direct Sparse Visual-Inertial Odometry Using Dynamic Marginalization”. In : *2018 IEEE International Conference on Robotics and Automation (ICRA)* (mai 2018), p. 2510-2517. DOI : 10.1109/ICRA.2018.8462905. arXiv : 1804.05625.

[166] Dingfu Zhou, Yuchao Dai et Hongdong Li. “Ground-Plane-Based Absolute Scale Estimation for Monocular Visual Odometry”. In : *IEEE Transactions on Intelligent Transportation Systems* 21.2 (fév. 2020), p. 791-802. ISSN : 1558-0016. DOI : 10.1109/TITS.2019.2900330.

[167] Davide Scaramuzza et Roland Siegwart. “Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles”. In : *IEEE Transactions on Robotics* 24.5 (oct. 2008), p. 1015-1026. ISSN : 1941-0468. DOI : 10.1109/TRO.2008.2004490.

[168] Sunglok Choi, Jaehyun Park et Wonpil Yu. “Resolving Scale Ambiguity for Monocular Visual Odometry”. In : *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Oct. 2013, p. 604-608. DOI : 10.1109/URAI.2013.6677403.

[169] Davide Scaramuzza et al. “Absolute Scale in Structure from Motion from a Single Vehicle Mounted Camera by Exploiting Nonholonomic Constraints”. In : *2009 IEEE 12th International Conference on Computer Vision*. 2009 IEEE 12th International Conference on Computer Vision. Sept. 2009, p. 1413-1419. DOI : 10.1109/ICCV.2009.5459294.

[170] Yijia He et al. “Camera-Odometer Calibration and Fusion Using Graph Based Optimization”. In : *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO). Déc. 2017, p. 1624-1629. DOI : 10.1109/ROBIO.2017.8324650.

[171] Ji Zhang et Sanjiv Singh. “Visual-Lidar Odometry and Mapping : Low-Drift, Robust, and Fast”. In : *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015 IEEE International Conference on Robotics and Automation (ICRA). Mai 2015, p. 2174-2181. DOI : 10.1109/ICRA.2015.7139486.

[172] Davide Scaramuzza et Zichao Zhang. “Aerial Robots, Visual-Inertial Odometry Of”. In : *Encyclopedia of Robotics*. Sous la dir. de Marcelo H Ang, Oussama Khatib et Bruno Siciliano. Berlin, Heidelberg : Springer, 2020, p. 1-9. ISBN : 978-3-642-41610-1. DOI : 10.1007/978-3-642-41610-1_71-1.

[173] Thien Hoang Nguyen et al. “Loosely-Coupled Ultra-wideband-Aided Scale Correction for Monocular Visual Odometry”. In : *Unmanned Systems* 08.02 (avr. 2020), p. 179-190. ISSN : 2301-3850. DOI : 10.1142/S2301385020500119.

[174] Vladyslav Usenko et al. “Direct Visual-Inertial Odometry with Stereo Cameras”. In : *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016 IEEE International Conference on Robotics and Automation (ICRA). Mai 2016, p. 1885-1892. doi : 10.1109/ICRA.2016.7487335.

Table des matières

Remerciements **I**

Résumé **III**

Abstract **V**

Liste d'acronymes **VII**

1 Introduction générale **1**
1.1 Une brève histoire de la robotique 1
1.2 Contexte 4
1.3 Problématique de la thèse 7
1.4 Organisation du manuscrit 9

2 État de l'art sur la localisation indoor **11**
2.1 Introduction 11
2.2 La localisation 13
2.3 Taxonomie du problème de la localisation 14
2.3.1 Classification par type de localisation 14
2.3.1.1 Localisation relative 15
2.3.1.2 Localisation absolue 15
2.3.1.3 Le problème du robot kidnappé 16
2.3.2 Classification par type d'environnement 17
2.3.2.1 Environnements statiques 17
2.3.2.2 Environnements dynamiques 17
2.3.3 Classification selon la réactivité de l'approche 18
2.3.4 Classification selon l'échelle d'application 19
2.4 Technologies de localisation 19
2.4.1 Solutions avec infrastructure 20
2.4.1.1 Capteurs passifs 20
Suivi par vision assistée 20
Suivi par vision centralisée 21
2.4.1.2 Capteurs actifs 22
Technologies radiofréquences 22

Communication par lumière visible . . . 23
Ultrasons . . . 23
2.4.2 Solutions sans infrastructure . . . 24
2.4.2.1 Capteurs passifs . . . 24
Localisation à l'estime (Codeurs, IMU) . . . 24
Caméra monoculaire ou stéréoscopique . . . 25
Caméra d'événements . . . 26
2.4.2.2 Capteurs actifs . . . 28
LiDARs . . . 28
Ultrasons . . . 28
2.5 La fusion multicapteur . . . 28
2.5.1 Définition . . . 29
2.5.2 Architecture des systèmes multicapteurs . . . 30
2.5.3 Le modèle JDL . . . 30
2.5.4 Fusion en trois niveaux et modèle DFD . . . 30
2.5.5 La configuration des capteurs . . . 31
2.5.5.1 Configuration complémentaire . . . 32
2.5.5.2 Configuration concurrente . . . 32
2.5.5.3 Configuration coopérative . . . 32
2.5.6 La fusion multicapteur pour la localisation indoor . . . 33
2.5.6.1 Filtre de Kalman . . . 34
2.5.6.2 Filtre de Kalman étendu (EKF) . . . 35
2.5.6.3 Filtre de Kalman non parfumé (UKF) . . . 35
2.5.6.4 Fusion par graphe de factorisation . . . 35
2.5.6.5 Filtre à particules . . . 36
2.5.6.6 Suivi multihypothèse (MHT) . . . 36
2.6 Conclusion . . . 37

3 Localisation par alignement des données multi-LiDARs 39
3.1 Introduction . . . 39
3.2 La plateforme expérimentale SmartTrolley . . . 40
3.2.1 Propulsion . . . 41
3.2.2 Calculateur embarqué . . . 43
3.2.3 Capteurs proprioceptifs . . . 43
3.2.4 Capteurs extéroceptifs . . . 44
3.3 Les LiDARs . . . 46
3.4 Appariement des données du laser *(scan-matching)* . . . 47
3.4.1 Alignement ICP – Iterative Closest Point . . . 49
3.4.2 Alignement ICL – Iterative Closest Line . . . 51
3.4.3 Alignement IDC – Iterative Dual Correspondence . . . 51
3.4.4 Alignement PM – Perfect Match . . . 52
3.4.5 Alignement corrélatif . . . 52
3.4.6 Alignement NDT – Normal Distribution Transform . . . 53
3.5 Système proposé pour la localisation relative multi-LiDARs . . . 54

3.5.1 Modélisation du système . . . 54
3.5.1.1 Le modèle cinématique . . . 54
3.5.1.2 L'odométrie des roues . . . 55
3.5.1.3 L'incertitude de l'odométrie . . . 57
3.5.1.4 Les données du LiDAR . . . 59
3.5.2 La méthode d'alignement . . . 59
3.5.2.1 Modélisation de l'environnement . . . 60
3.5.2.2 Alignement des scans et estimation de la pose . . . 61
3.5.3 Fusion par filtre de Kalman étendu . . . 64
3.5.3.1 Phase de prédiction . . . 65
3.5.3.2 Phase de correction . . . 66
3.6 Expérimentation et résultats . . . 66
3.6.1 Plateforme et environnement de l'éxpérience . . . 66
3.6.2 Résultats et discussion . . . 67
3.7 Conclusion . . . 71

4 Localisation indoor par odométrie visuelle à caméra verticale 73
4.1 Introduction . . . 73
4.2 Notions de base en vision par ordinateur . . . 74
4.2.1 Modélisation et calibrage de la caméra . . . 75
4.2.1.1 Paramètres extrinsèques . . . 76
4.2.1.2 Coordonnées homogènes . . . 77
4.2.1.3 Paramètres intrinsèques . . . 78
4.2.2 Groupes et algèbres de Lie . . . 82
4.3 Odométrie visuelle . . . 83
4.3.1 Formulation du problème . . . 83
4.4 Estimation des mouvements . . . 85
4.4.1 Estimation de mouvements 3D à 3D . . . 85
4.4.2 Estimation de mouvements 3D à 2D . . . 86
4.4.3 Estimation de mouvements 2D à 2D . . . 86
4.5 Classifications des méthodes de la VO . . . 87
4.5.1 Méthodes indirectes . . . 87
4.5.2 Méthodes directes . . . 88
4.5.3 Méthodes structurelles . . . 89
4.5.4 Dense vs. éparse . . . 89
4.5.5 Odométrie visuelle vs. SfM . . . 90
4.5.6 Odométrie visuelle vs. vSLAM . . . 90
4.6 La vision verticale *(ceiling-vision)* . . . 91
4.7 Notre contribution . . . 93
4.8 Évaluation de la VO verticale Ceiling-DSO . . . 94
4.8.1 La DSO (Direct Sparse Odometry) . . . 94
4.8.1.1 Formulation . . . 95
4.9 Résultats et validation . . . 99
4.9.1 Plateforme expérimentale . . . 99

4.9.2 Jeu de données . . . 101
4.9.3 Méthodologie . . . 102
4.9.4 Résultats et discussion . . . 104
4.10 Conclusion . . . 112

5 Jeux de données multicapteurs pour la vision verticale 115
5.1 Introduction . . . 115
5.2 Travaux similaires . . . 116
5.2.1 Jeux de données pour VO et vSLAM en indoor . . . 117
5.2.1.1 Rawseeds . . . 118
5.2.1.2 TUM RGB-D . . . 119
5.2.1.3 TUM monoVO . . . 119
5.2.1.4 FusionPortable . . . 120
5.2.1.5 Hilti-Oxford . . . 121
5.2.1.6 Synthèse et discussion . . . 122
5.3 Plateforme robotique et capteurs . . . 122
5.3.1 Caméras Intel® RealSense™ . . . 123
5.3.2 Capteurs de flux optique PMW3901 . . . 124
5.3.2.1 Système d'odométrie à base de capteurs de flux optique . . . 126
5.3.2.2 Conception du système . . . 127
5.3.2.3 Modélisation du capteur . . . 128
5.3.2.4 Calibrage du système . . . 129
5.3.2.5 Données fournies . . . 130
5.3.3 Le LiDAR IDEC SE2L . . . 130
5.3.4 L'odométrie des roues . . . 130
5.4 Le jeu de données . . . 132
5.5 Génération d'une vérité terrain . . . 134
5.5.1 Systèmes de localisation de haute précision . . . 135
5.5.2 Traçage au sol . . . 136
5.5.3 Reconstruction par calcul exhaustif . . . 136
5.6 L'outil de photogrammétrie COLMAP . . . 136
5.6.1 La recherche des correspondances . . . 137
5.6.2 La reconstruction incrémentale . . . 138
5.7 Validation du jeu de données . . . 139
5.8 Validation de la Ceiling-DSO sur le jeu de données . . . 144
5.9 Conclusion . . . 148

6 Conclusion générale 151
6.1 Bilan . . . 151
6.2 Perspectives . . . 155

A Estimation de l'échelle métrique pour la VO : un état de l'art 157
A.1 Introduction . . . 157
A.2 Vision monoculaire vs. vision stéréoscopique . . . 157
A.3 Estimation de l'échelle métrique en vision monoculaire . . . 158

A.3.1 Intégration de connaissances préalables 159
A.3.2 Fusion multicapteur 161
A.3.2.1 Fusion avec les codeurs des roues 161
A.3.2.2 Fusion avec les données du LiDAR 161
A.3.2.3 Fusion avec les mesures inertielles 162
Les méthodes de filtrage 162
Les lisseurs à retard fixe 162
Les lisseurs complets 163
A.3.2.4 Autres approches 163
A.4 L'odométrie visuelle-inertielle (VIO) 163
A.5 Conclusion 164

B Jeu de données : détails techniques **167**
B.1 Structures de données détaillées 167
B.1.1 La structure de données `sensor_msgs/LaserScan` de ROS 167
B.1.2 La structure `OpticalFlowDuo` 167
B.1.3 La structure `nav_msgs/Odometry` de ROS 168
B.2 Configuration de COLMAP 168

Bibliographie **171**