Estimation de pose multimodale- Approche robuste par les droites 2D et 3D

Louis Lecrosnier

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Automatique, Signal, Productique, Robotique

Préparée au sein de l'université Rouen Normandie

Estimation de pose multimodale
Approche robuste par les droites 2D et 3D

Présentée et soutenue par
Louis LECROSNIER

Thèse soutenue publiquement le 19 décembre 2019 devant le jury composé de		
Mme Luce MORIN	Professeure des universités, INSA de Rennes	Rapporteure
Mr Vincent FREMONT	Professeur des universités, Ecole Centrale Nantes	Rapporteur
Mme Samia BOUCHAFA	Professeure des universités, Université d'Evry Val d'Essonne/Université Paris-Saclay	Examinatrice
Mr Pascal VASSEUR	Professeur des universités, Université de Rouen Normandie	Directeur de thèse
Mr Rémi BOUTTEAU	Enseignant-Chercheur HDR, Esigelec	Co-directeur de thèse
Mr Xavier SAVATIER	Enseignant-Chercheur HDR, Esigelec	Invité

Thèse dirigée par Pascal VASSEUR, laboratoire LITIS, Université de Rouen Normandie et Rémi BOUTTEAU, laboratoire IRSEEM, Esigelec

UNIVERSITÉ DE ROUEN NORMANDIE

MI/IΣ

litis irseem ESIGELEC

Pour mes grands-parents,
qui m'ont transmis leur envie d'étudier.

Pour Sophie,
à qui je dois cette victoire.

Remerciements

Je tiens à remercier chaleureusement Pascal Vasseur et Xavier Savatier d'avoir accepté de diriger cette thèse, ainsi que Rémi Boutteau, pour son encadrement, sa disponibilité, et sa reprise du flambeau de Xavier sur la direction en fin de thèse. Merci à tous les trois d'avoir su me faire confiance et m'avoir accompagné pendant ces trois années. Je souhaite remercier Pierre Merriaux, pour son soutien de l'ombre et ses conseils précieux lors de ma première année.

Samia Bouchafa, pour avoir accepté de présider mon jury de thèse. Luce Morin et Vincent Frémont, qui ont accepté le travail de rapporteur.e et qui m'ont fourni des remarques pertinentes m'ayant permis d'améliorer mon manuscrit. Merci à tous trois de s'être montrés aussi bienveillants envers le jeune chercheur que j'étais lors de ma soutenance.

Friedrich Fraundorfer, de m'avoir accueilli pendant deux semaines très productives dans son équipe du TÜ de Graz, et de m'avoir proposé de pistes de recherche prometteuses au bon moment.

Merci à mes collègues du département Systèmes Embarqués de l'ESIGELEC, qui m'ont acueilli à bras ouverts et m'ont tout de suite fait me sentir chez moi malgré mes travers linguistiques. Je dédie une mention spéciale à Romain Rossi, dont j'ai eu la chance de partager le bureau durant ces trois ans. Merci pour ces discussions riches sur des sujets tellement variés que parfois en rapport avec notre travail. Merci avant tout d'avoir supporté avec bonne humeur mon humour que d'aucuns s'entendent à qualifier au mieux de douteux.

Merci à mes parents, qui ont tout fait pour que mes études aillent aussi loin que je le voulais. À mon père, pour avoir sû se montrer présent tout au long de mes études, et à ma mère pour avoir fait en sorte que cette présence ne devienne pas trop envahissante.

Merci finalement à Sophie, sans qui rien de tout cela n'aurait été possible. Partenaire au quotidien dans ce marathon scientifique et émotionnel qu'est le doctorat, tu as sû tour à tour me pousser à me dépasser, m'encourager dans les coups durs et les baisses de régime, m'empecher de dénigrer mes réussites. Compagne d'infortune devenue à ma plus grande joie compagne tout court, tu as subi généreusement les tentatives d'explication de mon code et des mathématique associées. Devenue malgré toi très au point sur les droites, notre rencontre justifie à elle seule toutes les formations mutualisées des écoles doctorales. Merci de m'avoir permis d'aller jusqu'au bout.

Introduction générale

Par définition, le rôle de la machine est d'assister, voire de remplacer l'humain dans des tâches fastidieuses, répétitives, ou dangereuses. Dans un premier temps, les automates se sont imposés en maîtres dans l'industrie. Capables de réaliser des suites d'actions prédéfinies, ils excellent dans la reproductibilité et la précision des tâches auxquelles ils s'attellent. Cependant, leur manque d'adaptabilité ne leur permet pas de se positionner sur tous types de travaux. Une grande part d'autonomie est nécessaire afin d'exécuter des tâches plus complexes, qui requièrent une réactivité aux changements de l'environnement. Ainsi, sortir d'un environnement contrôlé pour évoluer en environnement naturel, nettement plus complexe, n'est pas une tâche que l'on peut confier à un système seulement capable d'actions comme l'automate. Ce genre de tâches est plus généralement réservé aux robots.

Contrairement à un automate, qui ne fait qu'agir dans son environnement, le robot interagit. En ce sens, on entend qu'il est doté d'une capacité de perception, d'analyse et de prise de décision, et d'action. Il agit en réponse à des stimuli externes avec un certain degré d'autonomie. La capacité de perception d'un robot est intrinsèquement liée à la présence de capteurs, qui permettent à la machine de recueillir des données, à la fois sur elle-même, mais aussi sur son environnement. Avec la complexification des tâches confiées aux robots, la perception de l'environnement, aussi appelée perception de scène, doit se faire de manière plus complète, plus fine. Améliorer cette perception passe alors généralement par l'acquisition d'une plus grande quantité de données, à l'aide de capteurs redondants, mais aussi par la multiplication des types de capteurs différents. On fait référence à cette multiplicité des types de capteurs par le terme de *multimodalité*.

Parmi les tâches les plus complexes confiées à des robots, la navigation autonome démontre parfaitement l'intérêt de la multimodalité. Une grande adaptabilité et une perception de scène efficaces sont nécessaires lorsque l'on doit faire face à des changements de luminosité jour/nuit, à la diminution de distance de vue que peut provoquer le brouillard, ou à la géométrie du véhicule qui implique des angles de vue réduits, à la gestion des imprévus (accidents, comportements erratiques des autre usagers, mais aussi cyclistes et piétons)... De nombreux capteurs ont été développés ou adaptés pour ces circonstances, comme par exemple les télémètres laser et les radars qui sont insensibles aux conditions d'ensoleillement et de brouillard et compensent ainsi les manques des cameras opérant dans le spectre de la lumière visible.

Pourtant, si la multimodalité semble alors la solution à de nombreux problèmes, elle apporte

aussi son lot de difficultés. L'exploitation de données issues de modalités différentes nécessite généralement de connaître la configuration géométrique, c'est-à-dire l'orientation et la position, de chaque capteur relativement aux autres. Le processus d'estimation de cette configuration, aussi appelé recalage de capteurs, ou estimation de pose suivant la littérature, est un sujet particulièrement étudié depuis les débuts de la recherche en vision par ordinateur.

L'accès désormais trivial à des moyens de communication sans fil rapides et fiables permet par ailleurs de commencer à considérer des applications dont tous les capteurs ne sont pas fixes et au sein d'un seul robot mobile, mais partagés entre différents agents, qu'il s'agisse d'autres robots, ou encore de l'infrastructure dans laquelle ces derniers évoluent. D'un cas de recalage traditionnel, où la pose de chaque capteur est fixe par rapport aux autres, on se place alors dans un cas où le recalage doit être effectué entre des capteurs mobiles. Cette tâche est d'autant plus difficile que les modalités considérées fournissent des informations de nature profondément différente. Dans ces circonstances, la recherche d'éléments communs entre les données de ces différentes modalités est la clé pour commencer à entrevoir une solution à ce problème. Plus particulièrement, les éléments de nature géométrique de la scène, telles les droites de l'environnement, sont notablement invariables aux changements de modalité, et fournissent un point de départ intéressant à l'estimation de pose robuste. La mise en correspondance efficace de ces primitives géométriques à travers plusieurs modalités est par ailleurs un sujet encore largement ouvert.

Le récent regain d'intérêt pour la navigation autonome, couplé au développement de nouveaux capteurs dédiés à ce domaine, fait ainsi croître les besoins de méthodes de recalage dites robustes. On entend par cela des méthodes résilientes aux erreurs générées par les capteurs ou le traitement des données préalables au recalage de capteurs.

Ainsi, les travaux présentés dans ce manuscrit se concentrent sur la problématique d'estimation de pose multimodale robuste, en considérant des modalités désormais fréquentes dans la robotique mobile. Ce travail de thèse a donné lieu à trois contributions principales : un algorithme d'estimation de pose s'appuyant sur des droites 2D et 3D et la connaissance *a priori* de la direction verticale, ainsi que deux algorithmes d'appariement employant ces mêmes primitives géométriques.

Ce mémoire de thèse se divise en trois chapitres, présentant un état de l'art des méthodes de recalage de capteurs jusqu'au développement de méthodes robustes de recalage dédiées à la multimodalité caméra-télémètre laser.

Le premier chapitre présente tout d'abord le principe de la multimodalité, en introduisant les concepts liés à la coopération entre systèmes robotiques et présentant certains des travaux de coopération s'appuyant sur les modalités caméra-télémètre laser. On y fait état des différents capteurs fréquemment employés en robotique mobile, avec une attention particulière portée sur le capteur caméra, au cœur de ce travail de thèse. Le processus de recalage de capteurs est également abordé au sein de cette partie, et on présente un état de l'art des méthodes associées.

Dans un second chapitre est présentée la première contribution majeure de cette thèse, sous la forme d'une méthode originale d'estimation de pose reposant sur la connaissance de la direction

verticale d'une caméra, et les correspondances entre des droites 2D et 3D. On se concentre ici sur l'aspect théorique fréquemment considéré dans la littérature, où l'appariement des droites extraites de chaque modalité est supposé *a priori* connu. Après l'introduction d'un état de l'art des méthodes de recalage de capteurs basées sur les droites, notre méthode est comparée aux algorithmes de l'état de l'art. Une étude de la robustesse extensive est réalisée sur un jeu de données en simulation. Enfin, ces résultats sont confrontés à des données réelles à l'aide d'un jeu de données constitué au sein du laboratoire IRSEEM, puis sur le jeu de données public KITTI. Lors de cette évaluation de la robustesse, les résultats obtenus démontrent une amélioration des performances, en comparaison avec les algorithmes de l'état de l'art.

Le troisième chapitre de ce mémoire introduit deux algorithmes d'appariement de droites 2D et 3D. Basés sur la célèbre méthode RANSAC, ces algorithmes se distinguent en employant respectivement deux et une paire(s) de droites pour réaliser une mise en correspondance, ou éliminer d'un jeu de données des correspondances de droites aberrantes. On se place dans ce cas dans une approche plus réaliste de l'estimation de pose, dans laquelle les correspondances entre les éléments saillants de chaque modalité ne sont pas connues. Une étape d'appariement est alors nécessaire. Dans un premier temps sont présentées les métriques spécifiques à l'appariement de droites, ainsi qu'un état de l'art des méthodes de ce domaine. Afin de valider le comportement de ces algorithmes, une étude de la robustesse de ces algorithmes est finalement proposée sur un jeu de données de simulation, ainsi que sur les jeux de données IRSEEM et KITTI. Les résultats présentés mettent en évidence une grande amélioration des performances dans la suppression de paires aberrantes pour un jeu de données pré-apparié, ainsi qu'en terme d'appariement en comparaison avec les méthodes de l'état de l'art.

Finalement, un résumé des différents points clés est présenté, et des perspectives aux problèmes soulevés sont présentées.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Multimodalité, coopération et recalage de capteurs

1.1 Introduction

La notion de multimodalité est fondamentalement liée à celle de capteurs. Qu'ils soient placés au sein d'un même robot, ou partagés entre plusieurs machines, l'exploitation de la multimodalité est grandement facilitée lorsque la position et l'orientation de ces systèmes, ou des capteurs qui les composent, sont connues. Appelé calibrage, estimation de pose ou recalage de capteurs en fonction des disciplines qui s'y attellent, l'estimation de ces paramètres est une problématique d'autant plus ouverte que de nouvelles technologies de capteurs émergent fréquemment.

Ce chapitre a la vocation triple de présenter de manière succincte les concepts liés à la multimodalité et les possibilités qu'offre la collaboration entre systèmes robotiques, d'exposer les technologies de capteurs les plus courantes en robotique mobile, et d'aborder les axes de recherche majeurs du recalage de capteurs.

1.2 Coopération et robotique

1.2.1 Concepts autour de la coopération

De l'automate au robot D'un point de vue logiciel comme matériel, l'architecture d'un robot est généralement pensée de manière à séparer les différentes tâches à effectuer suivant plusieurs niveaux d'abstraction, qui sont de plus en plus nombreux à mesure que la complexité de l'environnement et de la tâche confiée au robot s'accentue. Aux niveaux les plus bas, on retrouve les compétences liées à l'automatisme. Ces compétences, telles que l'asservissement des actionneurs, sont communes au robot et à l'automate. Aux niveaux les plus hauts, on retrouve les compétences les plus complexes, telles que la localisation, la planification d'actions et de

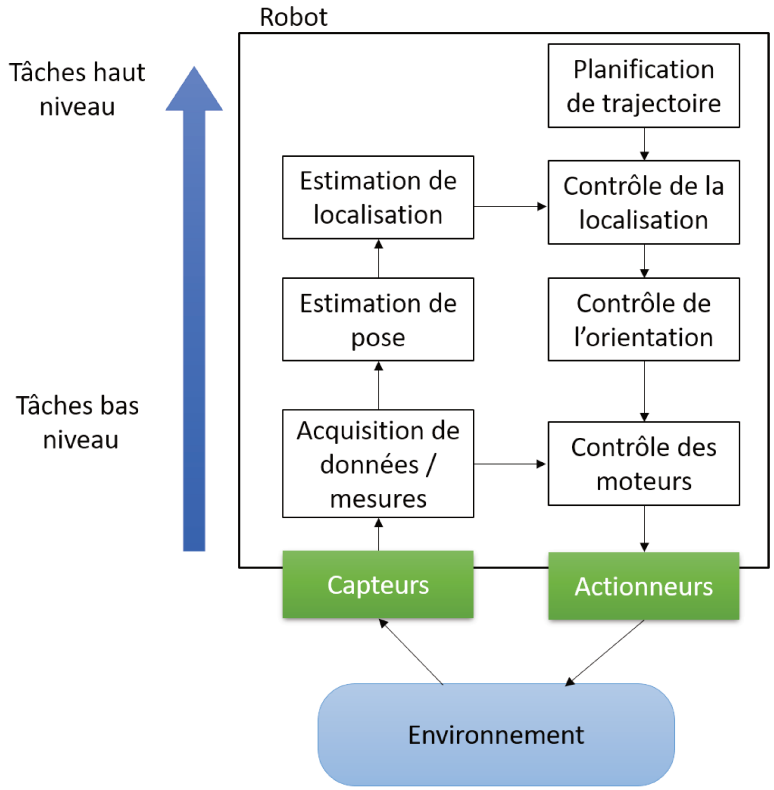

FIGURE 1.1 – Cycle perception, décision, action d'un système robotique.

trajectoires, ... La figure 1.1 illustre ces éléments. Les différentes architectures de contrôle d'un robot dépendent des interactions, généralement cycliques, entre ces fonctions majeures que sont la perception, la décision et l'action, elles-mêmes réparties en un nombre variable de couches d'abstractions.

Redondance, complémentarité et dissemblance : Pourquoi collaborer ? Au sein d'un seul robot, de nombreux sous-systèmes collaborent dans l'optique de réaliser une tâche commune. Pour les tâches les plus complexes, les plus critiques, ou dans des environnements particulièrement contraints (intervention en milieu radioactif, aéronautique, ferroviaire, ...), certaines fonctions peuvent être dupliquées à des fins de *redondance* ou de *dissemblance*. En aéronautique, par exemple, la vélocité relative de l'appareil par rapport à l'air environnant est mesurée à l'aide de dispositifs nommés sondes de Pitot. Ces sondes sont en général au minimum triplées, et un vote est constamment effectué par le calculateur de vol afin de déterminer si une des sondes est défectueuse. De cette manière, on peut isoler les valeurs aberrantes. Il s'agit du principe de *redondance*. Pour accentuer encore la sûreté de fonctionnement, il est fréquent que certains calculateurs avioniques soient doublés ou triplés, mais avec des implémentations dans des langages de programmation utilisant des bases différentes (Ada et C, par exemple). Ainsi, si une avarie ou un arrêt inattendu lié à la logique d'implémentation survient, les autres calculateurs ont moins de chances d'être impactés. On fait ici référence au principe de la *dissemblance*. Dans la robotique mobile, cette fois, il est désormais relativement fréquent de rencontrer des robots munis de capteurs *complémentaires*. On note par exemple les caméras dites RGB-D (RGB pour *Red Green Blue Depth* en anglais, ou Rouge Vert et Bleu - Distance). Ces cameras fournissent une image du spectre visible, augmentée d'une estimation de la profondeur de la scène. Cette profondeur est mesurée à l'aide de textures projetées dans le spectre infrarouge, et observées par une seconde caméra scrutant ce spectre lumineux. Un traitement en ligne permet *in fine* d'associer une profondeur à chaque pixel RGB. Ce type de capteur utilise le principe de la *complémentarité* des capteurs du spectre visible et infrarouge.

La figure 1.2 présente un autre intérêt à la prise de vue infrarouge par rapport à une vue du spectre visible dans le cas d'un environnement enfumé. L'image de droite est acquise à l'aide d'une caméra infrarouge Flir.

Systèmes indépendants, systèmes communicants, systèmes collaborants Depuis le début du siècle dernier, nous assistons à un développement exponentiel des moyens de communication sans fil pour le grand public. Avec une récente explosion au cours de ces trois dernières décennies, cette tendance s'est aussi largement imposée dans le domaine de la recherche robotique, où le coût en baisse de ces technologies permet désormais une communication sans fil d'un volume de données conséquent.

Cette disponibilité de la communication sans fil ouvre de nouvelles possibilités quant à l'architecture robotique, autant d'un point de vue matériel que logiciel. De systèmes robotisés indépendants, on observe l'émergence de systèmes que l'on qualifie de communicants, ou de collaborants.

FIGURE 1.2 – Image dans le spectre visible (gauche) et dans le spectre infrarouge (droite).

Ce changement de paradigme permet d'envisager des tâches jusque là particulièrement ardues, comme la collaboration simultanée d'un grand nombre de robots.

La figure 1.3 illustre ce propos. L'entreprise Intel réussit à deux reprises à battre le record du monde de drones pilotés en simultané par un seul ordinateur. On observe ici une architecture *basée infrastructure*, où les tâches de bas niveau (asservissement des moteurs, contrôle de la stabilité en conditions de vent, asservissement en position) sont embarquées par les drones, et les tâches de plus haut niveau, telles la gestion des trajectoires, sont débarquées dans une unité de contrôle au sol qui relaie les informations à chaque drone. La prise de décision est centralisée et ne nécessite pas de communication entre les individus. De plus, la communication est ici monodirectionnelle.

Dans le cas de cette démonstration, chacun des drones suit une trajectoire pré-programmée afin de réaliser un ballet aérien. Une unité centrale basée au sol transmet la trajectoire à suivre à chaque drone qui repère sa position dans l'espace à l'aide de relais GPS différentiels. Aucun des drones n'est ici équipé d'un système de détection d'obstacles, et les collisions sont uniquement évitées grâce à la relative précision de la localisation par GPS et la planification en amont de la trajectoire de chaque élément.

Pour réaliser des tâches plus complexes, d'autres travaux, tels ceux de Käslin *et al.* (2016), se dispensent d'un centre de décision et proposent une plus grande autonomie par robot. On peut parler dans ce cas de systèmes en collaboration. Ici, un drone et un robot au sol collaborent dans l'exploration d'un environnement contrôlé. Le drone survole la scène et compose une carte globale à l'aide d'une caméra monoculaire et s'y localise. En parallèle, un robot au sol compose une carte locale d'après un capteur LiDAR. La carte aérienne est envoyée au robot terrestre qui s'y localise.

Dans ce second cas, la complémentarité des modalités LiDAR et caméra n'est pas pas mise en avant, mais la vue aérienne du drone permet d'obtenir des informations qui seraient inaccessibles au robot terrestre, et ainsi de préciser sa localisation dans une carte globale.

D'autres travaux concernant la collaboration entre drones sont à citer : Bekmezci *et al.* (2013) proposent une étude des réseaux de drones ad-hoc, Forster *et al.* (2012) présentent un SLAM (localisation et cartographie simultanées) collaboratif entre plusieurs drones.

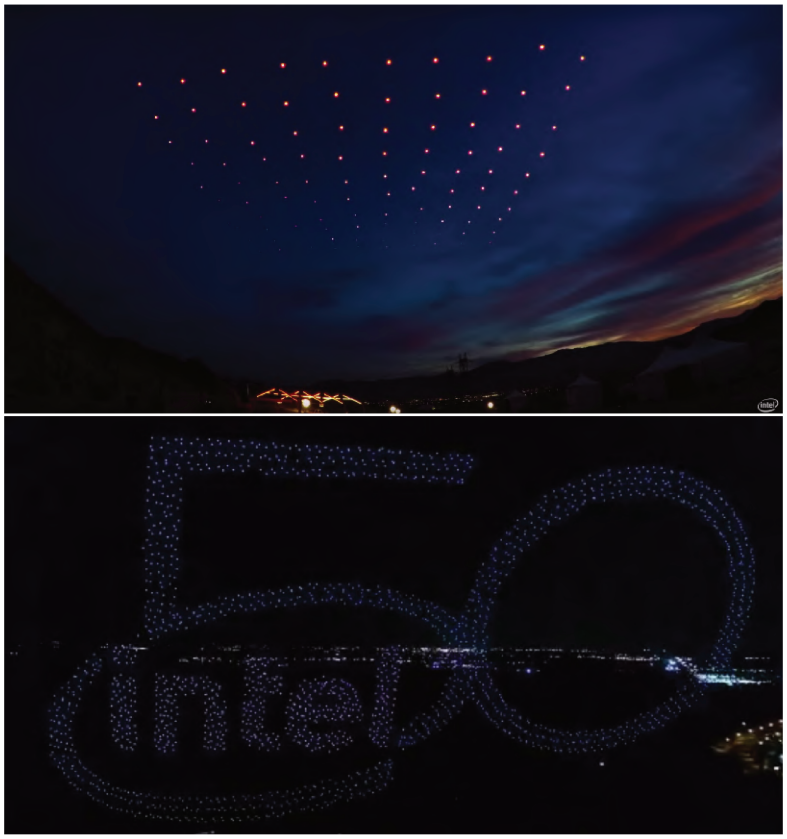

FIGURE 1.3 – Évolution des performances de commande de drones volant en simultané : en haut, record du monde par Intel en 2015 avec 100 drones, en bas, record du monde par Intel en 2018 avec 2018 drones.

Exemple d'application : Exploration collaborative Drone-Véhicule S'il existe un très grand nombre de projets de robotique collaborative, les applications intégrant des modes de déplacement différents permettent d'illustrer de manière plus marquante les possibilités de cette collaboration. On présente ici un exemple de collaboration concret : la collaboration drone-véhicule.

La complémentarité drone-véhicule au sol ne présente pas uniquement des intérêts théoriques. D'un point de vue pratique, on comprend aisément l'avantage que représente la mobilité d'un robot volant ultra-léger. Fort d'une vitesse de déplacement pouvant égaler celle d'un véhicule roulant hors-agglomération, et pouvant rester en vol stationnaire, suivant les modèles, les drones embarquent généralement des caméras légères et des moyens de communication permettant de transmettre des images. En survol de véhicule au sol, on obtient donc un point de vue aérien, à basse altitude, appelé *Bird's Eye View* (point de vue d'un oiseau). Dans le cas d'une collaboration entre un véhicule routier autonome et un drone léger, ce point de vue aérien est particulièrement complémentaire aux capteurs embarqués dans le véhicule terrestre, et permet par exemple de

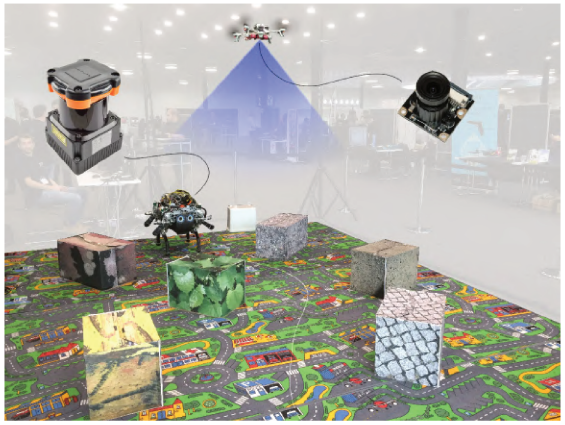

FIGURE 1.4 – Exploration collaborative de l'environnement par un robot au sol et un drone, par Käslin *et al.* (2016).

résoudre des problèmes d'occultation inhérents à la perception de scène en zone de circulation urbaine pouvant entraîner des accidents de circulation. Cette complémentarité des modalités est donc utile dans la sécurisation du trafic routier, mais aussi utile dans le cadre d'exploration air-sol, de reconnaissance...

Parmi les recherches en cours dans ce domaine, on peut citer les travaux de Dille *et al.* (2011), qui proposent une surveillance collaborative dans une vaste zone extérieure ; Christie *et al.* (2016), qui présentent une exploration collaborative d'une zone irradiée ; Cantelli *et al.* (2013), qui développent une collaboration air-sol dédiée au déminage ; Käslin *et al.* (2016), qui exploitent un déplacement aérien pour une construction de carte ; Gawel *et al.* (2016), qui présentent une construction de carte collaborative.

1.2.2 Configurations spatiales

Transformation rigide et recalage Si la complémentarité de modalités au sein d'un système ou d'un ensemble de sous-systèmes robotiques peut permettre de réaliser des tâches représentant sans cela un challenge trop important, une première difficulté doit être surmontée. Quand l'objectif est la localisation du robot dans son environnement, la première étape de la fusion de ces informations complémentaires nécessite généralement une connaissance précise de la pose de chaque capteur dans un référentiel donné. Mathématiquement, on doit alors connaître la transformation $\mathrm{Tr}_{\mathscr{R}\leftarrow\mathscr{C}} = [\mathbf{R}_{\mathscr{R}\leftarrow\mathscr{C}}|\mathbf{T}_{\mathscr{R}\leftarrow\mathscr{C}}]$ qui lie le référentiel $\mathscr{C}$ de chaque capteur à un référentiel commun, qui peut être celui du robot $\mathscr{R}$, ou encore un référentiel associé à la scène $\mathscr{M}$. Ici, $\boldsymbol{R}$ et $\mathbf{T}$ décrivent respectivement le changement d'orientation et de position du référentiel $\mathscr{C}$ vers le référentiel $\mathscr{R}$. On appelle l'ensemble des transformations liant les capteurs entre-eux les *paramètres extrinsèques* du système.

L'estimation de ces transformations est appelée *recalage de capteurs*, ou *registration* en anglais, et est abordée avec plus de détails en fin de chapitre.

Trois configurations physiques de capteurs sont possibles :

— capteurs fixes : dans un premier cas, les capteurs sont liés au robot par une transformation rigide et qui reste supposément inchangée après leur installation sur le système. Cette configuration est retrouvée par exemple dans les véhicules autonomes, où caméras, radars, LiDARs et autres capteurs sont fixés sur le véhicule. Étant donné que la transformation rigide entre les capteurs ne change pas au cours du temps, le recalage de capteurs doit être effectué une seule fois *via* une étape de calibrage du système.

— capteurs mobiles avec transformation rigide connue : dans cette seconde configuration, les capteurs sont liés à des parties mobiles du robot dont la pose est connue à l'aide de capteurs proprioceptifs. On peut imaginer un robot dont la caméra est montée sur un bras télescopique (c.f. Figure 1.5), lui même muni de capteurs de déplacement linéaire. De la même manière, un bras manipulateur 6-axes muni à son extrémité d'un capteur de type palpeur rentre dans cette catégorie. La transformation entre le palpeur et la base du bras manipulateur est susceptible de changer à chaque commande, mais est connue à l'aide des encodeurs liés à chaque axe de rotation. Ici, le recalage de capteurs peut lui aussi être effectué durant une phase de calibrage. Cependant, suivant la précision requise dans la tâche du robot, la pose du capteur mobile peut ne pas être considérée comme absolue et doit intégrer une approche probabiliste.

— capteurs libres : dans cette dernière catégorie sont regroupées les configurations multi-robots, ou les configurations pour lesquelles les transformations entre les capteurs doivent être estimées continuellement car variant au cours du temps. On retrouve ici des travaux comme ceux de Käslin *et al.* (2016), Kim *et al.* (2010) et plus généralement les applications d'exploration coordonnée et simultanée (collaboration drone et véhicule au sol, exploration par essaim de drones, voiture autonome communicant avec l'infrastructure et d'autres véhicules, ...). L'estimation du déplacement d'un même capteur au cours du temps peut être considérée comme le recalage de plusieurs capteurs identiques à des positions différentes. Pour cette raison, ce cas particulier est regroupé au sein de cette catégorie.

1.2.3 Aspect temporel

Coopération synchrone, asynchrone et souple La collaboration multi-robots évoque en premier lieu un déploiement simultané et un échange de données constant d'un groupe de robots. Pourtant, d'autres configurations sont à considérer. En effet, l'aspect temporel de l'échange de données, ou *synchronisation* peut prendre plusieurs formes. On peut présenter cet aspect en trois types distincts :

— Tout d'abord, la coopération asynchrone. Lorsque l'on considère les capteurs au sein d'un même robot, une acquisition asynchrone signifie l'absence de signal permettant de déclencher l'acquisition simultanée de données. En fonction des applications, un traitement supplémentaire de l'information doit être effectué afin de prendre en compte ces écarts temporels.

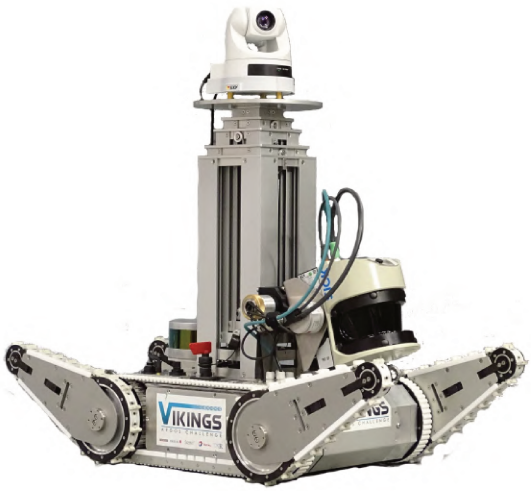

FIGURE 1.5 – Le robot VIKING de l'équipe IRSEEM, réalisé à l'occasion du challenge ARGOS .

— Ensuite, on considère la coopération synchrone. Ici, une forte contrainte temporelle est présente et les données échangées comportent généralement une estampe indiquant leur date d'acquisition, de traitement, ou les deux. Ce procédé permet d'ordonner le traitement des données et d'écarter des données trop anciennes qui pourraient être devenues moins pertinentes.
— Enfin, on observe une approche hybride, dite semi-asynchrone, ou synchronisation souple. Dans ce dernier cas, la synchronisation des données se fait lorsque cela est possible, ou nécessaire. Ce mode de synchronisation est particulièrement adapté lorsque la communication est soumise à des contraintes de débit, ou de disponibilité. Par exemple, dans le cas d'une exploration collaborative où plusieurs robots cartographient des parties différentes d'une même scène, et composent une carte globale lorsque leurs trajectoires se recoupent.

1.3 Multimodalité et représentation de l'environnement

La multimodalité est au cœur des systèmes robotiques coopérants. Il convient donc de présenter les principaux types de capteurs équipant couramment les robots. Cette section présente donc les capteurs caméra et LiDAR, qui sont utilisés dans mes travaux, ainsi que d'autres capteurs courants en robotique mobile.

1.3.1 Capteur Caméra

Le sténopé

Le sténopé (ou *pinhole*, en anglais) peut être considéré comme l'un des premiers dispositifs de prise d'image sans lentille (cf. Figure 1.6). Il s'agit d'une boîte fermée dont l'une des faces est percée d'un trou qui laisse passer la lumière. Une image se forme sur la face opposée au trou, et peut être capturée grâce à un support photosensible.

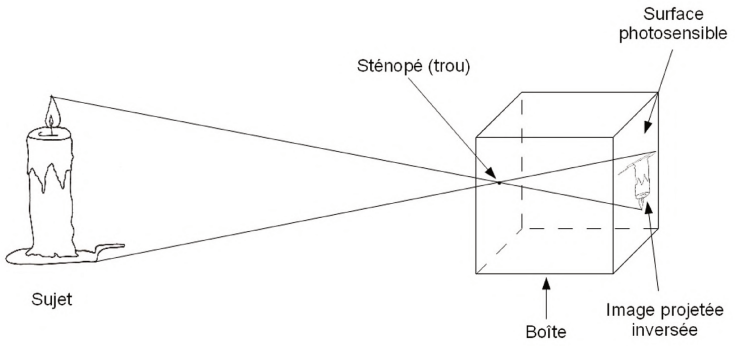

FIGURE 1.6 – Formation d'une image dans un sténopé.

Le sténopé ne laisse cependant passer qu'une petite quantité de lumière, à cause des dimensions de son ouverture. Le temps d'exposition du support photosensible doit donc être de plusieurs heures pour permettre une prise de vue.

Pour réduire le temps d'exposition, les caméras modernes utilisent une ouverture de taille plus importante, combinée à un système de lentilles, ce qui permet de concentrer plus de rayons lumineux sur le dispositif d'acquisition d'image. La surface photosensible est remplacée par un capteur photographique de type CCD ou plus récemment CMOS (cf. Figure 1.7). Ces capteurs mesurent l'intensité des rayons lumineux qui sont projetés à leur surface.

Même si le sténopé n'est plus adapté à la prise d'images, en comparaison avec les caméras modernes, son modèle géométrique simple permet de décrire la formation d'une image bidimensionnelle à partir d'une scène tridimensionnelle.

Modèle de caméra simple

On utilise communément la géométrie projective pour exprimer les transformations permettant de passer du monde tridimensionnel aux images bidimensionnelles. Bien que ces opérations soient possibles en géométrie euclidienne, les coordonnées homogènes facilitent grandement ce traitement.

Un premier modèle géométrique basé sur le sténopé permet la modélisation de caméras dites perspectives, comme le montre la figure 1.8.

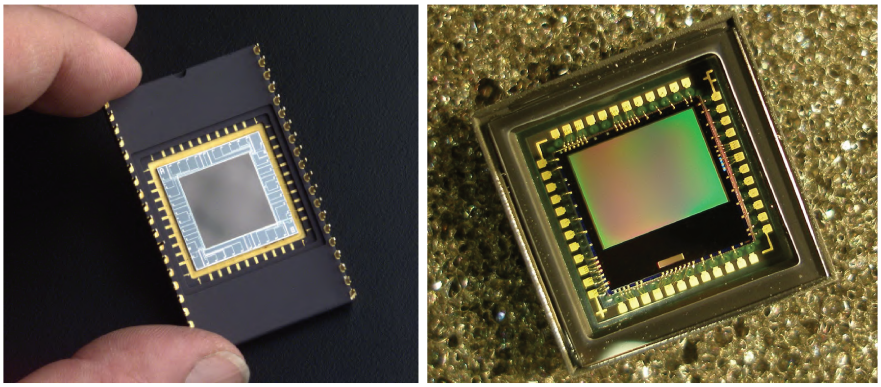

FIGURE 1.7 – Capteur CCD (gauche, Wikipedia (2019)) et CMOS (droite, Wikipedia contributors (2019a))

Le plan image représente le capteur optique de la caméra (capteur CCD ou CMOS). Le centre optique, aussi appelé centre de projection $\mathbf{C}$, est situé à la distance focale f du point principal, qui est l'intersection entre l'axe optique et le plan image. L'axe optique est la droite perpendiculaire au plan image passant par le centre de projection $\mathbf{C}$. Un point tridimensionnel $\mathbf{M}$ est généralement exprimé en coordonnées homogènes dans un référentiel arbitraire appelé référentiel ou repère monde $\mathscr{M}$, et dans une unité métrique, tel que $\mathbf{M}_{\mathscr{M}} = (X_{\mathscr{M}}, Y_{\mathscr{M}}, Z_{\mathscr{M}}, 1)^{\boldsymbol{T}}$.

On nomme $\mathbf{m}$ l'intersection du plan image et du rayon lumineux de ($\mathbf{CM}$). $\mathbf{m}$ est la projection de $\mathbf{M}$. On peut décomposer cette projection à l'aide de trois transformations successives. Une première transformation rigide $\mathbf{Tr}_{\mathscr{C}\leftarrow\mathscr{M}}$ est obtenue à l'aide des paramètres extrinsèques de la caméra. Cette transformation nous permet d'obtenir $\mathbf{M}_{\mathscr{C}}$, qui est l'expression des coordonnées du point $\mathbf{M}_{\mathscr{M}}$ dans le référentiel $\mathscr{C}$ de la caméra. L'origine de $\mathscr{C}$ est confondue avec le centre optique $\mathbf{C}$. La relation entre $\mathbf{M}_{\mathscr{M}}$ et $\mathbf{M}_{\mathscr{C}}$ est donnée par :

$$\begin{bmatrix} X_{\mathscr{C}} \\ Y_{\mathscr{C}} \\ Z_{\mathscr{C}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{0_{3x1}} \\ \mathbf{0_{1x3}} & 1 \end{bmatrix} . \begin{bmatrix} X_{\mathscr{M}} \\ Y_{\mathscr{M}} \\ Z_{\mathscr{M}} \\ 1 \end{bmatrix} + \mathbf{T} = \begin{bmatrix} \boldsymbol{R} & \mathbf{T} \\ \mathbf{0_{1x3}} & 1 \end{bmatrix} . \begin{bmatrix} X_{\mathscr{M}} \\ Y_{\mathscr{M}} \\ Z_{\mathscr{M}} \\ 1 \end{bmatrix} = \mathbf{Tr}_{\mathscr{C}\leftarrow\mathscr{M}} . \mathbf{M}_{\mathscr{M}}, \tag{1.1}$$

où $\mathbf{R}$ est une matrice de rotation entre les repères $\mathscr{C}$ et $\mathscr{M}$, et $\mathbf{T}$ est le vecteur de translation entre ces deux repères.

La seconde transformation permet de projeter sur le plan image le point 3D $\mathbf{M}_{\mathscr{C}}$ en un point image $\mathbf{m}_{\mathscr{C}} = [x, y, 1]^T$. Cette transformation est rendue possible par la matrice de projection perspective $\mathbf{P}$ et peut s'écrire :

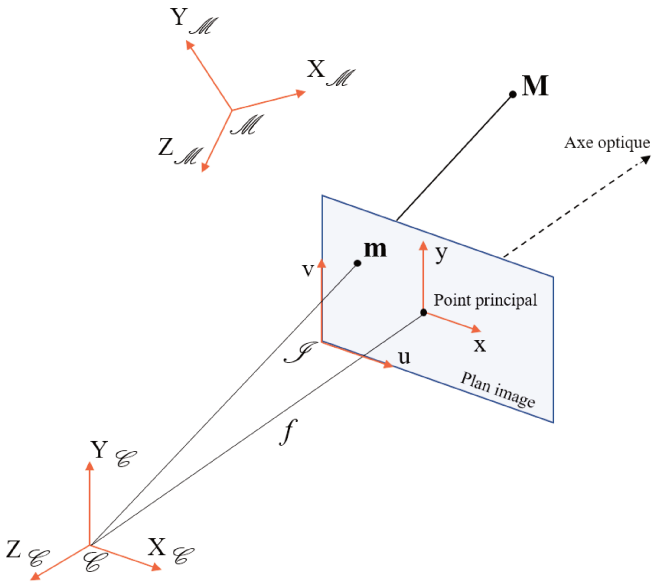

FIGURE 1.8 – Modèle géométrique du sténopé, courtoisie de Boutteau (2010)

$$\mathbf{m}_{\mathscr{C}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_{\mathscr{C}} \\ Y_{\mathscr{C}} \\ Z_{\mathscr{C}} \\ 1 \end{bmatrix} \sim \mathbf{P}.\mathbf{M}_{\mathscr{C}}, \tag{1.2}$$

où f représente la distance focale.

Une troisième transformation $\mathbf{A}$, affine cette fois, permet finalement de passer d'une expression du point $\mathbf{m}_{\mathscr{C}}$ en coordonnées métriques, vers un point $\mathbf{m}_{\mathscr{I}}$ exprimé en coordonnées pixeliques dans le référentiel $\mathscr{I}$, dit référentiel image. Cette transformation s'appuie sur :

- le nombre de pixels horizontaux k_u et verticaux k_v par unité de longueur du capteur photosensible, suivant les directions u et v (pour des pixels carrés, $k_u = k_v$),
- les coordonnées du point principal u_0 et v_0, exprimées en pixels,
- l'orientation des lignes et colonnes de l'image θ.

Cette transformation est donnée par :

$$\mathbf{m}_{\mathscr{I}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & -k_u.\cot\theta & u_0 \\ 0 & k_v/\sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}.\mathbf{M}_{\mathscr{C}}. \tag{1.3}$$

On considère en pratique que les lignes et colonnes de l'image sont orthogonales, ce qui se traduit par un angle $\theta = \pi/2$. On simplifie alors (1.3) en :

$$\mathbf{m}_{\mathscr{I}} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{A}.\mathbf{M}_{\mathscr{C}} \tag{1.4}$$

Le produit des matrices de projection affine **A** et perspective **P** est indépendant de la position de la caméra dans l'espace, ainsi que de la structure de la scène. **A.P** peut donc être pré-calculé et regroupé :

$$\mathbf{A}.\mathbf{P} = \begin{bmatrix} f.k_u & 0 & u_0 & 0 \\ 0 & f.k_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1.5}$$

On peut alors poser $\alpha_u = f.k_u$ et $\alpha_v = f.k_v$. L'équation 1.5 devient alors :

$$\mathbf{A}.\mathbf{P} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [\mathbf{K}|\mathbf{0}] \tag{1.6}$$

Le produit **A.P** regroupe toutes les grandeurs physiques liées à la caméra, aussi appelées paramètres intrinsèques de la caméra :

— la focale exprimée en pixels dans les directions u et v, notées α_u et α_v

— les coordonnées du point principal u_0 et v_0

— le paramètre d'oblicité θ, s'il n'a pas été considéré négligeable.

Si elle est exprimée en coordonnées homogènes, la projection du point $\mathbf{M}_{\mathscr{M}}$ en un point image $\mathbf{m}_{\mathscr{I}}$ est donnée par la relation :

$$\mathbf{m}_{\mathscr{I}} \sim [\mathbf{K}|\mathbf{0}].\mathbf{Tr}_{\mathscr{C}\leftarrow\mathscr{M}}.\mathbf{M}_{\mathscr{M}} \sim \mathbf{P}_{\boldsymbol{proj}}.\mathbf{M}_{\mathscr{M}} \tag{1.7}$$

Modèle de caméra réelle

Bien que le sténopé permette de poser les bases géométriques de la création d'images sur une surface plane, il ne permet pas, en pratique, de modéliser fidèlement une caméra réelle. Afin de corriger les problèmes de luminosité du sténopé, les caméras modernes sont munies d'une ouverture plus grande qui permet de capter d'avantage de lumière, ainsi que d'un système de lentilles qui permet la convergence des rayons lumineux sur le plan image. Cependant, l'introduction de lentilles n'est pas sans contrepartie, car elles induisent des déformations dans les images.

Profondeur de champ Le premier défaut de l'image lié à l'introduction de lentilles est le flou en dehors de la profondeur de champ. La profondeur de champ correspond à la zone dans laquelle doit se trouver un objet afin que son image soit nette. La taille de cette zone dépend de plusieurs paramètres lors de la prise de vue, notamment de la distance de mise au point, et du degré d'ouverture du diaphragme de la caméra. Dans le cas du sténopé, l'ouverture est très

FIGURE 1.9 – Impact de l'ouverture sur la profondeur de champ. Petite ouverture et grande profondeur de champ (gauche), grande ouverture et petite profondeur de champ (droite), Wikipedia contributors (2019b)

petite, rendant la profondeur de champ pratiquement infinie. Au contraire, le choix de l'ouverture est déterminant dans une prise de vue avec une caméra traditionnelle. La figure 1.9 illustre ce propos.

Distorsions radiales et tangentielles Les distorsions radiales et tangentielles sont des aberrations géométriques liées à l'objectif de la caméra. Deux types de distorsions peuvent apparaître dans les images Brand *et al.* (1993) :

— Les distorsions radiales sont dues à la courbure des lentilles. Elles sont observées lorsque l'on s'écarte des conditions de Gauss, c'est-à-dire lorsque l'angle α que forme le rayon lumineux et l'axe optique devient trop grand pour que l'approximation $\sin\alpha \approx \tan\alpha \approx \alpha$ reste valide. La figure 1.10 illustre ces déformations.

— Les distorsions tangentielles sont visibles lorsque les lentilles ne sont pas disposées de manière parfaitement orthogonale à l'axe optique.

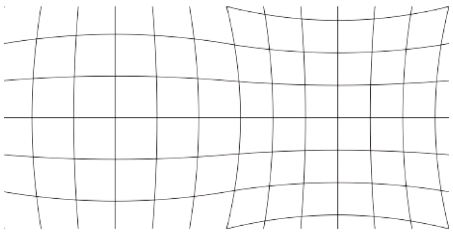

FIGURE 1.10 – Exemple de distorsions radiales. Distorsion en barillet (gauche) et distorsion en coussinet (droite).

Aberrations chromatiques Ce défaut provient de la réfraction des rayons lumineux par la lentille. L'angle de réfraction d'un rayon lumineux à travers une lentille dépend de la longueur d'onde du rayon. Dans le cas de lentilles sans dispositif de correction, on peut observer la décomposition de la lumière (cf. figure 1.11) qui entraîne un effet de flou et une irisation. Pour

compenser ce défaut, il est nécessaire de réaliser un post-traitement de l'image ou d'ajouter une lentille appelée achromat devant la lentille principale.

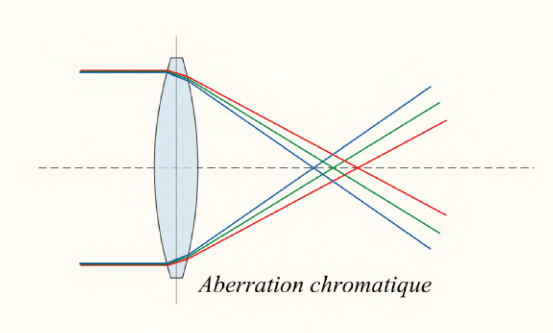

FIGURE 1.11 – Diffraction de la lumière à travers une lentille. On note la décomposition de la lumière en fonction de sa longueur d'onde. Wikipédia (2019a)

Aberrations sphériques Les aberrations sphériques surviennent lorsque les rayons lumineux frappant le bord de la lentille (la partie la plus éloignée perpendiculairement à l'axe optique) et ceux frappant son centre ne focalisent pas au même endroit, comme le montre la figure 1.12. Ainsi, on peut observer un effet de flou sur l'image.

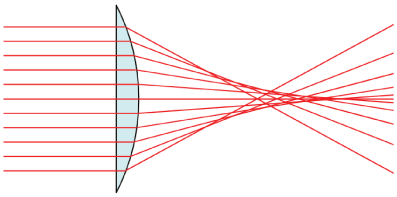

FIGURE 1.12 – Aberration sphérique : tous les rayons lumineux ne passent pas par le point focal (bas). Ce défaut entraîne un flou de l'image. Wikipedia contributors (2019c)

1.3.2 Capteur LiDAR

Définition générale du capteur

Le capteur LiDAR (Light Detection and Ranging, signifiant détection et mesure de distance par lumière, en anglais) est un système optoélectronique de télémétrie qui utilise un faisceau laser pour déterminer la distance à laquelle se trouvent les objets de son environnement.

Sur la figure 1.13, on note le fonctionnement général d'un système LiDAR. Un faisceau laser impulsionnel cohérent et polarisé, rayonnant dans les domaines infrarouge, visible et/ou

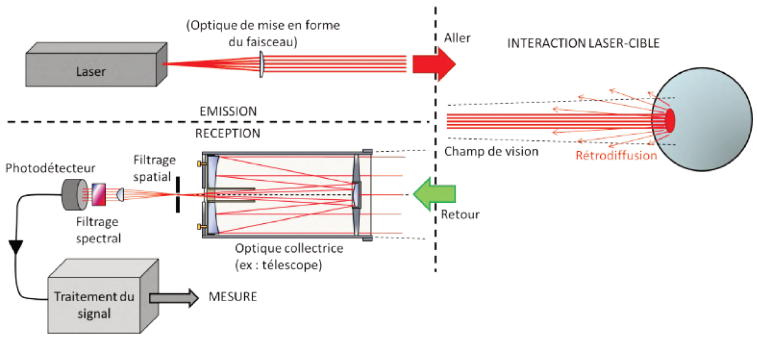

FIGURE 1.13 – Schéma d'un capteur LiDAR,Wikipédia (2019c)

ultraviolet est émis. La distance mesurée est une fonction du délai entre l'émission d'une impulsion et sa réponse issue de la réflexion du faisceau, aussi appelée écho, contre un obstacle : $d_{mesure} = \frac{c}{2}(t_{echo,max} - t_0)$, où $t_{echo,max}$ est l'instant de réception de l'écho, t_0 l'instant d'émission de l'impulsion et c la célérité de la lumière. Un sous-système électronique capable de mesurer des temps de vol de l'ordre de 10^{-10}s est intégré pour obtenir une précision de l'ordre du centimètre.

Parmi les différents systèmes LiDAR, on notera particulièrement les LiDARs à balayage, dont le faisceau laser est couplé à un système de miroir motorisé ainsi qu'à un codeur. Cet ensemble permet de projeter le faisceau suivant des pas angulaires précis. La vitesse de rotation du miroir correspond à la fréquence de rafraîchissement, et l'angle du miroir entre deux impacts donne la résolution angulaire. On obtient ainsi une carte des distances sur un plan, aussi appelée nappe, autour du dispositif LiDAR. À l'aide de cette distance d mesurée, et de l'angle θ du faisceau au moment du tir laser, on obtient un point $\boldsymbol{P_{capteur}}$ en coordonnées cartésiennes depuis les coordonnées polaires grâce la relation suivante :

$$\boldsymbol{P_{capteur}} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 1/d \end{bmatrix} .d \tag{1.8}$$

Certains capteurs LiDAR sont capables de mesurer des distances sur plusieurs nappes simultanément. On parle alors de LiDARs multi-nappes.

Chaque mesure de distance étant obtenue à un écart angulaire fixe de la mesure précédente, il est possible de convertir la carte des distances en un ensemble de points en trois dimensions, appelé nuage de points. Suivant le type de LiDAR, on obtient un nuage de points plus ou moins dense de l'environnement.

La plupart des LiDARs modernes, en plus de mesurer une distance avec un impact, renvoient en plus une information sur la réflectance. Il s'agit de l'intensité lumineuse qui est retournée au LiDAR après un écho. Cette information est particulièrement utile pour détecter les marquages

(a) LiDARs mononappe : de gauche à droite SICK MLS511, Hokuyo UTM-30LX et Leica Scanstation P50

(b) LiDARs multi-nappe : Velodyne HDL35-E (gauche) et HDL64-e (droite), respectivement avec 32 et 64 nappes

FIGURE 1.14 – LiDARs mono et multi-nappes

au sol et panneaux routiers, qui sont par définition réfléchissants.

Représentation dense Certains LiDARs mono-nappe, tels que le Leica Scanstation P50, sont utilisés pour reconstruire une carte tridimensionnelle dense et précise de l'environnement. Par sa construction, la nappe de ce LiDAR est verticale, mais un second actuateur permet de faire tourner cette nappe à 360° suivant un axe de rotation vertical. Les pas angulaires étant connus, on peut collecter un grand nombre de points de mesure par unité de surface (8 secondes d'arc ou $8/3600°$ ou encore $0.00022°$ de résolution angulaire en vertical et horizontal, pour ce modèle), au détriment d'une acquisition de plusieurs dizaines de secondes à plusieurs minutes. La figure 1.16 illustre la recomposition d'un nuage de points dense à l'aide d'un LiDAR mono-nappe couplé à un servomoteur.

Il est à noter que pour numériser un environnement complexe tel que montré sur la figure 1.15, une seule acquisition n'est pas suffisante. Le faisceau laser ne passant pas à travers les objets, la station doit être déplacée pour en scanner toute la surface. On doit employer des algorithmes de recalage de nuages de points, comme l'algorithme ICP Chen et Medioni (1992), pour estimer les déplacements de la station LiDAR et « recoller » les nuages de points complémentaires en un seul ensemble cohérent.

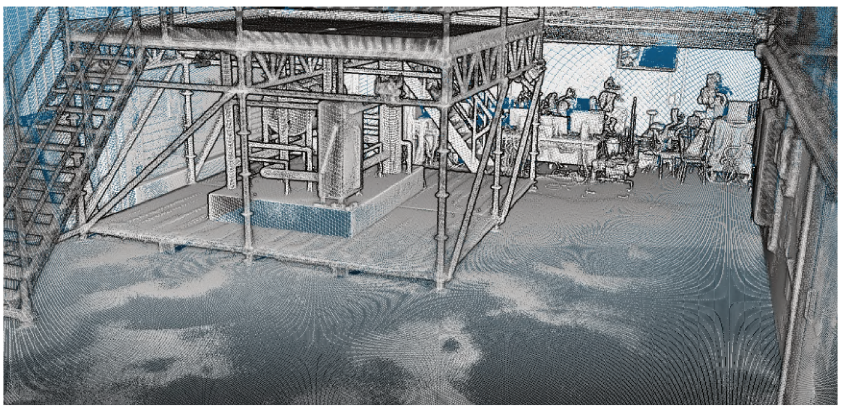

FIGURE 1.15 – Nuage de points dense représentant le laboratoire de navigation autonome de l'IRSEEM. Acquisition réalisée à l'aide d'une station LiDAR Leica ScanStation .

FIGURE 1.16 – Nuage de points reconstruit à l'aide d'un LiDAR mono-nappe combiné à un servomoteur.

Représentation éparse *A contrario*, des LiDARs tels que les SICK MLS511, HDL32-E et HDL64-E proposent une acquisition rapide des environs du capteur, avec en contrepartie d'une représentation éparse de l'environnement. Dans le cas du SICK MLS511, LiDAR mono-nappe, on obtient un nuage de points bidimensionnel.

Dans le cas des LiDAR multi-nappes, les mesures ne sont plus effectuées sur un plan : le faisceau laser est réflechi suivant plusieurs angles, fournissant ainsi une carte des distance représentant l'intersection de cônes de sommets confondus et de l'environnement, comme le montre la figure 1.17. Les LiDARs multi-nappes comptent parmi les technologies considérées dans le développement de véhicules autonomes.

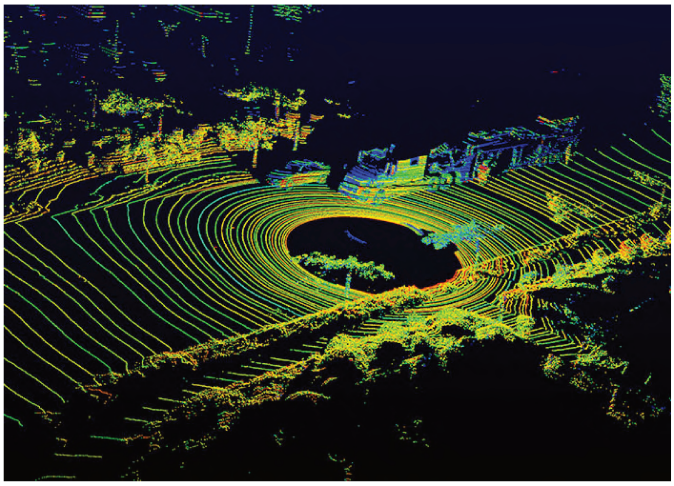

FIGURE 1.17 – Nuage de points acquis par un capteur LiDAR multi-nappes HDL64-E de la société Velodyne. https ://velodyneLiDAR.com/hdl-64e.html

Stockage d'un nuage de points Afin de stocker ces nuages de points denses, plusieurs stratégies sont possibles :

— Un tableau 3D, représentant implicitement la structure géométrique à l'aide des indices du tableau. Cette méthode implique une discrétisation plus ou moins fine de l'environnement, et nécessite de stocker aussi bien les zones sans données que les points d'impact.

— Une liste de points, pour laquelle chaque point est représenté par ses trois coordonnées, qui présente l'avantage de ne stocker que les points d'impacts, mais n'ordonne pas les données d'un point de vue géométrique.

— Un octree, qui est une structure récursive de type arbre permettant de représenter efficacement un environnement 3D. Chaque niveau de profondeur de nœud est subdivisé en 8 sous-espaces égaux, ou octants. Ainsi, l'espace occupé peut être stocké avec une granularité plus fine. Cette représentation est particulièrement adaptée aux nuages de points épars Wurm *et al.* (2010) Merriaux (2016).

1.3.3 Autres capteurs

Camera RGB-D Les caméras RGB-D (Red-Green-Blue-Depth) permettent d'acquérir une image en couleur dont une partie ou l'ensemble des pixels sont associés à une mesure de profondeur. Ces caméras proposent des taux de rafraîchissement suffisants pour permettre leur emploi dans des applications de robotique mobile, tout en affichant un prix suffisamment abordable et une compacité élevée. Ces capteurs sont constitués de l'assemblage d'une ou plusieurs caméras et d'un projecteur infrarouge, et mesurent la profondeur d'une scène en se basant sur les technologies dites de temps de vol (*Time-of-Flight* , ou ToF en anglais) ou de lumière structurée (*Structured Light* ou SL*)*. La figure 1.18 reprend ces principes, et la figure 1.19 présente deux

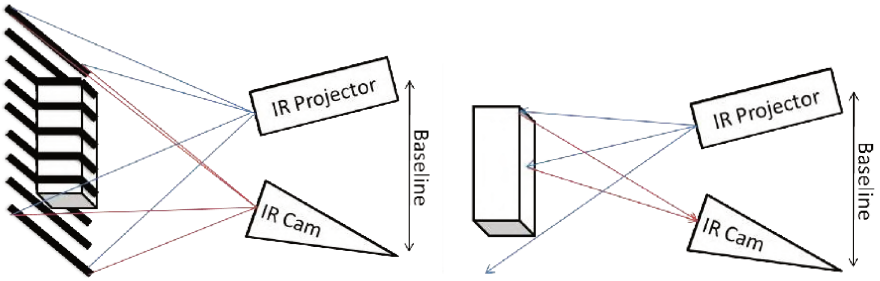

FIGURE 1.18 – Technologies de mesure de distance par lumière structurée (gauche) et temps de vol (droite) Munoz (2018)

FIGURE 1.19 – Caméras RGB-D pour la grande consommation. Kinect de Microsoft (gauche), Realsense d'Intel (droite)

modèles commerciaux de caméras RGB-D.

Le principe de la technologie ToF est de projeter une onde (électromagnétique, ou mécanique) et d'estimer une distance en fonction du temps d'écho mesuré. Dans le cas des capteurs caméra, on projette une lumière infrarouge modulée sous forme d'onde carrée pulsée ou sinusoïde continue dont on mesure respectivement soit directement le vol, soit le déphasage de l'écho. Les capteurs LiDAR, radar et télémètres à ultrasons sont aussi basés sur des principes similaires.

Dans le cas de la technologie par lumière structurée, on projette sur la scène un motif connu de lignes ou de points permettant de texturer l'environnement. Les déformations du motif projeté permettent d'estimer la profondeur de la scène, mais la portée des mesures est limitée. .

Localisation par satellite La localisation par satellite, dont le plus célèbre représentant est le système GPS (Global Positioning System) est une technologie de mesure de position géo-spatiale basée sur la triangulation. Une constellation de satellites dont la position est connue à chaque instant émet des signaux radio synchronisés temporellement à l'aide d'horloges atomiques intégrées et au sol. Ces signaux comportent un horodatage, aussi appelé *timestamp*.

La figure 1.20 reprend le principe de la triangulation satellite. En connaissant la position de chaque satellite visible par un récepteur, on peut estimer la distance satellite-récepteur en mesurant le temps de vol, connu à partir du *timestamp* et du temps d'arrivée d'un signal. Si l'intersection des signaux de deux satellites répartit les positions possibles sur un disque, celle des signaux de trois satellites donne deux possibilités ponctuelles de localisation, dont une seule est cohérente car à la surface de la planète. La synchronisation des signaux par timestamp

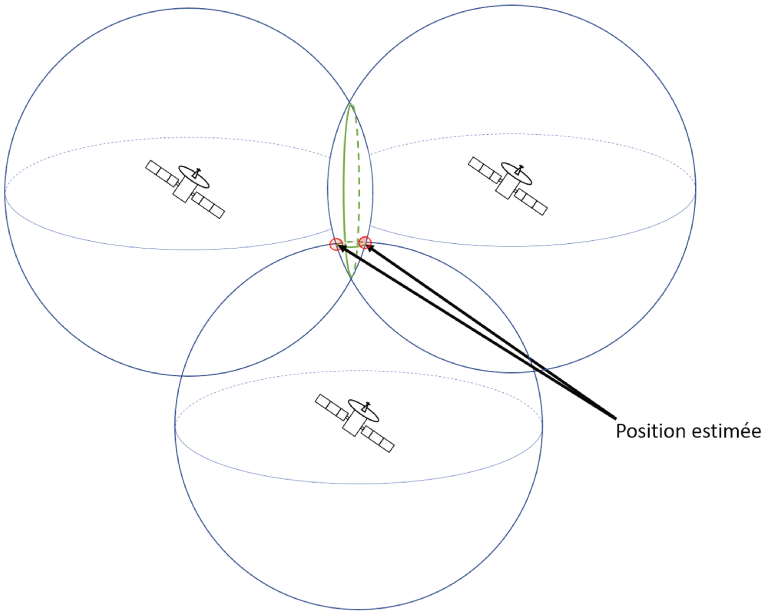

FIGURE 1.20 – Principe de la triangulation par satellite

peut, dans le meilleur des cas fournir une localisation globale à 7.4m près pour le système russe GLONASS, 5m pour le système GPS, et jusqu'à 0.01m près pour le système européen Galileo. Pour des raisons stratégiques, un délai est ajouté aux signaux. Cela diminue la précision des dispositifs de réception civils qui n'intègrent évidemment pas les clés de déchiffrement militaire. Ce délai permet néanmoins une localisation de l'ordre de 1m à 10m.

En s'appuyant sur un réseau de bornes au sol dont les positions réelles sont connues précisément, le GPS différentiel, ou DGPS, permet de compenser dans des zones localisées l'erreur induite dans les signaux satellites. Les meilleures implémentations de ce système permettent d'obtenir une localisation avec une précision de l'ordre de 1 à 3cm.

Radar, Sonar Le radar (pour RAdio Detection And Ranging) est un dispositif qui permet de mesurer à la fois la distance, la vélocité et la position d'un objet. Ces mesures sont effectuées en envoyant un signal radio modulé dont on capte l'écho. La distance est estimée en mesurant le temps de vol de l'onde, et la vitesse à l'aide de l'effet Doppler. L'effet Doppler est le décalage en fréquence d'une onde mécanique ou électromagnétique induit par le déplacement de l'objet qu'elle frappe. On note que la bande de fréquence radio, ainsi que la longueur d'onde utilisée varient en fonction de l'application. Par exemple de 3 à 30MhZ et 10 à 100m pour les radars côtiers, jusqu'à 75 à 110 GHz et 2.7 à 4.0mm pour les radars anti-collision automobiles.

Avec un coût moindre, les sonars fonctionnent sur un principe similaire, mais émettent une onde mécanique (acoustique) en lieu et place d'une onde électromagnétique.

FIGURE 1.21 – Capteur de proximité infrarouge SHARP GP2Y0A21YK.

Capteur de proximité infrarouge Les capteurs de proximité basés infrarouge sont constitués d'un émetteur et d'un récepteur infrarouge. Ils mesurent une distance en émettant une courte pulsation infrarouge. Si un obstacle se trouve sur sa course, cette pulsation rebondit et frappe le récepteur. Une mesure d'angle est ensuite effectuée entre émetteur, objet et récepteur. Ce angle varie en fonction de la distance entre capteur et obstacle. La figure 1.21 présente un capteur de proximité infrarouge SHARP.

Centrale inertielle Les centrales inertielles, ou IMU (Inertial Measurement Unit), sont des dispositifs dédiés à la mesure de l'orientation, et parfois de la vitesse d'un mobile en combinant gyroscopes, accéléromètres, et parfois magnétomètres. Le gyroscope permet de mesurer l'orientation et la vitesse angulaire, tandis que l'accéléromètre permet d'estimer l'accélération. Certains dispositifs de navigation inertielle intègrent par ailleurs les valeurs d'accélération pour estimer la vitesse, ou encore la position du mobile. Néanmoins, ces intégrations respectivement simples et doubles introduisent une dérive des mesures par une accumulation des erreurs au cours du temps.

Parmi les technologies de centrales inertielles les plus abouties, on notera particulièrement les centrales intégrant des gyroscopes 3-axes à fibre optique (FOG), avec des dérives de l'ordre du millième de degré par heure, avec néanmoins un coût à la hauteur de cette précision. Les centrales FOG utilisent le concept d'interférométrie pour mesurer une vitesse angulaire. Des technologies moins onéreuses à base de MEMS (MicroElecroMEchanical Systems) ont permis la démocratisation et la miniaturisation des accéléromètres et gyroscopes. Utilisant des éléments micro-mécaniques lithographiés, ces composants sont désormais intégrés dans la plupart des IMU de grande consommation, telles que celles que l'on peut trouver dans les téléphones cellulaires, appareils photo, systèmes d'airbag...

1.3.4 Complémentarité des modalités

Si les applications robotiques les plus simples peuvent accomplir des tâches à l'aide d'un nombre de capteurs limité, la multiplicité et la multimodalité des capteurs devient rapidement nécessaire pour les applications les plus exigeantes, comme par exemple les tâches de robotique

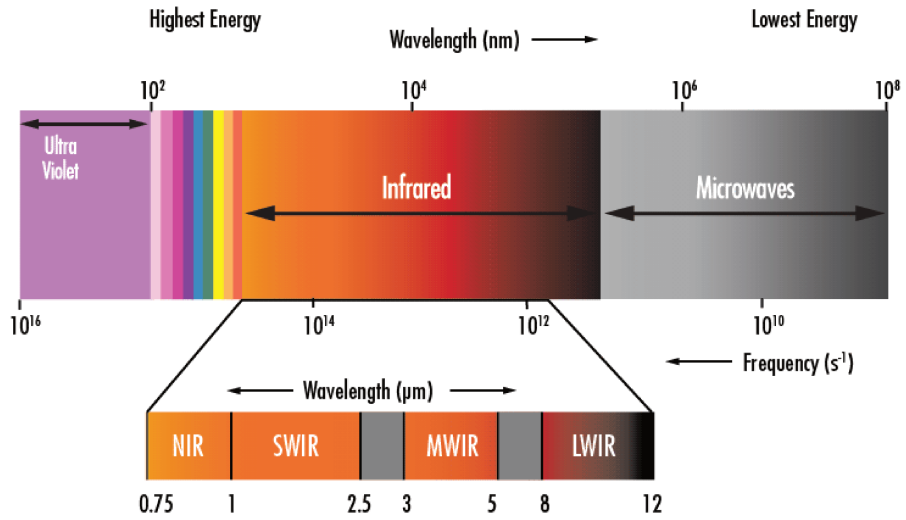

FIGURE 1.22 – Étendue du spectre électromagnétique proche du visible, edmundoptics.fr (2019)

mobile et en particulier de navigation autonome. En effet, lors du cycle Perception-Décision-Action, une prise de décision pertinente implique de percevoir suffisamment de données valables sur la scène. La multimodalité s'impose donc comme un vecteur de complémentarité indispensable pour combler les manques de tel ou tel capteur mis en défaut, car inadapté à des conditions particulières de environnement.

En robotique mobile, il est fréquent de retrouver une composition des différents capteurs cités précédemment. Afin de compenser des conditions de visibilité dégradées (circulation nocturne, brouillard, fumée, ...) il est fréquent de combiner caméras visibles et caméras infrarouges captant des longueurs d'onde différentes, de type SWIR, LWIR, NIR (cf. figure 1.22). Les figures 1.2 et 1.23 illustrent ce propos.

Dans le cas du véhicule autonome, un système de navigation par satellite est immanquablement intégré, mais la précision de la localisation globale disponible pour les applications civiles, de l'ordre du mètre, empêche son usage exclusif pour la navigation. Avec une largeur de route dans le même ordre de grandeur, on comprend aisément qu'une autre modalité doit être ajoutée afin de compléter cette localisation globale par une localisation plus locale. On retrouve alors une combinaison variable entre des capteurs de type caméra, LiDAR pour une estimation de la position plus fine, et centrale inertielle pour affiner l'estimation de l'orientation. La combinaison caméra et LiDAR/radar est d'autant plus intéressante qu'elle permet d'ajouter directement aux images des informations de profondeur, qui sont peu aisées à obtenir à l'aide d'une caméra seule ou d'un système de caméra stéréoscopique.

FIGURE 1.23 – Tunnel en conditions de brouillard, vu par des caméras scrutant le spectre visible, infragouge proche (NIR), infrarouge courte longueur d'onde (SWIR) et longue longueur d'onde (LWIR) Pinchon *et al.* (2018)

1.4 Recalage de capteurs

Exploiter les informations recueillies par un ensemble de capteurs extéroceptifs et proprioceptifs d'un robot dans une optique de perception de scène fait référence au procédé que l'on appelle fusion d'informations de capteur, ou fusion de capteurs. Cette fusion est grandement facilitée lorsque l'ensemble des transformations géométriques liant les référentiels de chaque capteur à celui du robot est connu. Comme cité précédemment, l'estimation de ces transformations est regroupée sous l'appellation *recalage de capteurs*.

Plusieurs termes peuvent être associés au recalage de capteurs. Dans le cas où l'on estime successivement au cours du temps les déplacements d'un même capteur, on fait référence à l'*odométrie* Scaramuzza et Fraundorfer (2011). Lorsque l'on estime la position d'un capteur dans une carte pré-établie, on aborde la *localisation*, qui est généralement une tâche de haut niveau dont l'odométrie est un des éléments constituants. La construction d'une carte de l'environnement s'appelle le *mapping* (pour cartographie, en français). Ces deux derniers processus peuvent être regroupés en une même tâche de localisation et *mapping* simultanés, que l'on appelle SLAM (*Simultaneous Localization And Mapping*). Cadena *et al.* (2016) proposent une revue récente du SLAM.

1.4.1 Chaîne de traitement du recalage de capteurs

Le principe du recalage de capteurs est d'estimer une transformation $\mathbf{Tr}$ maximisant le recoupement d'un jeu de données de référence $\boldsymbol{M}$ et d'un jeu de données à recaler $\tilde{\boldsymbol{M}}$ (ou jeu de données cible) comprenant N correspondances. $\boldsymbol{M}$ et $\tilde{\boldsymbol{M}}$ sont généralement des vecteurs de don-

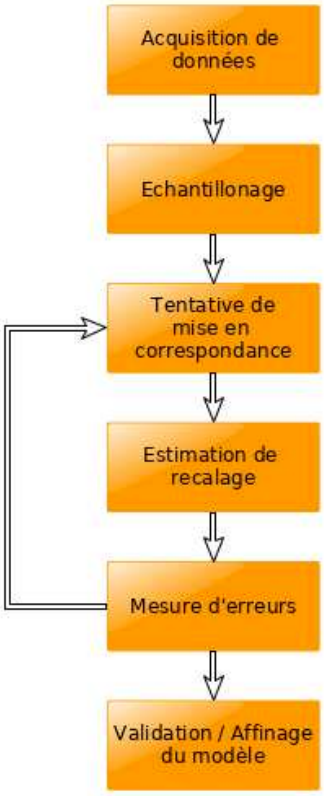

FIGURE 1.24 – Schéma général du recalage de capteurs

nées multidimensionnels, pouvant représenter directement les données empilées (liste de points 3D, empilement d'images...), ou des informations de plus haut niveau (coordonnées de zones de fort contraste, paramètres d'équation de droites...).

Afin d'atteindre cet objectif, on cherche donc à minimiser une distance entre ces deux jeux de données, telle que :

$$\mathbf{Tr} = \arg\min \sum_{i=1}^{N} \tilde{\boldsymbol{M}}_{\boldsymbol{i}} - f(\mathbf{Tr}, \boldsymbol{M}_{\boldsymbol{i}}), \tag{1.9}$$

où f représente une fonction permettant d'appliquer la transformation Tr au sous ensemble $\boldsymbol{M}_{\boldsymbol{i}}$. Pour plus de clarté, on suppose dans ce cas simplifié que $\boldsymbol{M}$ et $\tilde{\boldsymbol{M}}$ sont de même nature et de même dimension.

La chaîne de traitement classique du processus de recalage de capteurs est présentée sur la figure 1.24.

Acquisition de données La première étape du recalage de capteurs est l'acquisition de données. La section 1.3 développe cet aspect multimodal et la représentation des données associées.

Échantillonnage Une fois l'acquisition effectuée par les capteurs, il est parfois nécessaire de procéder à un ré-échantillonnage des données afin d'accélérer le recalage. En effet, il n'est pas toujours possible, ni souhaitable de traiter l'intégralité des données brutes. Dans le cas de nuages de points denses, par exemple, le volume de données représentant un modèle complexe comme celui du laboratoire de navigation autonome (LNA) de l'IRSEEM est déjà de l'ordre de plusieurs dizaines de millions de points, regroupés dans une liste textuelle de plus d'un gigaoctet. En fonction des algorithmes de recalage utilisés, on aura tendance à sous-échantillonner ce nuage, de manière à diminuer le nombre de points à considérer et ainsi réduire le coût de calcul. À l'inverse, dans le cas d'images RGB-D, l'image brute des distances comporte des zones pour lesquelles la profondeur n'est pas connue (manque de couverture du projecteur infrarouge, occultation partielle de la scène, trop grande absorption infrarouge du milieu...). On doit alors faire appel à des méthodes de reconstruction des donnés manquantes (interpolation, méthodes probabilistes...) pour obtenir une estimation de la profondeur pour chaque pixel. On notera toutefois que les caméras RGB-D modernes intègrent directement ces étapes pour fournir une estimation de la profondeur pour chaque pixel sans nécessiter une sur-implémentation par l'utilisateur final.

Les méthodes de recalage dites *directes* se basent directement sur les données dont le ré-échantillonnage ne modifie pas la nature de l'information (coordonnées de points 3D, intensité des pixels), pour procéder à l'étape de recalage de capteurs.

Une autre partie de la littérature s'appuie sur l'extraction d'informations de plus haut niveau, *feature extraction*, ou extraction de caractéristiques, pour procéder au recalage de capteurs. On parle alors de recalage basé sur des amers, ou *feature-based registration.* L'idée est ici d'identifier des caractéristiques locales, aussi appelées amers, ou *features*, qui peuvent être retrouvées après un déplacement des capteurs, car discriminantes dans la scène. Les amers peuvent être de nature photométrique, c'est-à-dire basés sur les variations de l'intensité des pixels d'une image, ou géométriques, donc issus d'une analyse structurelle de la scène. L'extraction d'amers est un sujet de recherche qui a été largement abordé avec l'essor du traitement d'image, mais qui reste ouvert dans le cas de données tridimensionnelles, comme celles générées par des capteurs LiDAR. Complexe, ce sujet est abordé dans la section 1.6.

Mise en correspondance, recalage, mesure d'erreur L'étape de mise en correspondance des données, que l'on considère une méthode directe ou indirecte de recalage, est dépendante de la transformation qui doit être estimée *in fine.* À travers cette étape, on cherche à établir des correspondances entre les données cible et de référence, tout en discriminant les correspondances aberrantes. Dans le cas des algorithmes de recalage basés sur des amers géométriques, cette mise en correspondance peut se faire de manière statistique, comme dans le célèbre algorithme RANSAC de Fischler et Bolles (1981), ou en minimisant une erreur algébrique, comme dans les travaux de Ferraz *et al.* (2014). L'idée est alors de proposer une ou plusieurs tentatives

de correspondances que l'on utilise pour estimer une transformation possible entre les jeux de données. On valide ou infirme ensuite cette proposition à l'aide de métriques, telles que l'erreur de reprojection.

La mise en correspondance dans le cadre de méthodes basées *features* est encore sujette à recherches, et les deux contributions principales de cette thèse rentrent dans cette catégorie. Le chapitre 2 présente mes résultats sur le recalage de capteurs par l'estimation de pose, et le chapitre 3 est dédié à ma contribution à la problématique d'appairage.

La mise en correspondance dédiée aux méthodes directes de recalage est présentée dans la section 1.5.

1.5 Approche directe

1.5.1 Recalage 3D/3D

Le recalage 3D/3D consiste à aligner deux jeux de données tridimensionnelles dont une partie se recoupe. Ces données peuvent être acquises par divers capteurs, comme des caméras stéréoscopiques, des réseaux de caméras, des LiDARs ou radars, caméras RGB-D... La section 1.3 donne des détails plus amples sur ces modalités. Concernant les méthodes directes de recalage 3D/3D, deux approches sont possibles en fonction des données disponibles. Si le recoupement des données est relativement important entre les deux jeux, comme par exemple dans le cas d'un faible déplacement entre deux acquisitions, on s'orientera vers des méthodes dites d'optimisation globale du recalage. Dans le cas contraire, pour des déplacements plus importants impliquant des angles de vues très différents et un recoupement faible des données, on s'orientera plutôt vers des approches dites d'optimisation locale.

L'algorithme ICP (*Iterative Closest Point*), de Besl et McKay (1992), et les variantes qui en découlent occupe une place prépondérante dans la littérature concernant les méthodes d'optimisation globale. Son objectif est de réduire itérativement la distance entre les correspondances de deux nuages de points. La version originale de l'ICP a été utilisée dans le calibrage des capteurs ayant permis de valider expérimentalement les contributions de cette thèse. Plus de détails sont fournis dans la section 2.4.3. Cet algorithme est relativement peu efficace, et la convergence vers une solution optimale peut échouer si les correspondances entre les nuages de points ne sont pas connues. Lorsque les correspondances ne sont pas connues entre les deux nuages de points, des stratégies complémentaires doivent être employées pour accélérer le recalage. On note par exemple l'utilisation d'arbres binaires à k-dimensions, ou *kd-tree* (Bentley (1975)), qui permettent d'organiser les données en fonction de leur répartition spatiale et ainsi accélérer la recherche du point le plus proche. La figure 1.25 illustre ce propos. On notera aussi des approches point à plan de Chen et Medioni (1992), ou plan à plan par Segal *et al.* (2009), contrairement à l'approche point à point de l'ICP traditionnel.

Dans le cas de méthodes d'optimisation locale, on considérera plutôt les approches de type BnB (*Branch and Bound)* Hartley et Kahl (2009) Parra Bustos *et al.* (2014) Olsson *et al.* (2008)

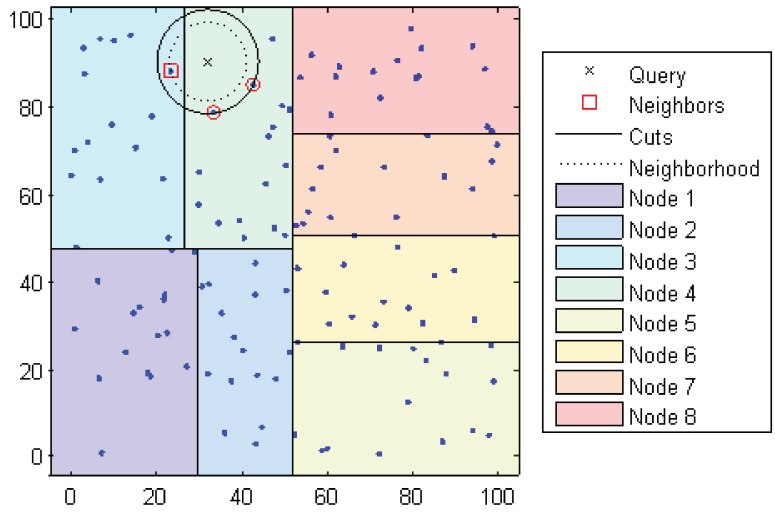

FIGURE 1.25 – Exemple d'organisation des données par un arbre binaire à deux dimensions. Découpage des données en 8 nœuds. (Mathworks (2019))

Yang *et al.* (2013) Campbell et Petersson (2016) Papazov et Burschka (2011). Cette catégorie d'algorithmes est courante dans la résolution de problèmes dont la recherche de solutions passe par l'analyse d'une fonction non convexe, c'est-à-dire dont on ne peut garantir qu'un de ses minima locaux est un minimum global. Dans le cadre du recalage de capteurs, un algorithme BnB subdivise l'espace des solutions (ici, l'espace des rotations et l'espace des translations) en sous-espaces dont les minima sont explorés. Ces méthodes varient principalement dans la représentation de l'espace des solutions (quaternions, représentation axe-angle ...), dans les motifs de séparation de ces espaces (sections de sphères, polyèdres, octree ...).

1.5.2 Recalage 2D/2D

Les méthodes directes de recalage 2D/2D, parfois appelées *template matching* (mise en correspondance de modèle), sont principalement utilisées pour le recalage d'images. Cette littérature est très liée au domaine de l'imagerie médicale, où le recalage de capteurs entre des prises de vues multimodales ou espacées au cours du temps est omniprésent.

Ces approches se basent sur des mesures de similarité sur l'ensemble, ou une partie d'une image de référence et d'une image cible, sans pour autant rechercher des zones ou des éléments géométriques discriminants. Généralement, une fenêtre de taille prédéterminée est déplacée sur l'image cible et comparée au contenu d'une fenêtre placée sur l'image de référence. Le terme « image » désigne ici une représentation en deux dimensions des données, mais pas nécessairement une intensité lumineuse. Des mesures basées sur la corrélation croisée (covariance croisée ou *cross-correlation*), la transformée de Fourrier ou l'information mutuelle sont employées pour mesurer la similarité du contenu des deux fenêtres.

La corrélation croisée et ses variantes sont définies dans Pratt (1991). L'un des principaux points faibles de ces méthodes repose sur l'aspect prédéterminé de la fenêtre déplacée. Si la transformation à estimer n'est pas une translation, une fenêtre donnée sur l'image source aura une correspondance déformée sur l'image cible. Ce problème est retrouvé dans le cas de fenêtres rectangulaires, mais aussi circulaires. Le fait de déplacer arbitrairement une fenêtre pose d'autres contraintes : deux fenêtres contenant des pixels ayant une faible variance au niveau de l'intensité auront une corrélation croisée élevée, malgré l'absence de correspondance réelle dans les images. Finalement, dans le cas des méthodes basées sur la corrélation croisée, l'utilisation directe de l'intensité des pixels provoque une forte sensibilité aux changements d'illumination de la scène, ou à la multimodalité (caméra visible-infrarouge, par exemple).

Des versions généralisées de ce procédé ont été proposées par Hanaizumi et Fujimur (1993), Berthilsson (1998) et Simper (1996) pour faire face aux déformations de l'image que la corrélation croisée normalisée simple ne peut gérer. Dans une optique similaire, Barnea et Silverman (1972) proposent un algorithme basé sur la somme des différences absolues de similarité d'intensité de pixels baptisé *Sequential Similarity Detection Algorithm* (SSDA). La somme du carré des différences de similarité SSDS a été introduite ensuite par Wolberg et Zokai (2000a). Ces deux derniers algorithmes, ainsi que la corrélation croisée normalisée, sont employés dans les applications de localisation 2.5D coopérative de Käslin *et al.* (2016), où l'intensité des pixels correspond à l'élévation du terrain. Le robot au sol se localise dans une carte 2D en comparant l'élévation qu'il observe à l'aide d'un capteur LiDAR à une carte aérienne d'élévation réalisée par un drone volant. Huttenlocher *et al.* (1992) introduit une nouvelle mesure de similarité basée sur la distance de Hausdorff Rockafellar et Wets (2009). Contrairement aux mesures de corrélation croisée, les mesures de similarité peuvent faire face aux différences entre images d'une même scène qui surviennent dans le cas d'une acquisition multimodale.

Les études de Brown (1992), Zitova et Flusser (2003) proposent un état de l'art plus exhaustif de ces méthodes.

Afin de traiter des images subissant un bruit dépendant de la fréquence, il est possible d'employer des méthodes basées sur la représentation des images dans le domaine fréquentiel que permet la transformée de Fourrier. Ces méthodes reposent sur l'analyse de la densité spectrale de puissance croisée qui est la transformée de Fourrier de la corrélation croisée des images cible et source. Elles s'appuient sur le théorème du décalage de phase de la transformée de Fourrier, introduit par Bracewell et Bracewell (1986). Parmi les méthodes remarquables, on notera les algorithmes de De Castro et Morandi (1987), Chen *et al.* (1994), Reddy et Chatterji (1996), Wolberg et Zokai (2000b), Anuta (1970), Foroosh *et al.* (2002).

Une dernière catégorie est basée sur l'information mutuelle, qui est issue de la théorie de l'information. L'information mutuelle mesure la dépendance statistique entre deux variables aléatoires, et est liée à la notion d'entropie. L'information mutuelle MI est telle que :

$$MI(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y), \qquad (1.10)$$

où H représente l'entropie, $H(.|.)$ représente l'entropie conditionnelle et $H(.,.)$ l'entropie conjointe Kullback (1997). Une des premières méthodes employant l'information mutuelle est donnée par Wells III *et al.* (1996). On notera les travaux de Thévenaz et Unser (1998), Ritter *et al.* (1999), Studholme *et al.* (1999), Maes *et al.* (1997), Pluim *et al.* (2001) et plus récemment de Wolcott et Eustice (2014).

1.6 Approche reposant sur des amers et descripteurs

Les méthodes de recalage de capteurs reposent sur la recherche de similarités dans les données acquises par les capteurs à recaler, et l'estimation des transformations géométriques associées. Si les méthodes directes de recalage s'appuient sur l'intégralité des données considérées sans discrimination, une autre catégorie de méthodes se base sur la recherche et l'extraction de caractéristiques locales discriminantes. Ces caractéristiques, appelées points d'intérêt, amers ou *feature*s, peuvent représenter la position d'un point spécifique d'une image ou d'un nuage de points. De tels éléments sont généralement recueillis à l'aide de détecteurs de coins ou de contours. Les amers peuvent aussi faire référence à une information de plus haut niveau, comme la direction d'une ligne ou d'un plan passant par un point donné. On parle dans ce cas d'amers géométriques. Enfin, les amers peuvent servir de point de référence pour décrire une zone remarquable. Cette description du voisinage d'une caractéristique discriminante fait référence aux descripteurs.

Feature detection, Feature extraction On différencie, dans la littérature, la détection et l'extraction de points d'intérêt, aussi appelés respectivement *feature detection* et *feature extraction*. Suivant les auteurs, ces deux termes peuvent cependant représenter des concepts similaires. De manière générale, la détection d'amers fait référence à la localisation, et éventuellement au calcul de l'orientation ou la forme, de ce point d'intérêt. Lors de l'extraction de *feature,* on inclue à l'étape de détection une phase de description de caractéristique. L'objectif est alors de décrire l'information utile d'un amer et de son entourage sous forme d'une signature géométrique ou photométrique, à l'aide d'un descripteur. On condense alors généralement ces descripteurs en un vecteur binaire ou de scalaires. Il est aussi possible, en lieu et place d'une description vectorielle du voisinage d'un point, de représenter directement les propriétés géométriques de l'amer (équation de droite, spline...). Cette dernière catégorie de représentation dépend de la nature des données disponibles.

Caractéristique locale idéale A travers leur étude, Tuytelaars *et al.* (2008) proposent une définition d'une caractéristique locale idéale :

— répétabilité : Les problématiques de recalage, ou d'estimation de pose, nécessitent de détecter un amer correspondant au même élément physique à travers de multiples modalités, lors d'un changement de point de vue (changement de pose) ou de conditions d'observation (jour/nuit, saisonnalité).
— caractère discriminant : L'élément détecté doit être suffisamment distinct et variable pour faciliter l'association de cette caractéristique à travers plusieurs modalités, prises de vue ou changement de pose.
— localité : Les caractéristiques extraites doivent présenter l'aspect le plus local possible. L'amer idéal serait un point, géométriquement parlant, c'est-à-dire un élément ponctuel possédant une position, mais sans volume. La localité permet de réduire les probabilités d'occultation et facilite l'estimation des transformations géométriques ou déformations photométriques. Des amers non ponctuels (droites, courbes ...) peuvent cependant présenter des avantages en cas d'importants changements de pose, ou de multimodalité où les coordonnées spatiales seules sont disponibles (nuage de points sans information de réflectance).
— quantité : La quantité de caractéristiques extraite doit être suffisante, de manière à pouvoir représenter efficacement les détails de la scène (objets de petite taille), sans pour autant sur-définir des zones comportant peu d'informations, ou des informations redondantes (plans, patchs d'intensité similaire ...). L'application finale définit ainsi la quantité à extraire.
— précision : Les amers extraits doivent être précisément localisés à travers les données ou modalités à recaler. Ce point est d'autant plus important dans les tâches de reconstruction 3D, de calibrage ou de recalage de capteurs. La précision implique aussi une considération de la taille et/ou de la forme des amers, si les caractéristiques recherchées prennent en compte ces éléments.
— performance : En lien avec la quantité de données, un compromis est à trouver concernant la performance recherchée. Pour une application dont le temps d'exécution est critique, on veillera à ne pas imposer un coût de calcul trop important dans les phases d'extraction et d'appairage des caractéristiques.

1.6.1 Détection de points d'intérêt, extraction d'amers géométriques

Détection de contours Lorsque l'on parle d'amers photométriques, on fait nécessairement référence aux éléments d'une image au sens large. Les amers photométriques correspondent à des pixels dont l'intensité diffère significativement de leurs éléments alentours, appelés voisinage. Cette intensité n'est d'ailleurs pas nécessairement une représentation en couleur (RGB) ou en nuance de gris des données, mais peut aussi représenter une profondeur (carte des distances projetée en 2D), une réflectance...

Historiquement, la détection d'amers repose sur le calcul d'une valeur de saillance, ou *cornerness*, pour chacun des points d'une image. Cette valeur peut être calculée à l'aide d'une fonction

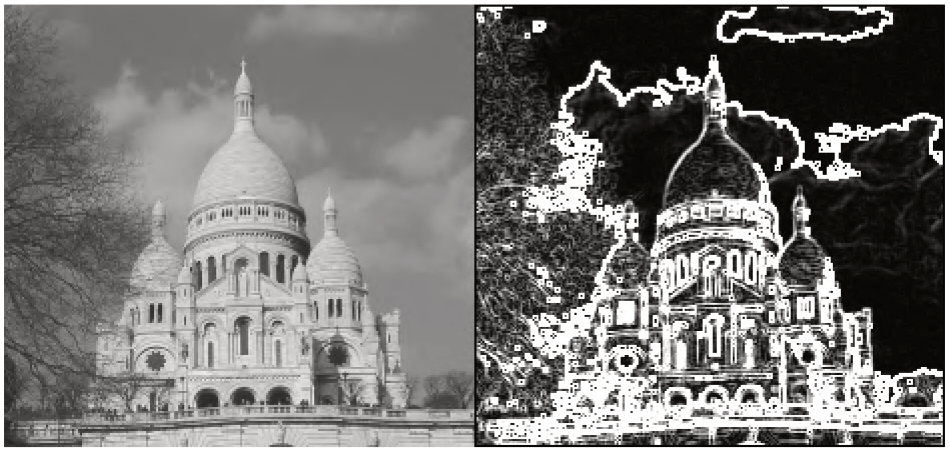

FIGURE 1.26 – Détection de contours par filtrage de Sobel (Wikipédia (2019b))

FIGURE 1.27 – Détection de contour par filtrage de Canny sur le jeu de données KITTI

C dite *corner response function*, dont la réponse est maximisée en présence d'un élément ponctuel saillant, ou à l'aide d'un filtrage de nature convolutive appliqué à chaque pixel.

Dans une image, les caractéristiques saillantes sont, par nature, des changements importants d'intensité entre plusieurs pixels successifs. Lorsque l'on observe une scène naturelle, ces changements d'intensité correspondent à des contours d'objets, des changements de texture ou de matériaux... Il est donc naturel que les premiers algorithmes de recherche de caractéristiques discriminantes soient basés sur l'extraction de contours. S'appuyant sur un filtrage convolutionnel, des algorithmes comme les filtres de Sobel et Feldman (1968) (cf. Figure 1.26), Prewitt (1970) et Canny (1986) (cf. Figure 1.27) comptent parmi les plus connus dans la littérature.

D'un point de vue géométrique, il est cependant plus compliqué d'estimer une transformation perspective à l'aide de contours libres plutôt qu'à partir de caractéristiques plus ponctuelles. Ainsi, de nouvelles méthodes orientées vers la détection de coins voient le jour.

Détection de coins

Dans Moravec (1980), on emploie une fonction qui calcule un score de saillance en se basant sur la comparaison entre l'intensité d'un patch, ou fenêtre, centré sur le pixel évalué, et les patchs centrés sur son voisinage. Le score est calculé à l'aide de la somme des différences au carré (SSD) du patch central et des fenêtres déplacées sur le voisinage de ce point, suivant les coordonnées x y de l'image. Harris et Stephens (1988) proposent une approximation de la méthode de Moravec (1980) en s'appuyant sur la dérivée seconde de la SSD. Une matrice A est définie telle que

$$\boldsymbol{A} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{1.11}$$

où l'opérateur $\langle\rangle$représente la moyenne sur les patchs évalués, et I_x et I_y sont les dérivées partielles respectivement en fonction de x et y de l'image. La fonction de réponse à la saillance C est alors définie par :

$$C = \det(\boldsymbol{A}) - k.(\mathrm{trace}(\mathbf{A}))^2 \tag{1.12}$$

avec k un hyper-paramètre déterminé empiriquement en fonction de la sensibilité souhaitée.

Shi *et al.* (1994) proposent un détecteur reposant sur cette même matrice A, mais proposent d'utiliser le minimum des valeurs propres λ_1 et λ_2 de $\boldsymbol{A}$ comme fonction C, qui devient alors :

$$C = \min(\lambda_1, \lambda_2) \tag{1.13}$$

Cette méthode est présentée comme plus efficace aux déformations affines. Dans le même esprit, d'autres contributions utilisent cette matrice $\boldsymbol{A}$, combinée à des métriques différentes. Zheng *et al.* (1999) propose de diminuer la complexité calculatoire en calculant seulement deux images des gradients de second ordre.

Smith et Brady (1997) introduisent le détecteur SUSAN (*Smallest Univalue Segment Assimilating Nucleus*), pour lequel chaque pixel $\overrightarrow{m}$, d'intensité $I(\overrightarrow{m})$ inclus dans un disque de rayon r est comparé au pixel central $\overrightarrow{m}$, ou noyau, et donne une réponse $c(\overrightarrow{m})$ telle que :

$$c(\overrightarrow{m}) = e^{-\left(\frac{I(\vec{m}) - I(\vec{m}_0)}{t}\right)^6} \tag{1.14}$$

Le seuil de détection t, de même que la mise à l'exposant 6 de la fonction c sont arbitrairement déterminés. La somme de ces réponses individuelles devient finalement le score de saillance du noyau.

Les détecteurs de la famille AST (*accelerated segment test*) s'inspirent de cette méthode, mais accélèrent grandement le calcul du score de saillance en ne considérant que les pixels situés sur le cercle de rayon r, au lieu de tous les pixels du disque. Le détecteur FAST Rosten et Drummond (2006), est l'un des plus cités de ce type.

FIGURE 1.28 – Vue aérienne de Manhattan, riche en lignes verticales

Détection de blobs D'autres méthodes sont plus orientées dans la détection des groupes de pixels saillants vis-à-vis de leur voisinage, aussi appelés blobs. Lindeberg (1998) introduit un détecteur basé sur le laplacien de gaussienne (*Laplacian of Gaussian, LoG*), introduisant l'idée d'un opérateur invariant au changement d'échelle. Ce concept est repris plus tard par Lowe (2004), qui propose une approximation du LoG sous la forme d'une différence de gaussienne (*DoG*). Ce détecteur est de plus invariant aux rotations. Le score de saillance est obtenu en effectuant la convolution d'une image avec une fonction gaussienne à deux dimensions sur plusieurs échelles, que l'on appelle pyramide d'échelles. Sima et Buckley (2013) proposent une méthode pour définir une pyramide d'échelles optimale.

Détection d'amers géométriques Par définition, l'intégralité des structures naturelles ou artificielles est soumise au champ gravitique terrestre. Cette contrainte impose aux structures bâties par l'homme une forte verticalité pour la solidité, et une horizontalité pour le confort de déplacement. D'un point de vue géométrique, ces contraintes se traduisent par un grand nombre de droites et courbes régulières dans l'environnement courant, comme l'illustre la figure 1.28.

Ces droites, plans et courbes offrent une information géométrique riche, qui peut être exploitée dans le processus de recalage de capteurs. De plus, la géométrie des structures humaines possède l'avantage certain d'être particulièrement invariante aux conditions d'éclairage, et, sauf cas particuliers, résistante aux changement saisonniers. Ces raisons ont justifié l'orientation de ce travail de thèse, qui s'articule autour de l'estimation de pose basée sur des droites.

De nombreuses méthodes de détection de ces caractéristiques géométriques ont ainsi été développées. Traditionnellement réalisée sur des images, la détection de lignes est une extension

FIGURE 1.29 – Détection de segments par filtrage de Canny et transformée de Hough

particulière de la détection de contours. Historiquement, la détection de lignes est proposée comme une étape supplémentaire d'un filtrage de contours. À l'issue d'un filtrage de contours, comme il peut être réalisé à l'aide d'un filtre de Canny (1986), une transformée de Hough (1962) permet d'extraire les lignes directrices de l'image considérée. Cette méthode est illustrée sur la figure 1.29. De nombreuses approches ont été proposées, avec des variantes permettant d'améliorer la détection : Xu *et al.* (1990), Kiryati *et al.* (1991), Matas *et al.* (2000), Galambos *et al.* (2001), Fernandes et Oliveira (2008), Li *et al.* (2009).

En parallèle, des méthodes ne reposant pas sur la transformée de Hough ont été développées. Etemadi (1992) et Chan et Yip (1996) s'appuient sur des cartes des contours. En particulier, Etemadi (1992) propose de créer une carte des contours, qui est ensuite parcourue à la recherche de courbes et segments.

Basées sur l'analyse de l'orientation des gradients associés à chaque pixels, on notera les méthodes de Von Gioi *et al.* (2010) avec leur Line Segment Detector (LSD), encore à l'état de l'art, ainsi que celles de Burns *et al.* (1986), Desolneux *et al.* (2000), Desolneux *et al.* (2007), ou encore EDlines d'Akinlar et Topal (2011) et Salaün *et al.* (2016) avec MLSD.

Si la plupart des méthodes présentées ici sont dédiées au traitement d'image, d'autres méthodes d'extraction d'amers géométriques adaptées aux données de profondeur existent.

Serafin *et al.* (2016) proposent d'extraire directement d'un nuage de points épars des plans et des droites en se basant sur une analyse en composante principale (Hotelling (1933)) du voisinage de chacun des points du nuage. On récupère ainsi la direction de la normale à chaque point, qui sert ensuite à extraire les plans et les droites du nuage. Un résultat est visible sur la figure 1.30. Lin *et al.* (2015) proposent d'extraire des droites d'un nuage de points dense d'une manière similaire. En calculant la normale de patchs, une carte des plans est reconstituée, et l'intersection de ces plans fournit des segments de droite.

Plutôt que de traiter directement un nuage de points dense ou épars, une autre approche dans le traitement de l'information d'un nuage de points consiste à profiter de la littérature très riche du traitement d'image : la simple visualisation d'un nuage de points sur un écran est après tout une projection de ce nuage sous forme d'image en deux dimensions. Il est fréquent d'encoder

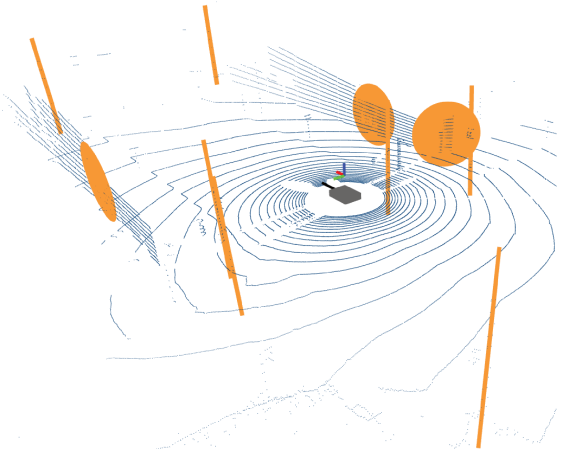

FIGURE 1.30 – Extraction de droites et de plans d'un nuage de points épars, par Serafin *et al.* (2016)

dans ce cas la distance des points par rapport à l'observateur suivant une échelle de gris. On dispose alors d'une image analysable avec les méthodes usuelles de traitement d'image (opérations morphologiques pour lisser les irrégularités du nuage de points, extracteurs d'amers, ...). Scaramuzza *et al.* (2007) calibrent un système caméra-LiDAR grâce à des amers extraits sur des scènes naturelles, tandis que Fremont *et al.* (2008) proposent l'introduction d'une mire circulaire perçue dans les deux modalités. Toujours dans une optique de calibrage, Pandey *et al.* (2015) proposent le recalage LiDAR-caméra en maximisant l'information mutuelle entre les modalités en s'appuyant sur des images d'intensité (caméra) et de réflectivité (LiDAR). Les travaux récents de Feng *et al.* (2016) sont notables pour leur utilisation des cartes des distances à partir d'un LiDAR monté sur un véhicule mobile. La figure 1.31 comporte les résultats intermédiaires de Cui *et al.* (2016), qui projettent un nuage de points sur une image, et extraient le contour d'un immeuble grâce à plusieurs étapes d'optimisation. Leurs travaux récents portent sur le recalage entre un nuage de points 3D et une image 2D extraite à partir d'une caméra panoramique tous deux intégrés à un véhicule mobile.

1.6.2 Descripteurs photométriques et géométriques

Nous avons vu dans la section précédente qu'à l'issue de la détection de points ou régions d'intérêt, plusieurs approches sont possibles quant au recalage de capteurs. La détection d'amers peut être complétée par une phase de description photométrique ou géométrique de son voisinage, qui fournit *in fine* une représentation vectorielle (binaire ou scalaire) de l'information, appelée descripteur. Il est aussi possible d'extraire des informations de nature géométrique de plus haut niveau, telles des plans, droites ou courbes passant par ces points d'intérêt. De plus, les différentes métriques applicables aux descripteurs permettent de faciliter la phase d'appairage préalable au recalage de capteurs.

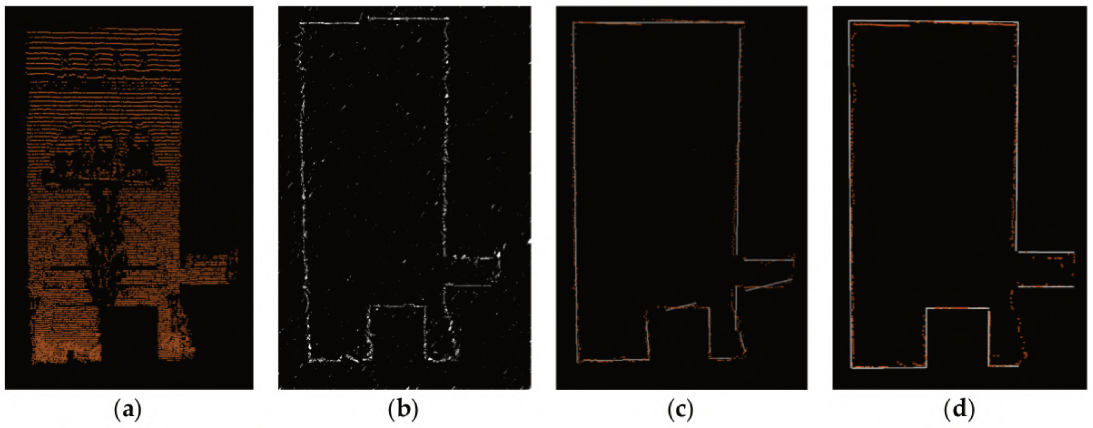

FIGURE 1.31 – Extraction de lignes 3D à partir d'un nuage de points : a) nuage de points 3D projeté en 2D, b) points en bordure de la projection, c) estimation des lignes limites par la méthode des moindres carrés, d) affinage de l'estimation à partir de contraintes physiques Cui *et al.* (2016)

Cette sous-section présente les différents descripteurs photométriques, ainsi que leurs pendants géométriques, aussi appelé descripteurs de formes.

Descripteurs photométriques Comme leur nom l'indique, les descripteurs décrivent les informations d'un patch ou d'une fenêtre centrés sur un point d'intérêt, généralement une caractéristique locale jugée pertinente par un algorithme de détection d'amers. Ces descripteurs représentent, en fonction des algorithmes, une vectorisation de l'information spatiale fréquentielle, de gradients d'intensité... Dans l'optique du recalage de capteurs, ces vecteurs sont comparés à l'aide de distances usuelles, comme les normes $L^1 et\ L^2$ ou la distance de Mahalanobis.

Parmi les descripteurs scalaires les plus cités, on retrouve les descripteurs SIFT (*Scale Invariant Feature Transform*) Lowe (2004) et SURF (*Speeded-Up Robust Features*) Bay *et al.* (2006), qui découle d'une approximation de SIFT. L'algorithme original de SIFT propose de calculer les gradients d'une fenêtre de 16 par 16 pixels autour d'un pixel central. On pondère ensuite ces gradients par une gaussienne, ce qui a pour effet d'atténuer les gradients les plus éloignés du centre. On divise ensuite ce patch en sous-ensembles de 4 par 4 pixels, dont on réalise l'histogramme des gradients d'après leur direction. On encode finalement ces histogrammes des orientations en un vecteur de dimension 128. L'algorithme SURF combine une détection basée sur une approximation de la matrice hessienne de patchs d'images, et une description s'appuyant sur la distribution de la réponse en ondelettes d'Haar du voisinage du point détecté. Il est présenté comme plus rapide à l'exécution que SIFT. Les figures 1.32 et 1.34 représentent un exemple d'extraction de *features* SURF et SIFT. Une revue plus détaillée des descripteurs scalaires est proposée dans Mikolajczyk et Schmid (2005).

Des algorithmes comme SIFT et SURF proposent d'encoder une description de caractéris-

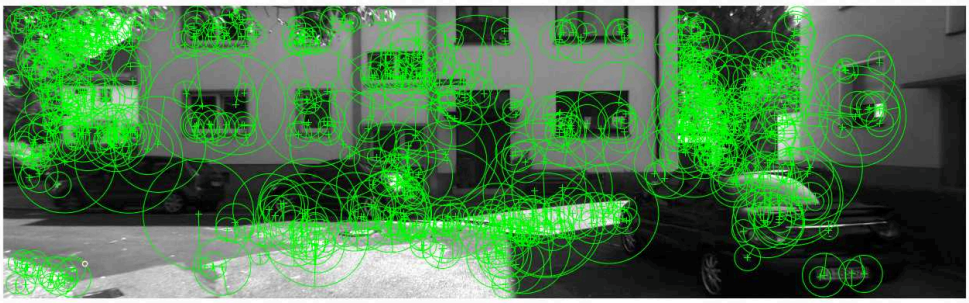

FIGURE 1.32 – Descripteurs SURF centrés autour de leurs points d'intérêt associés. Image du jeu de données KITTI

FIGURE 1.33 – Descripteurs ORB et amers associés. Image du jeu de données KITTI

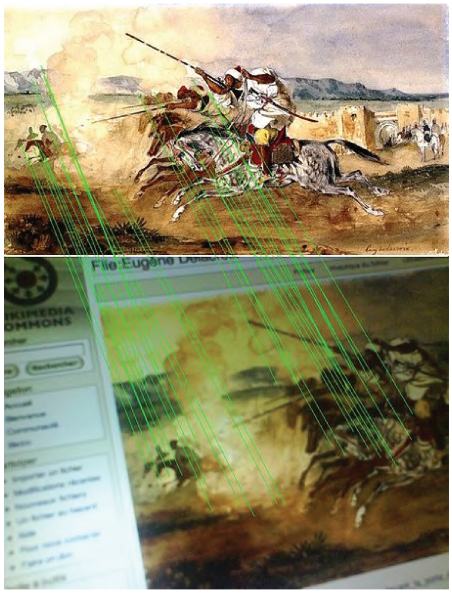

FIGURE 1.34 – Mise en correspondance de descripteurs SIFT (Wikipédia (2018))

tiques en des vecteurs scalaires de dimension respective 128 et 64. Dans des tâches comme l'estimation de pose relative ou le recalage d'image, il est nécessaire de calculer une distance entre de nombreux vecteurs, ce qui s'avère d'autant plus coûteux en calcul que les vecteurs considérés sont scalaires, et de grande dimension. Dans l'idée d'accélérer ce processus, des méthodes alternatives proposent une description binaire des points d'intérêt, et permettent l'utilisation d'autres métriques, comme la distance de Hamming. C'est le cas d'algorithmes de description tels que BRIEF, de Calonder *et al.* (2010), où des tests binaires d'intensité entre des pixels aléatoirement tirés dans un patch sont directement encodés, ou plus récemment ORB (*Oriented FAST and Rotated BRIEF*), de Rublee *et al.* (2011), qui combine le détecteur FAST et une évolution de BRIEF prenant en compte les rotations. Une revue comparative de descripteurs binaires est proposée dans Heinly *et al.* (2012).

Descripteurs de forme 2D et 3D A travers la description de forme, l'objectif n'est plus de décrire le voisinage d'un point d'intérêt d'un point de vue photométrique, mais plutôt de décrire la forme d'une zone de points saillants. Cette approche est un sujet de recherche très actif depuis les trois dernières décennies, car particulièrement utilisée dans des moteurs de recherche de formes 3D, ou encore dans l'infographie.

La nécessité de traiter efficacement les mesures de profondeur de scène connaît de plus un regain d'intérêt avec la démocratisation des dispositifs d'acquisition d'information de profondeur, comme les LiDARs, caméras RGB-D, ou encore le perfectionnement des méthodes d'estimation de profondeur à l'aide de caméras stéréoscopiques. Vranic et Saupe (2004) proposent une étude extensive des descripteurs 3D dédiés à la recherche et à la mise en correspondance de modèles 3D.

Descripteurs de contour	Basés transformée de Fourrier (FD)
	Basés transformation en Ondelette (WD)
	Basés Espace de courbure multi-échelle (CSS)
	Basés contexte de forme
Descripteurs de région	Basés Moment de Zernike
	Basés SIFT
	Basés Transformation Radiale Angulaire (ART)
Descripteurs hybrides	FD + ART
	FD + ZMD

TABLE 1.1 – Classification des descripteurs de forme 2D par Kazmi *et al.* (2013)

Plus récemment, Kazmi *et al.* (2013) apportent une revue des différents descripteurs de forme 2D et 3D.

Concernant les descripteurs de forme 2D, on note trois grandes catégories : les descripteurs de contour, de région et les descripteurs hybrides. Ces catégories sont reprises dans le tableau 1.1. De la même manière, les descripteurs de forme 3D sont regroupés en cinq catégories :

— les méthodes basées sur la silhouette d'une forme 3D selon plusieurs points de vues (*View-based methods*),
— les méthodes s'appuyant une description par histogramme, où l'on sépare l'environnement en sous-volumes appelés *bins*, dont on compte les points qu'ils renferment,
— les méthodes basées sur un changement d'espace de représentation, comme le passage en représentation fréquentielle par la transformée de Fourrier, ou la décomposition à l'aide de fonctions harmoniques sphériques,
— les méthodes faisant appel à la théorie des graphes,
— les approches hybrides qui combinent différentes méthodes citées ci-dessus.

Le tableau 1.2 reprend cette classification.

Les travaux de Gawel *et al.* (2016) comparent des descripteurs volumiques appliqués à un nuage de points. Ces descripteurs sont ici des volumes cloisonnés de l'espace dans lesquels on compte le nombre de points du nuage de points emprisonnés. Ces volumes servent ensuite dans une méthode de description par histogramme. Sur la figure 1.35, on peut observer les volumes des descripteurs Gestalt et Boxoli.

Avec l'intérêt que représente l'apprentissage profond depuis une dizaine d'années, certaines méthodes reposant sur les réseaux de neurones convolutionnels ont vu le jour. On notera particulièrement les travaux de Zeng *et al.* (2016), qui proposent d'entraîner un réseau à localiser des points d'intérêts dans des nuages de points denses de type RGB-D, et d'extraire puis appairer des descripteurs. Ils permettent ainsi un recalage de deux nuages de points denses. Un extrait du résultat de l'apprentissage de leur réseau de neurones 3DMatch est présenté sur la figure 1.36.

Conclusion A travers ce chapitre, nous avons introduit les différents concepts liés à la multimodalité robotique. Après avoir présenté les capteurs les plus courants en robotique mobile,

Changement de Vue	*Adaptative view clustering*
	Descripteur compact Multi-vue (CMVD)
	Descripteur de champ lumineux (LFD)
Histogramme	Spectre de forme
	Distribution de forme généralisée
	Bag-of-Features (BoF)
Transformation de l'espace de représentation	Descriptions par fonctions harmoniques sphériques (SHD)
	Analyse en composante principale de la SHD
	descripteur par *Spherical Trace Transform* (STT)
Graphs	Graph du squelette du modèle
	graph de Reeb
Descripteurs hybrides	CMVD+STT
	Rétention de profondeur (Depth buffer) + Silhouette + REXT
	SIFT + Bag-of-Feature
	Depth-buffer + Description par harmoniques sphériques

TABLE 1.2 – Classification des descripteurs de forme 3D, par Kazmi *et al.* (2013)

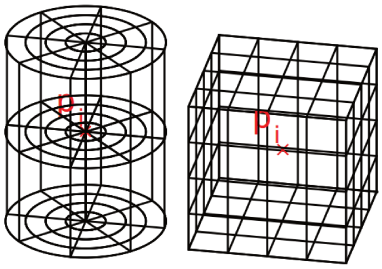

FIGURE 1.35 – Descripteurs Gestalt (gauche) et Boxoli (droite), par Gawel *et al.* (2016)

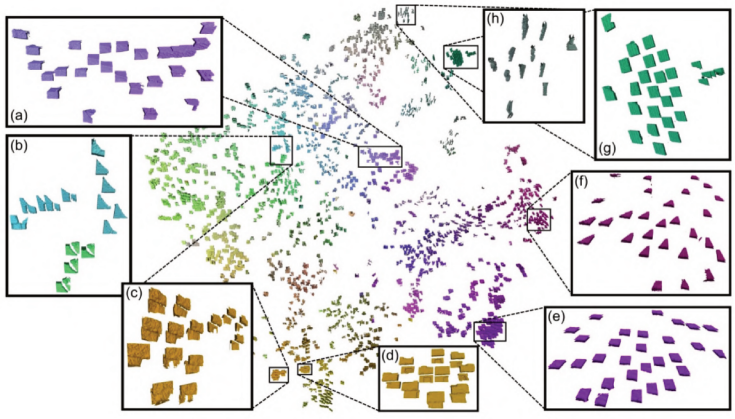

FIGURE 1.36 – Apprentissage d'amers 3D par réseau de neurones convolutionnel 3DMatch par Zeng *et al.* (2016)

nous avons proposé un état de l'art des méthodes de recalage de capteurs en s'appuyant sur la chaîne de traitement typique.

Nous avons présenté les méthodes directes de recalage, s'appuyant sur la totalité des données fournies pour estimer la pose d'un capteur. En contre-pied de ces algorithmes, nous avons introduit les méthodes reposant sur la recherche de données saillantes dans leur voisinage, dites méthodes basées sur des amers. Nous avons finalement fait la distinction entre méthodes reposant uniquement sur l'aspect géométrique des amers pour procéder au recalage de capteurs, et les méthodes utilisant en complément une description du voisinage de ces points saillants, appelée méthodes basée descripteurs.

Si de nombreuses approches existent dans le sujet très actif qu'est le recalage de capteurs, nous avons identifié qu'une grande différence dans la nature des données issues des modalités caméras et LiDAR complique considérablement le processus de recalage. Nous avons vu que recalage de capteurs nécessite d'identifier des informations communes entre les modalités, et invariantes aux conditions d'acquisition. Nous avons ainsi déterminé que les approches s'appuyant sur la géométrie seule peuvent être un point d'entrée intéressant pour aborder ce problème. Dans cette optique, nous nous sommes intéressés à la recherche d'amers dits géométriques, qu'il s'agisse de points, courbes, droites... Dans le chapitre suivant, nous proposons ainsi d'explorer les méthodes d'estimation de pose s'appuyant sur des droites de l'environnement.

Chapitre 2

Contribution à l'Estimation de pose basée lignes : Cas théorique avec mise en correspondance connue

2.1 Introduction

A travers le chapitre précédent, nous avons relevé que le recalage de capteurs pouvait être effectué en considérant la totalité des données à disposition sans distinction, c'est-à-dire en utilisant des méthodes dites directes. Ce recalage peut aussi être réalisé en s'appuyant sur des caractéristiques particulières, se démarquant de leur voisinage, et que l'on appelle amers. Nous avons aussi vu que ces amers peuvent être enrichis d'une description de leur voisinage, appelée descripteur, pour faciliter leur appariement à travers diverses conditions d'acquisition ou plusieurs modalités. Le recalage consiste néanmoins à retrouver des informations communes entre modalités, et à les apparier. Cet appariement constitue alors la base des méthodes permettant d'estimer une transformation liée au changement de référentiel entre les capteurs à recaler.

Nous présentons sans ce chapitre la première contribution de ce travail de thèse, sous la forme d'une méthode d'estimation de pose basée lignes, s'appuyant sur une connaissance *a priori* de la direction verticale.

Le problème de la multimodalité LiDAR - caméra Recaler des capteurs fournissant une information 2D, comme des caméras, avec des capteurs produisant des informations de profondeur, comme les LiDARs, doit se faire en recherchant des éléments communs entre les modalités. Cette tâche est d'autant plus malaisée que la nature des données à appairer est différente. Dans le cas de LiDARs fournissant des nuages de points épars, la difficulté est encore accrue, car les données fournies sont par nature lacunaires. L'une des pistes pour compenser cela

est de s'appuyer directement sur des éléments de la géométrie naturellement invariants au point de vue, et à cette représentation éparse. Les droites, présentes en grand nombre dans des scènes aussi bien naturelles qu'artificielles, sont un support d'information riche en ce sens.

Une part de la littérature du recalage 2D-3D s'appuie sur ces droites, segments ou lignes, pour estimer la pose d'une caméra. On cherche alors à estimer les paramètres permettant de projeter ces droites 3D sur le plan image d'une caméra. Cette estimation des paramètres liés à la projection perspective basée sur n correspondances entre lignes 2D et 3D s'appelle *Perspective-n-Line*, ou PnL.

Deux principales catégories d'algorithmes d'estimation de pose basées lignes se distinguent : les méthodes itératives et les méthodes algébriques.

2.2 Méthodes générales d'estimation de pose basée lignes

Dans le cas d'une acquisition de données à partir de capteurs réels, les correspondances entre amers en deux et trois dimensions ne sont pas nécessairement connues. Suivant la méthode d'acquisition, les deux jeux de données peuvent contenir des amers sans correspondance, ou aberrants (aussi appelés *outliers* en anglais).

Dans ces deux cas, il est nécessaire de procéder à une étape d'appariement des amers 2D et 3D, qui doit pouvoir intégrer une étape de rejet des paires aberrantes avant de pouvoir réaliser l'estimation de pose. Sans cette phase de rejet, même les méthodes de l'état de l'art perdent en précision (Xu *et al.* (2017)).

Dans la plupart des recherches concernant le PnL, on suppose la correspondance entre amers connue. On utilise alors préalablement ou simultanément à l'estimation de pose un procédé de filtrage des mauvaises correspondances. On note toutefois les travaux de David *et al.* (2003) puis Zhang *et al.* (2012b) qui permettent simultanément d'appairer les amers et de d'estimer la pose.

2.2.1 Méthodes générales

Méthodes itératives Les méthodes itératives d'estimation de pose reposent sur la formulation de ces problèmes sous forme d'une minimisation itérative d'une erreur géométrique (erreur de reprojection, ...) ou algébrique (fonction d'erreur spécifique à chaque formulation).

Les travaux de Liu *et al.* (1990) reposent sur l'estimation successive des paramètres des matrices de rotation et de translation d'une caméra comme étant deux sous-problèmes distincts. Leur méthode est appelée R_then_ T (R puis T, en anglais).

Kumar et Hanson (1994) développent plus tard une méthode appelée R_and_T permettant d'estimer simultanément ces deux matrices.

Récemment, les travaux de Zhang *et al.* (2016) ont apporté à l'état de l'art deux nouvelles formulations de la méthode R_and_T en introduisant un paramètre d'incertitude dans les extrémités des segments (segment endpoints, en anglais).

Les algorithmes itératifs ont deux inconvénients majeurs s'ils ne sont pas initialisés correctement : ils convergent lentement, en comparaison avec les méthodes algébriques, et ils ont tendance à converger vers des minima locaux de la fonction d'erreur, rendant la pose estimée parfois loin de la réalité.

Méthodes algébriques Le principe des méthodes algébriques est d'estimer la pose d'une caméra en formulant le problème sous forme d'un système d'équations. Dhome *et al.* (1989) et Chen (1990) proposent tous deux des solutions minimales au problème de PnL à partir de trois correspondances, ou P3L.

La formulation de Dhome *et al.* (1989) ne fonctionne pas avec plus de trois correspondances de lignes, et impose une configuration géométrique spécifique.

Mathématiquement, il existe au minimum huit solutions au problème de P3L dans le cas général (Chen (1990)).

A l'époque, aucune de ces méthodes n'exploite d'autres correspondances pour réduire le nombre de solutions possibles. De plus, ces méthodes sont sensibles au bruit.

Le travail d'Ansar et Daniilidis (2003) apporte une méthode permettant d'exploiter plus de quatre correspondances, tout en réduisant de cette manière le nombre de solutions. Cette approche a cependant une complexité $O(n^2)$, n correspondant au nombre de lignes. L'obtention de la solution est alors très chère en temps de calcul pour un grand nombre de lignes considérées. De plus, cette méthode ne fonctionne pas dans le cas de configurations spécifiques (lignes orthogonales) où la formulation du problème sous forme d'un système polynomial renvoie un ensemble de solutions. L'algorithme tolère mal le bruit et finit par renvoyer des solutions dans l'ensemble des complexes.

Mirzaei et Roumeliotis (2011) ont récemment apporté une amélioration significative à la résolution algébrique du problème de PnL. Avec une méthode de complexité $O(n)$, l'algorithme de Mizraei permet de traiter au minimum trois correspondances de lignes, et est plus robuste au bruit que les précédentes méthodes proposées. Cette méthode repose sur la construction d'un système polynomial à 23 solutions possibles. La solution finale est identifiée grâce à une décomposition en valeurs propres d'une matrice produite à partir des solutions, appelée matrice de multiplication. La pose de la caméra est récupérée en deux étapes. L'orientation est calculée à partir d'une minimisation de l'erreur quadratique, et la position est obtenue grâce à une formulation linéaire des moindres carrés. La matrice de multiplication étant relativement longue à produire, cet algorithme souffre d'un temps de calcul élevé (78ms pour 10 lignes) malgré sa complexité moindre. Il a aussi tendance à converger vers plusieurs solutions.

Zhang *et al.* (2012a) présentent l'algorithme RPnL, fonctionnant avec au minimum quatre lignes et qui est plus précis que l'algorithme de Mirzaei et Roumeliotis (2011). Les correspondances sont divisées en triplets qui servent à résoudre un système polynomial de P3L. La solution optimale est enfin identifiée à partir des racines des dérivées du système, dans le sens des résidus des moindres carrés.

La méthode RPnL a été récemment améliorée par Xu *et al.* (2017) avec l'algorithme ASPnL (Accurate Subset based PnL). Cet algorithme est plus précis sur les jeux de données réduits, mais est très sensible aux mauvais appariements ce qui limite son intérêt dans le cadre d'une application réelle où les appariements aberrants sont courants. Cet algorithme est lui aussi coûteux en temps de calcul (jusqu'à 880ms pour un jeu de données de 1000 lignes).

Méthodes algébriques basées sur la formulation linéaire du PnL : LPnL Plutôt que de formuler le problème d'estimation de pose sous forme d'un système d'équations polynomiales, une partie de la recherche sur le PnL se concentre sur une formulation sous forme d'un système d'équations linéaires. La taille des systèmes d'équations ainsi formés dépend alors du nombre de correspondances entre les lignes considérées. L'ensemble de ces méthodes est regroupé sous l'appellation LPnL, pour *Linear Pespective-n-Line*.

La résolution de systèmes d'équations linéaires est moins coûteuse en calcul que la résolution de systèmes d'équations de plus haut rang. Les méthodes de LPnL sont donc généralement plus rapides à l'exécution que les méthodes de PnL s'appuyant sur une formulation polynomiale, quel que soit le nombre de droites considérées.

Les différentes méthodes de l'état de l'art se distinguent sur deux points : premièrement par le système de coordonnées utilisé pour générer un système d'équations linéaires, et deuxièmement par l'algorithme de résolution de ce système.

La manière la plus directe de résoudre les systèmes d'équations de la formulation LPnL est attribuée à Hartley et Zisserman (2003) et leur algorithme DLT (Direct Linear Transform). Une section de l'ouvrage de Hartley et Zisserman y est consacrée.

L'algorithme DLT permet de transformer les correspondances entre plusieurs lignes en un système homogène d'équations linéaires. Ces correspondances sont formulées à travers des matrices que l'on appelle matrices de mesure. La solution de ce système d'équations se trouve ainsi dans le noyau (ou kernel, ou matrix nullspace en anglais) des matrices de mesures. L'une des conditions impératives à l'application de l'algorithme DLT est que les différentes matrices $\mathbf{M}$ doivent avoir la même norme. Il est donc nécessaire de normaliser les données, sans quoi l'algorithme est trop sensible au bruit et produit des résultats aberrants.

La première méthode de résolution de PnL grâce à l'algorithme DLT est apportée par Hartley et Zisserman (2003). Cette méthode ne formule pas directement le problème de PnL par rapport à des lignes, mais plutôt par rapport à des points inclus sur celles-ci. En effet, si un point repose sur une ligne, cela est aussi vrai pour sa projection. Cette méthode est appelée DLT_Lines par Silva *et al.* (2012), et a besoin d'au moins 6 correspondances pour fonctionner.

Les travaux de Přibyl *et al.* (2016) portent sur l'application de la méthode DLT à des lignes paramétrisées en coordonnées de Plücker. Leur algorithme est appelé DLT_Plucker_Lines, et nécessite au minimum 9 lignes pour fonctionner.

Plus récemment encore, Xu *et al.* (2017) ont développé plusieurs variantes de l'algorithme DLT_Lines, chacune s'appuyant sur des lignes 2D et des points 3D. Leurs méthodes sont ap-

Méthode	Nb. de droites nécessaires	Nb. de solutions
P3L	3	9
Mizraei	3	27
Ansar	4	1
RPnL	4	1
ASPnL	4	1
DLT_lines	6	1
DLT_Plücker_Lines	9	1
DLT_Combines_Lines	5	1
LPnL_Enull	6	1

TABLE 2.1 – Récapitulatif des méthodes algébriques d'estimation de pose 6DDL.

pelées LPnL_DLT_LS, LPnL_DLT_ENull, LPnL_Bar_LS et LPnL_Bar_ENull, et sont une variation de l'algorithme DLT de Hartley et Zisserman (2003) en changeant les systèmes de coordonnées (cartésien et barycentrique avec une paramétrisation Cayley-Gibbs-Rodriguez), et le solveur d'équations (adapté de Lepetit *et al.* (2009)).

Přibyl *et al.* (2017) combinent la redondance structurelle des points et des lignes de manière à réduire le nombre de correspondances nécessaires à la résolution du système linéaire à 5. Un récapitulatif des méthodes algébriques, du nombre de droites requises à l'estimation de pose, et le nombre de solutions associées est présenté sur le tableau 2.1.

2.3 Estimation de pose basée ligne avec connaissance de la direction verticale

2.3.1 État de l'art

De nombreux systèmes sont aujourd'hui équipés de capteurs d'orientation. Ces capteurs permettent d'avoir un *a priori* sur l'orientation du système, et diminuent ainsi la complexité de l'estimation de pose de 6 à 4 degrés de liberté (deux des trois rotations sont connues).

Bien que plusieurs méthodes aient été mises au point concernant l'estimation de pose basée points (PnP) Lepetit *et al.* (2009), Hesch et Roumeliotis (2011), Ferraz *et al.* (2014) avec connaissance de la composante verticale, ce sujet est largement sous-étudié dans le cas de lignes. On note les travaux de Horanyi et Kato (2017a) Horanyi et Kato (2017b), qui traitent de ce sujet en proposant une formulation linéaire et polynomiale de degré 3. Les lignes 3D sont ici représentées par un vecteur directeur et un point. Le formulation polynomiale retourne trois solutions, la solution finale étant celle donnant la plus faible erreur de reprojection.

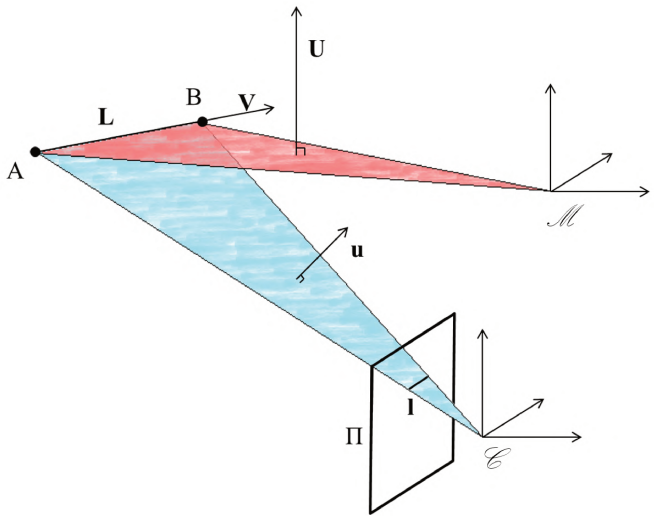

FIGURE 2.1 – Projection d'une ligne 3D **L** sur le plan image Π en une ligne 2D **l**.

2.3.2 Contribution à l'estimation de pose basée ligne avec connaissance de la direction verticale

Préambule Première contribution de ce travail de thèse, une méthode d'estimation de pose avec connaissance de la direction verticale est présentée dans cette sous-section. Cette méthode, publiée dans Lecrosnier *et al.* (2019), s'appuie sur la formulation du problème d'estimation de pose en coordonnées de Plücker introduite par Bartoli et Sturm (2001) et reprise par Přibyl *et al.* (2016), mais propose une solution unique pour la rotation et la translation à l'aide de respectivement deux et trois correspondances de lignes 2D et 3D, en s'appuyant sur une connaissance de la direction verticale.

Coordonnées de Plücker Le problème d'estimation de pose que nous souhaitons résoudre concerne des lignes en deux et trois dimensions. Il est donc nécessaire de représenter ces lignes dans le système de coordonnées homogène utilisé pour représenter les points de l'espace. Cette paramétrisation de lignes en coordonnées homogènes est appelée paramétrisation en coordonnées de Plücker, du nom du mathématicien Julius Plücker (XIXe siècle). Les travaux récents de Bartoli et Sturm (2001) et Přibyl *et al.* (2016) traitent directement de la paramétrisation de lignes 2D et 3D en coordonnées de Plücker pour résoudre un problème d'estimation de pose.

Comme on peut le voir sur la figure 2.1, on définit une droite 3D **L** en coordonnées de Plücker dans un référentiel monde $\mathscr{M}$ à l'aide de sa direction **V** et du plan passant par cette ligne et l'origine de $\mathscr{M}$, dont la normale est donnée par un vecteur **U**. On décrit **L** à l'aide d'un vecteur de dimension 6.

Soient deux points 3D exprimés en coordonnées homogènes et appartenant à la droite **L**, tels que $\mathrm{A} = (a_1, a_2, a_3, a_4)^T$ et $\mathrm{B} = (b_1, b_2, b_3, b_4)^T$. On définit alors $\mathbf{L} = (\mathbf{U}^T, \boldsymbol{V}^T)^T$ telle que :

$$\mathbf{L} = (L_1, L_2, L_3, L_4, L_5, L_6)^T \tag{2.1}$$

avec

$$\begin{aligned} \mathbf{U} &= (a_1, a_2, a_3) \times (b_1, b_2, b_3) \\ \mathbf{V} &= a_4.(b_1, b_2, b_3) - b_4.(a_1, a_2, a_3) \end{aligned} \tag{2.2}$$

où $\times$ est le produit vectoriel. Il est important de noter que $\mathbf{L}$ doit satisfaire la contrainte bilinéaire $\boldsymbol{U}^T.\boldsymbol{V} = 0$.

Notons désormais $\mathrm{L}^{\mathscr{M}}$ l'expression de $\mathbf{L}$ dans le référentiel monde $\mathscr{M}$ et $\mathbf{L}^{\mathscr{C}}$ celle de $\mathbf{L}$ dans le référentiel caméra $\mathscr{C}$. On exprime le changement de référentiel de $\mathbf{L}^{\mathscr{M}}$ vers $\mathbf{L}^{\mathscr{C}}$ à l'aide de la matrice de déplacement $\mathbf{M}$ qui est telle que

$$\mathbf{L}^{\mathscr{C}} = \boldsymbol{M}.\boldsymbol{L}^{\mathscr{M}} \tag{2.3}$$

avec

$$\mathbf{M} = \begin{pmatrix} \mathbf{R} & -\boldsymbol{R}.\mathbf{T}_{[\times]} \\ \mathbf{0_{3\times 3}} & \mathbf{R} \end{pmatrix} \tag{2.4}$$

où $[\times]$ représente la matrice antisymétrique d'un vecteur de dimension 3. $\mathbf{R}$ et $\mathbf{T}$ sont respectivement les vecteurs translations et matrice de rotation permettant le changement du référentiel monde vers le référentiel caméra, et sont respectivement notés $\mathbf{T}_{\mathscr{C}\leftarrow\mathscr{M}}$ et $\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}$ dans la suite de cette thèse.

On nomme

$$\mathbf{l} = (l_1, l_2, l_3) \tag{2.5}$$

la projection de la ligne 3D $\mathbf{L}$ sur le plan image. Par extension, son expression dans le référentiel caméra est $\mathbf{l}^{\mathscr{C}}$. $\mathbf{l}$ est définie comme l'intersection des plans Π et du plan dit plan de projection, défini par sa normale $\mathbf{u}$ dans le référentiel caméra, et passant par l'origine du référentiel $\mathscr{C}$ et $\mathrm{L}^{\mathscr{C}}$. Pour cette raison, le vecteur $\mathbf{u}$ de dimension 3 est suffisant pour définir de manière unique $\boldsymbol{l}$. On a alors

$$\mathbf{l}^{\mathscr{C}} \backsim \mathbf{u}. \tag{2.6}$$

La relation $\sim$ indique une égalité à un facteur d'échelle non nul près.

La projection de $\mathbf{L}^{\mathscr{M}}$ en $\mathbf{l}^{\mathscr{C}}$ peut être exprimée à l'aide de la partie supérieure de la matrice de déplacement $\mathbf{M}$, nommée matrice de projection $\mathbf{P}$ de droite, telle que :

$$\mathbf{P} = \begin{pmatrix} \mathbf{R} & -\mathbf{R}.\boldsymbol{T}_{[\times]} \end{pmatrix}. \tag{2.7}$$

La projection s'exprime alors

$$\mathbf{l}^{\mathscr{C}} \sim \mathbf{P}.\boldsymbol{L}^{\mathscr{M}}. \tag{2.8}$$

On explique cela par le fait que toute droite 3D incluse dans le plan de projection défini par $\mathbf{u}$ peut être projetée en une même droite 2D $\mathbf{l}^{\mathscr{C}}$. Cette paramétrisation est identique à celle de Přibyl *et al.* (2016), qui dérive elle-même de la paramétrisation de Bartoli et Sturm (2001).

Définitions préalables Soient $\mathbf{l}_\mathrm{i} = (l_{i1}, l_{i2}, l_{i3})^T$ la projection sur le plan image Π de la ligne 3D $\mathbf{L}_\mathrm{i} = (L_{i1}, L_{i2}, L_{i3}, L_{i4}, L_{i5}, L_{i6})^T$, $\boldsymbol{l_i}$ et $\boldsymbol{L_i}$ étant les $i^{èmes}$ éléments d'un jeu de données comprenant n lignes. On a alors $\boldsymbol{L}_i = (\mathbf{U}_i^T, \mathbf{V}_i^T)^T$.

La résolution du problème d'estimation dans un référentiel cartésien implique l'estimation de trois rotations, que l'on nommera en angles d'Euler ρ, θ, ψ, ainsi que trois translations t_x, t_y, et t_z.

Estimation de la rotation Dans le chapitre précédent, nous avons vu que certains capteurs, tels les centrales inertielles, peuvent fournir une estimation de la direction verticale du système auquel ils sont attachés. Connaître la direction verticale équivaut à connaître une estimation partielle de l'orientation du système, c'est-à-dire une connaissance *a priori* de deux des trois angles à estimer. Ainsi, on réduit le problème de l'estimation d'une pose de six degrés de liberté (6 DDL) à 4 DDL.

Par définition, $\mathbf{L}_i$ et $\boldsymbol{l}_i$ sont incluses dans le plan de projection. Le vecteur directeur $\mathbf{V}_i$ de la ligne $\mathbf{L}_i$ est donc orthogonal au plan de projection, et ainsi perpendiculaire au vecteur $\boldsymbol{u}$. On en déduit alors la contrainte suivante :

$$(\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}.\mathbf{V}_i^{\mathscr{M}})^T.\boldsymbol{l}_i^{\mathscr{C}} = (\mathbf{V}^{\mathscr{C}})^T.\mathbf{l}_i^{\mathscr{C}} = 0, \tag{2.9}$$

avec $\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}$ la matrice de rotation du référentiel monde vers le référentiel caméra telle que

$$\begin{aligned}\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}} &= \boldsymbol{R}_z.\boldsymbol{R}_y.\boldsymbol{R}_x \\ &= \begin{pmatrix} c_z & -s_z & 0 \\ s_z & c_z & 0 \\ 0 & 0 & 1 \end{pmatrix}. \begin{pmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{pmatrix}. \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_x & -s_x \\ 0 & s_x & c_x \end{pmatrix}\end{aligned} \tag{2.10}$$

avec $c_x = \cos(\rho)$, $s_x = \sin(\rho)$, $c_y = \cos(\theta)$, $s_y = \sin(\theta)$, $c_z = \cos(\psi)$, $s_z = \sin(\psi)$. En posant $\boldsymbol{L_i} = \begin{pmatrix} L_{i1} & L_{i2} & L_{i3} & L_{i4} & L_{i5} & L_{i6} \end{pmatrix}^T$ et $\boldsymbol{l_i} = \begin{pmatrix} l_{i1} & l_{i2} & l_{i3} \end{pmatrix}^T$, on obtient de la contrainte (2.9) le développement suivant :

$$\begin{aligned}&(L_{i5}.(c_x.l_{i2} + l_{i1}.s_x.s_y) - L_{i6}.(l_{i2}.s_x - c_x.l_{i1}.s_y) + L_{i4}.c_y.l_{i1}).c_z \\ &+ (L_{i6}.(l_{i1}.s_x + c_x.l_{i2}.s_y) - L_{i5}.(c_x.l_{i1} - l_{i2}.s_x.s_y) + L_{i4}.c_y.l_{i2}).s_z \\ &+ L_{i6}.c_x.c_y.l_{i3} - L_{i4}.l_{i3}.s_y + L_{i5}.c_y.l_{i3}.s_x = 0.\end{aligned} \tag{2.11}$$

La connaissance *a priori* de la direction verticale implique que les angles ρ et θ, c'est-à-dire que le produit matriciel $\mathbf{R}_y.\mathbf{R}_x$ est connu. Ainsi, les seules inconnues de l'équation (2.9) sont c_z et s_z. Pour deux lignes 3D et leur projections, on peut créer un système linéaire d'équations de rang 2, résoudre c_z et s_z de manière unique, et finalement estimer l'angle ψ. Au delà de deux correspondances 2D/3D, on se trouve en présence d'un système sur-défini, que l'on peut néanmoins résoudre à l'aide d'une méthode traditionnelle d'optimisation, comme celle des moindres carrés.

Dans le cas de données non bruitées, on obtient directement $\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}$. Néanmoins, ce cas de figure est peu vraisemblable si l'on considère des données réelles. On risque alors d'obtenir deux valeurs pour c_z et s_z ne respectant pas la contrainte trigonométrique $c_z^2 + s_z^2 = 1$, et qui une fois introduits dans la relation (2.10) produisent une matrice $\tilde{\boldsymbol{R}}$ qui n'est pas une matrice de rotation, et qui ne respecte pas la contrainte $\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}^T.\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}} = \mathbf{I_3}$. $\boldsymbol{I}_3$ représente la matrice identité. Une étape supplémentaire doit donc être apportée de manière à respecter cette contrainte. L'approche choisie ici repose sur la décomposition en valeurs singulières (DVS, ou SVD en anglais Golub et Reinsch (1970)) de $\tilde{\mathbf{R}}$. Cette méthode permet de décomposer $\tilde{\mathbf{R}}$ en trois matrices $\mathbf{U}$, $\boldsymbol{\Sigma}$ et $\boldsymbol{V}^\star$ telles que

$$\begin{aligned} \mathbf{U}.\boldsymbol{\Sigma}.\mathbf{V}^\star &= \tilde{\mathbf{R}} \\ \mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}} &= \mathbf{U}.\mathbf{V}^{\star T}. \end{aligned} \tag{2.12}$$

$\boldsymbol{U}$ et $\boldsymbol{V}^\star$ représentent respectivement l'ensemble des vecteurs de base orthonormée de $\mathbb{R}^3$, et sont appelés vecteurs de sortie et d'entrée. On veillera ici à ne pas confondre ces termes avec la normale $\mathbf{U}$ et la direction $\mathbf{V}$ d'une droite 3D exprimées en coordonnées de Plücker. Σ est ici une matrice diagonale dont les éléments sont les valeurs singulières de $\tilde{\mathbf{R}}$. La prochaine étape du traitement consiste donc à estimer la translation $\mathbf{T}_{\mathscr{C}\leftarrow\mathscr{M}} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}$.

Estimation de la translation Soit un vecteur $\boldsymbol{a}$ de scalaires défini dans $\mathbb{R}^3$ tel que $\boldsymbol{a} = \begin{pmatrix} a_1 & a_2 & a_3 \end{pmatrix}^T$ et $\boldsymbol{a}_{[\times]}$ la matrice antisymétrique de dimension 3×3 de $\boldsymbol{a}$. L'une des propriétés remarquables de la matrice antisymétrique composée à l'aide d'un vecteur réside dans le fait que son produit avec ce dernier s'annule. On obtient donc l'égalité suivante :

$$\boldsymbol{a}_{[\times]}.\boldsymbol{a} = \mathbf{0_{3\times 1}} \tag{2.13}$$

Dans l'optique d'estimer le vecteur de translation $\boldsymbol{T}_{\mathscr{C}\leftarrow\mathscr{M}}$, on s'appuie sur la propriété (2.13) pour reformuler l'équation (2.8). Pour $i = 1..n$ droites, on obtient alors la relation suivante

$$\boldsymbol{l}_{i[\times]}^{\mathscr{C}}.\boldsymbol{P}.\boldsymbol{L_i}^{\mathscr{M}} = \mathbf{0}_{3\times 1}. \tag{2.14}$$

Pour une correspondance entre une ligne 3D et sa projection, on obtient donc un système de trois équations, dont la première est :

$$
\begin{aligned}
&(-L_{i5}.(l_{i3}.(c_z.s_x - c_x.s_y.s_z) + c_x.c_y.l_{i2}) - L_{i6}.(l_{i3}.(c_x.c_z + s_x.s_y.s_z) - c_y.l_{i2}.s_x)).t_x \\
&+ (L_{i4}.(l_{i3}.(c_z.s_x - c_x.s_y.s_z) + c_x.c_y.l_{i2}) + L_{i6}.(l_{i2}.s_y + c_y.l_{i3}.s_z)).t_y \\
&+ (L_{i4}.(l_{i3}.(c_x.c_z + s_x.s_y.s_z) - c_y.l_{i2}.s_x) - L_{i5}.(l_{i2}.s_y + c_y.l_{i3}.s_z)).t_z \\
&+ L_{i3}.(l_{i3}.(c_z.s_x - c_x.s_y.s_z) + c_x.c_y.l_{i2}) \\
&- L_{i2}.(l_{i3}.(c_x.c_z + s_x.s_y.s_z) - c_y.l_{i2}.s_x) \\
&- L_{i1}.(l_{i2}.s_y + c_y.l_{i3}.s_z) = 0
\end{aligned}
\tag{2.15}
$$

Néanmoins, ce système n'est pas résoluble en l'état, car ses lignes sont linéairement dépendantes : ce système est de rang déficient. Un minimum de trois correspondances est nécessaire pour résoudre la translation de manière unique (Hartley et Zisserman (2003)).

Une fois ces trois correspondances identifiées, on obtient finalement un système linéaire de rang 3, permettant de recouvrer les trois dernières inconnues t_x, t_y et t_z. On réarrange alors ce système d'équations en une matrice $\boldsymbol{M}$, telle que

$$\boldsymbol{M}.\boldsymbol{T}_{\mathscr{C}\leftarrow\mathscr{M}} = \boldsymbol{N}, \tag{2.16}$$

chaque colonne de $\boldsymbol{M}$ étant respectivement exprimée en fonction de t_x, t_y et t_z. La matrice $\boldsymbol{N}$ comprend tous les termes indépendants de t_x, t_y et t_z. On résout finalement l'équation (2.16) à l'aide de la méthode des moindres carrés, c'est-à-dire

$$\boldsymbol{T}_{\mathscr{C}\leftarrow\mathscr{M}} = (\boldsymbol{M}^T.\boldsymbol{M})^{-1}\boldsymbol{M}^T.\boldsymbol{N}.$$

Dans le reste de ce mémoire, on fera référence à cette méthode par l'acronyme VPnL_LS, pour *Vertical Perspective-n-Line.*

Étude de la résolubilité en fonction des configurations spatiales Que l'on considère des droites 2D ou 3D, certaines configurations spatiales communes impliquent des résultats remarquables quant aux opérateurs utilisés dans l'algorithme d'estimation de pose présenté ci-dessus. On considère en particulier les produits vectoriels et scalaires de droites perpendiculaires, coplanaires et parallèles. Le tableau 2.2 regroupe une étude de la solvabilité des systèmes d'équations introduits dans les équations (2.16) et (2.11), lorsque la direction verticale est connue. On y considère des configurations minimales à deux, trois et quatre droites distinctes $\boldsymbol{L_1}$,$\boldsymbol{L_2}$,$\boldsymbol{L_3}$ et $\boldsymbol{L_4}$ aléatoires ou incluses dans deux plans Π_1 et Π_2. $||$ décrit le parallélisme, $\perp$ la perpendicularité et $\subset$ l'inclusion dans un plan.

On remarque que lorsque la direction verticale est connue, la rotation est résoluble pour toute configuration spatiale 3D. En revanche, la direction ne l'est qu'à partir d'une configuration à trois droites, du moment qu'au moins deux droites ne sont pas parallèles.

Configuration 3D	**Rang du système pour résoudre R**	**Rang du système pour résoudre T**	**R résoluble**	**T résoluble**
$\boldsymbol{v_1}$ et $\boldsymbol{v_2}$ aléatoires	2	2	oui	non
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2}$	2	2	oui	non
$\boldsymbol{v_1} \perp \boldsymbol{v_2}$	2	2	oui	non
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2} \mid\mid \boldsymbol{v_3}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}\} \subset \Pi_1$	2	2	oui	non
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2} \mid\mid \boldsymbol{v_3}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}\} \subset \Pi_1$ $\{\boldsymbol{v_3}\} \subset \Pi_2$	2	2	oui	non
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2}$ $\boldsymbol{v_1} \perp \boldsymbol{v_3}$, $\boldsymbol{v_2} \perp \boldsymbol{v_3}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}\} \subset \Pi_1$	2	3	oui	oui
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2}$ $\boldsymbol{v_1} \perp \boldsymbol{v_3}$, $\boldsymbol{v_2} \perp \boldsymbol{v_3}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}\} \subset \Pi_1$ $\{\boldsymbol{v_3}\} \subset \Pi_2$	2	3	oui	oui
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2}$ $\boldsymbol{v_3} \mid\mid \boldsymbol{v_4}$ $\boldsymbol{v_1} \perp \boldsymbol{v_3}$, $\boldsymbol{v_2} \perp \boldsymbol{v_4}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}, \boldsymbol{v_4}\} \subset \Pi_1$	2	3	oui	oui
$\boldsymbol{v_1} \mid\mid \boldsymbol{v_2}$ $\boldsymbol{v_3} \mid\mid \boldsymbol{v_4}$ $\boldsymbol{v_1} \perp \boldsymbol{v_3}$, $\boldsymbol{v_2} \perp \boldsymbol{v_4}$ $\{\boldsymbol{v_1}, \boldsymbol{v_2}\} \subset \Pi_1$ $\{\boldsymbol{v_3}, \boldsymbol{v_4}\} \subset \Pi_2$	2	3	oui	oui

TABLE 2.2 – Solvabilité de la pose avec connaissance de la direction verticale, pour des configurations spatiales de droites 3D particulières

2.3.3 Optimisation de la pose

Dans la méthode VPnL_LS, la rotation et la translation sont estimées séquentiellement. Dans cette optique, une seconde contribution est apportée sous la forme d'une optimisation, elle aussi séquentielle, de la rotation et de la translation. Ce raffinement de la pose est aussi introduit dans Lecrosnier *et al.* (2019), et est présenté dans cette sous-section.

Optimisation de la rotation Notre méthode repose sur la connaissance *a priori* de la direction verticale, aussi appelée vecteur gravité, pour estimer une pose complète. Comme cité précédemment, à travers une mesure de l'accélération et de la vitesse angulaire, des capteurs permettent de donner une estimation du vecteur gravité. Néanmoins, mis à part dans le cas de simulations, l'estimation du vecteur gravité est dépendante de la précision du capteur qui fournit sa mesure. Ainsi, en fonction de la centrale inertielle fournissant cette mesure, on peut s'attendre à une précision angulaire de 0.02 pour les IMU haut de gamme, à 0.5 degrés pour les centrales inertielles d'entrée de gamme (Albl *et al.* (2016)). On comprend alors que l'insertion d'une erreur dans la direction verticale induit une diminution dans la précision de l'estimation de pose. Pour cette raison, une méthode de raffinement de la rotation basée sur l'algorithme d'optimisation de Gauss-Newton est proposée ici. On cherche alors une contrainte géométrique valide pour l'ensemble des lignes 3D et leurs projections, et faisant appel au critère à optimiser, ici une contrainte impliquant l'orientation du système.

Soient $A_i^{\mathscr{M}}$ et $B_i^{\mathscr{M}}$ deux points exprimés en coordonnées cartésiennes dans le référentiel monde $\mathscr{M}$, définissant la $i^{ème}$ ligne 3D d'un jeu de données comprenant $n \in \mathbb{N}$ lignes 3D et leurs n correspondances respectives 2D. Soit $\mathbf{l_i}$ la projection de la ligne 3D $\mathbf{L_i}$ sur le plan image Π, et $\mathbf{V_i^{\mathscr{M}}}$ et $\mathbf{V_i^{\mathscr{C}}}$ la direction de $\mathbf{L_i}$ exprimée respectivement dans le référentiel monde et caméra.

Pour un jeu de n lignes 3D et leurs correspondances, on exprime la contrainte (2.9) sous forme d'un ensemble de n fonctions $\boldsymbol{f} = (f_1, ...f_n)$ dépendant des trois angles de rotation ρ, θ, ψ, tels que définis dans l'équation (2.10). Les fonctions $\boldsymbol{f}$ sont telles que

$$f_i(\boldsymbol{R_{C\leftarrow M}}) = (\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}.\mathbf{V}_i^{\mathscr{M}})^T.\mathbf{l_i} = 0. \tag{2.17}$$

Par itérations successives, et en supposant une valeur initiale $\boldsymbol{\beta}^0$ des paramètres à optimiser telle que $\boldsymbol{\beta^0} = \begin{pmatrix}\rho & \theta & \psi\end{pmatrix}^T$, on recherche un incrément δ faisant varier les paramètres de $\boldsymbol{\beta}$ d'un pas s à un pas $s+1$, tout en minimisant la somme du carré des fonctions $f_{i=1..n}$, tel que

$$\boldsymbol{\beta}^{s+1} = \boldsymbol{\beta}^s + \delta\boldsymbol{\beta}. \tag{2.18}$$

L'algorithme de Gauss-Newton décrit $\delta\boldsymbol{\beta}$ tel que

$$(\mathbf{J_R^T}.\mathbf{J_R})^{-1}\delta\boldsymbol{\beta} = -\mathbf{J_R^T}\boldsymbol{f}. \tag{2.19}$$

$\boldsymbol{J_R}$ étant la matrice jacobienne regroupant les dérivées partielles de chaque équation f_i de $\boldsymbol{f}$ en

fonction des angles ρ, θ et ψ. $\boldsymbol{J_R}$, et telle que :

$$\mathbf{J_R} = \begin{pmatrix} \frac{\partial f_1}{\partial \rho} & \frac{\partial f_1}{\partial \theta} & \frac{\partial f_1}{\partial \psi} \\ ... & ... & ... \\ \frac{\partial f_n}{\partial \rho} & \frac{\partial f_n}{\partial \theta} & \frac{\partial f_n}{\partial \psi} \end{pmatrix}. \tag{2.20}$$

En reformulant les équations 2.18 et 2.19, on pose finalement

$$\boldsymbol{\beta^{s+1}} = \boldsymbol{\beta^s} - (\mathbf{J_R^T}.\mathbf{J_R})^{-1}.\mathbf{J_R}.\boldsymbol{f} \tag{2.21}$$

qui nous permet de raffiner itérativement l'estimation de la rotation.

Optimisation de la translation De la même manière que pour l'optimisation de la rotation, on s'appuie sur l'algorithme de Gauss-Newton pour optimiser l'estimation de la translation. On cherche ici à formuler une contrainte géométrique entre les lignes 3D de notre jeu de données et leurs projections respectives, et faisant appel au critère à optimiser, ici la translation.

On sait que la projection de tout point 3D $\mathbf{A_i}$ situé sur la ligne 3D $\mathbf{L_i}$ se trouve sur la projection $\mathbf{l_i}$ de $\mathbf{L_i}$. Ainsi, de manière similaire à l'équation (2.17), on exprime cette contrainte pour un jeu $\boldsymbol{g} = (g_1, ..., g_n)$ de fonctions dépendant de la translation, telles que

$$g_i(\mathbf{T}_{\mathscr{C}\leftarrow\mathscr{M}}) = (\mathbf{R_{C\leftarrow M}}.\mathbf{A}_i^{\mathscr{M}} + \mathbf{T}_{\mathscr{C}\leftarrow\mathscr{M}})^T.\boldsymbol{l}_i = 0. \tag{2.22}$$

On exprime alors la matrice jacobienne $\mathbf{J_T}$ de $\boldsymbol{g}$, telle que

$$\boldsymbol{J}_T = \begin{pmatrix} \frac{\partial g_1}{\partial t_x} & \frac{\partial g_1}{\partial t_y} & \frac{\partial g_1}{\partial t_z} \\ ... & ... & ... \\ \frac{\partial g_n}{\partial t_x} & \frac{\partial g_n}{\partial t_y} & \frac{\partial g_n}{\partial t_z} \end{pmatrix}. \tag{2.23}$$

On procède finalement à la recherche itérative du minimum de la somme du carré des g_i, avec un incrément des paramètres défini tel que

$$\boldsymbol{\beta^{s+1}} = \boldsymbol{\beta^s} - (\mathbf{J_T^T}.\mathbf{J_T})^{-1}.\mathbf{J_T}.\boldsymbol{g}. \tag{2.24}$$

Ainsi, on procède à l'optimisation de la translation.

Dans le reste de ce document, on nomme VPnL_GN la combinaison de la méthode VPnL_LS et de ces deux optimisations successives des rotations et translations estimées.

Remarques sur l'optimisation par Gauss-Newton De par la nature non linéaire des fonctions $f_{i=1..n}$, l'algorithme de Gauss-Newton ne garantit pas de convergence vers une solution globalement optimale (Mascarenhas (2014)), mais seulement vers un minimum local « accessible »

depuis les paramètres d'initialisation. Pour cette raison, l'optimisation de l'orientation ne peut être employée seule, et nécessite une initialisation proche de la solution finale. En revanche, dans le cas de l'optimisation de la translation subséquente à l'estimation de l'orientation, les composantes de la rotation sont présumées connues. On se retrouve donc dans le cas d'un système linéaire, dont la convergence vers une solution globalement optimale est garantie par la méthode de Gauss-Newton. On note toutefois que la solution optimale retrouvée ici est optimale vis-à-vis de l'estimation de l'orientation fournie. Si une grande erreur est présente dans cette première orientation, la translation raffinée sera estimée au regard d'une orientation erronée, donc éloignée de la position réelle du système.

Comme dans de nombreux algorithmes itératifs, le nombre d'itérations maximales doit être soit régulé par un critère d'arrêt, soit fixé manuellement. Nous choisissons ici une approche hybride. Certains algorithmes itératifs d'optimisation ont tendance à stagner ou osciller très faiblement lorsque les paramètres sont très proches d'un minimum local ou global. Pour cette raison, l'arrêt de ces algorithmes de raffinement de pose s'effectue lorsqu'une limite dure de 20 itérations est dépassée, ou lorsque l'on atteint un point stationnaire (variation de l'ordre de $10^{-5}m$ ou $10^{-5\circ}$).

2.4 Expérimentations

Après avoir posé les bases théoriques de cette méthode d'estimation de pose, couplée ou non à une optimisation des paramètres, il convient d'en démontrer la validité expérimentalement. Pour ce faire, on évalue dans un premier temps les performances sur un jeu de données de simulation, permettant de maîtriser pleinement les paramètres expérimentaux. Dans un second temps, ces algorithmes sont validés sur des données réelles.

2.4.1 Métriques

L'objectif des algorithmes de PnL est d'estimer la pose d'une caméra à l'aide de correspondances connues entre lignes 2D et lignes 3D. On cherche donc à mesurer la précision de l'estimation de l'orientation et de la position de la caméra. Ainsi on considère pour le reste de ce chapitre deux métriques, qui sont l'erreur d'estimation de positon, ou erreur de translation, et l'erreur d'estimation de rotation.

L'erreur de translation est donnée sous forme de taux d'erreur de translation, plutôt que de distance. On comprend aisément qu'une erreur de quelques centimètres sur un déplacement de plusieurs mètres est moindre comparée à une erreur de quelques centimètres sur un déplacement du même ordre de grandeur. Plus formellement, ce taux d'erreur de translation est donné par le produit de la distance euclidienne entre la position réelle et la position estimée. Cette distance est ensuite normalisée par la norme de la position réelle. Ainsi, pour deux positions $\boldsymbol{P_{estimé}} = \begin{pmatrix} p_1 & p_2 & p_3 \end{pmatrix}^T$ $\boldsymbol{Q_{réel}} = \begin{pmatrix} q_1 & q_2 & q_3 \end{pmatrix}^T$ exprimées dans le référentiel monde en coordonnées

cartésiennes, le taux d'erreur de translation entre ces deux points $t(\boldsymbol{P_{estimé}}, \boldsymbol{Q_{réel}})$ est donné par

$$t(\boldsymbol{P_{estimé}}, \boldsymbol{Q_{réel}}) = \frac{\sqrt{(q1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2}}{\sqrt{{q_1}^2 + q_2^2 + q_3^2}}.100. \tag{2.25}$$

Concernant l'erreur de rotation, plusieurs choix sont possibles en fonction de la représentation choisie pour l'orientation. Comme le montre l'équation (2.10), la composition des rotations est réalisée dans ces travaux à partir des angles d'Euler. La connaissance *a priori* de la direction verticale signifie que sur trois angles (ρ, θ, ψ), deux sont connus (ρ, θ), et ψ reste à calculer.

Dans le jeu de données de simulation détaillé ci-après, on introduit des erreurs dans les angles supposés connus. Dans le calcul d'erreur, on recherchera alors à ne mesurer l'erreur de rotation que sur l'angle restant à déterminer. On calcule donc dans ce cas l'erreur de rotation en réalisant la valeur absolue de la différence de l'angle estimé et de l'angle réel. Cette mesure d'erreur sur un seul angle a pour inconvénient d'ignorer l'erreur introduite sur les angles (ρ, θ) lorsque l'on simule une défaillance de l'IMU.

Dans le jeu de données réelles, aucune erreur n'est introduite dans les rotations. On cherche donc à mesurer un écart angulaire sur les trois angles entre une valeur estimée et une vérité terrain. On utilise alors la différence angulaire entre les matrices de rotation. Cette différence angulaire Λ (en radians) entre deux matrices de rotations $\boldsymbol{R_1}$ et $\boldsymbol{R_2}$ tire son origine de la représentation vectorielle des rotations, et s'exprime

$$\Lambda = arccos(\frac{tr(\boldsymbol{R_1}.\boldsymbol{R_2^T}) - 1}{2}). \tag{2.26}$$

2.4.2 Simulation

Introduction Dans le cas du test en simulation d'un algorithme de *Perspective-n-Line* avec connaissance de la direction verticale, on cherche à maîtriser principalement quatre paramètres expérimentaux, à savoir :

- Les paramètres extrinsèques de la caméra, c'est-à-dire la pose, ou encore la position et l'orientation de la caméra dans un référentiel de référence, ici le référentiel monde R_M ; on fait varier ce paramètre durant chaque test de la simulation de manière à valider le comportement de l'algorithme pour toutes les poses possibles,
- Les paramètres intrinsèques de la caméra, c'est-à-dire les spécificités liées à la nature optique de la caméra, comme les déformations dues aux lentilles,
- Le bruit des capteurs, qui fait référence aux erreurs de mesure des capteurs, comme par exemple une imprécision dans la mesure du vecteur gravité ; on évalue la robustesse de nos algorithmes en simulant une centrale inertielle à bas coût, puis en simulant une défaillance croissante de cette IMU,
- Le bruit lié au traitement des données, induisant une différence entre les droites réelles et les droites extraites de manière automatique ; on introduit ici des imprécisions en déplaçant

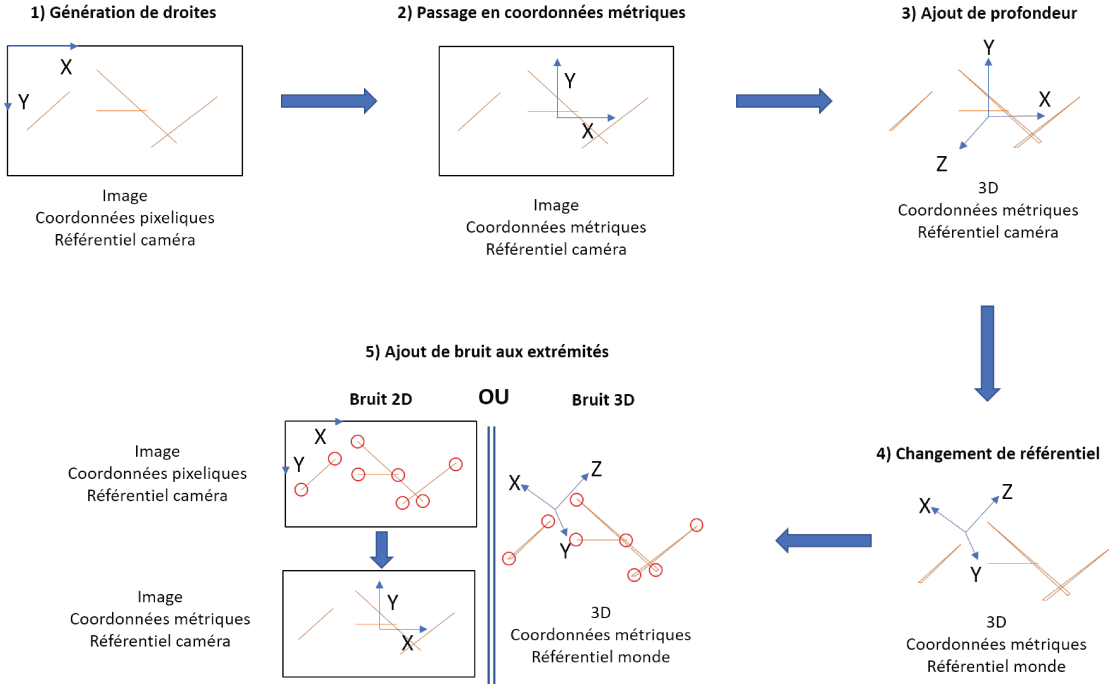

FIGURE 2.2 – Procédure de génération du jeu de données de simulation

les points 2D ou 3D définissant les droites du jeu de données.

Protocole expérimental

Dans ce jeu de données de simulation, les caractéristiques de notre caméra virtuelle sont calquées sur une caméra USB à bas coût, telle que celle utilisée avec notre jeu de données réelles. Ainsi, les conditions simulées sont au plus proche des conditions réelles, et on évite un écart de performances entre expérimentation simulée et réelle.

Pour cette expérimentation en simulation, on fixe la caméra comme possédant un capteur de 640 par 480 pixels, une distance focale de 655 pixels et aucune distorsion radiale ni tangentielle.

La figure 2.2 reprend les étapes de la génération des données de simulation décrites ci-après.

1. Les droites du jeu de données sont générées tout d'abord en deux dimensions, sur le plan image, à l'aide de paires de points tous séparés d'une distance d'au moins 70 pixels.
2. Ces points, initialement exprimés en coordonnées pixeliques, sont transformés en coordonnées métriques à l'aide des paramètres intrinsèques de la caméra.
3. On donne une profondeur aléatoire à ces points, comprise entre 2m et 20m. Cette profondeur est typique d'une application routière en zone urbaine.
4. On fixe aléatoirement une transformation $\boldsymbol{Tr}_{\mathscr{M}\leftarrow\mathscr{C}}$ qui permet d'exprimer les points 2D du référentiel caméra vers le référentiel monde. Cette transformation est bornée à une

translation comprise en 2m et 20m, et un changement d'orientation entre 0 rad et 2πrad. On impose ainsi la pose de caméra à retrouver.

5. Suivant les cas de tests, on simulera une extraction imparfaite d'amers en ajoutant un bruit gaussien aux points servant de support aux droites 2D ou 3D. Dans le cas du bruit 3D, on agit directement sur les points exprimés dans le référentiel monde, en les déplaçant suivant les trois axes, d'une distance de σ_{3D} mm. Pour les points 2D, ce processus inclut une étape supplémentaire. On doit bruiter les points de l'image, en coordonnées pixeliques, que l'on déplace de σ pixels suivant les deux axes. On ré-exprime ensuite ces points en coordonnées métriques.

Finalement, on estime la pose de la caméra à l'aide de la méthode de PnL, et on calcule les métriques présentées dans la sous-section 2.4.1.

Évaluer la robustesse de notre algorithme sans élément de comparaison ne serait pas rigoureux. On compare donc nos résultats avec les différents algorithmes d'estimation de pose à l'état de l'art. Trois algorithmes d'estimation de pose basée lignes sont donc ici considérés :

— NPnLUpL, de Horanyi et Kato (2017a) qui, comme notre algorithme VPnL_LS, s'appuie sur une connaissance *a priori* de la direction verticale fournie par une IMU pour estimer la pose d'une caméra.
— NPnLUpC, de Horanyi et Kato (2017b) qui est une variante de NPnLUpL basée sur des équations quadratiques
— L'algorithme ASPnL, de Xu *et al.* (2017), qui ne s'appuie pas sur la direction verticale, et calcule la pose complète.

Étant donné que l'on compare ici des algorithmes reposant sur la direction verticale et d'autres calculant une pose complète, on ajoute un bruit constant de 0.5 degrés sur la direction verticale pour plus d'équité. Ce bruit correspond à celui d'une IMU d'entrée de gamme. Lorsque cela n'est pas précisé, cette erreur est ajoutée en entrée des fonctions d'estimation de pose, sur un des deux angles définissant la direction verticale.

Pour chaque incrément de bruit ou de données aberrantes insérés, le nombre de tests effectués est de 2000. La valeur médiane des erreurs est retournée, car certains des algorithmes testés expérimentalement produisent dans des cas marginaux des erreurs particulièrement importantes, non représentatives de leur comportement standard et induisant une valeur moyenne elle non plus non représentative.

Robustesse au bruit 2D et 3D

Le premier test considéré est celui de la robustesse au bruit 2D et 3D. On simule ici une extraction d'amers imprécise en déplaçant les extrémités des points 2D ou 3D définissant les droites utilisées dans les algorithmes de PnL. Comme le montre la figure 2.3, on ajoute un bruit croissant, dans un premier temps sur les points 3D allant de 0 à 500mm, puis dans second temps sur les points 2D de 0 à 20 pixels, puis on mesure le taux d'erreur médian en translation, et le

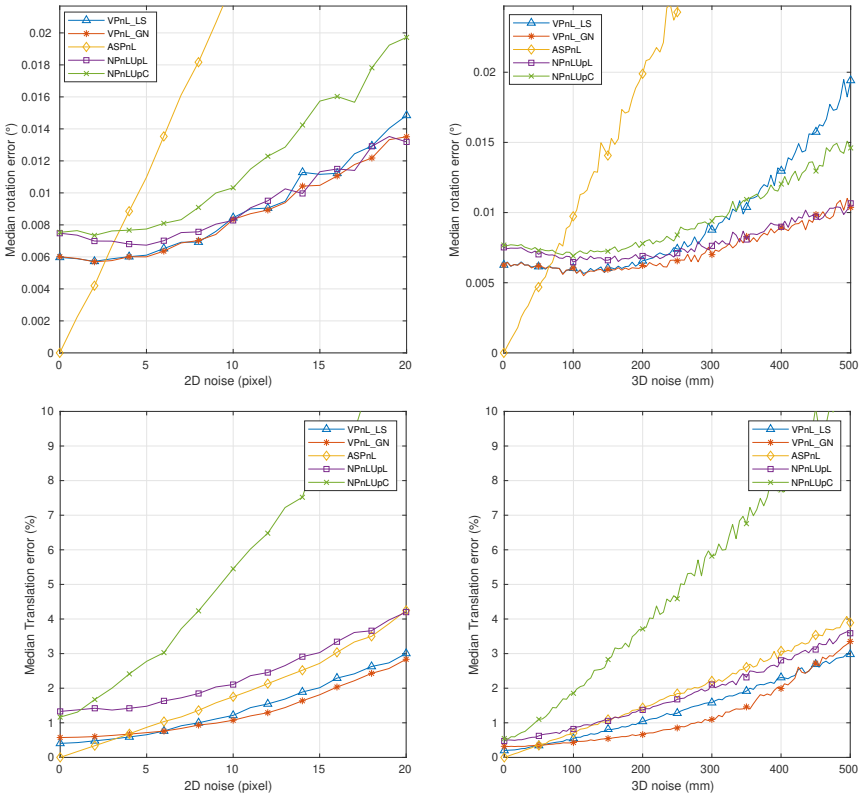

FIGURE 2.3 – Robustesse au bruit 2D et 3D en présence de 0.5° d'erreur sur le vecteur gravité

l'erreur médiane de rotation. Les incréments de bruits 2D et 3D sont respectivement des pas de 1 pixel et 5 mm. Il est à noter que la limite haute de ces bruits est particulièrement extrême.

Sans surprise, l'algorithme ASPnL surpasse les autres algorithmes dans une configuration avec un faible bruit 2D et 3D. En effet, il est par nature insensible aux erreurs introduites dans le vecteur gravité. Il délivre néanmoins des performances moindres comparé aux algorithmes s'appuyant sur le vecteur gravité lorsque le bruit augmente, en terme d'erreur de rotation. Dans cette catégorie, notre algorithme VPnL_GN fournit les meilleures performances, en concurrence directe avec NPnLUpL. Tous deux fournissent une erreur de rotation de l'ordre de 0.01° à 500mm de bruit 3D ou 14 pixels de bruit 2D, quand les performances de notre VPnL_LS sont en retrait, tout comme NPnLUpC.

Dans le cas du taux d'erreur en translation, les résultats placent nos deux algorithmes en tête des performances, avec un taux d'erreur de l'ordre de 3% quand NPnLUpL et ASPnL arrivent à 4% en limite haute du bruit testé.

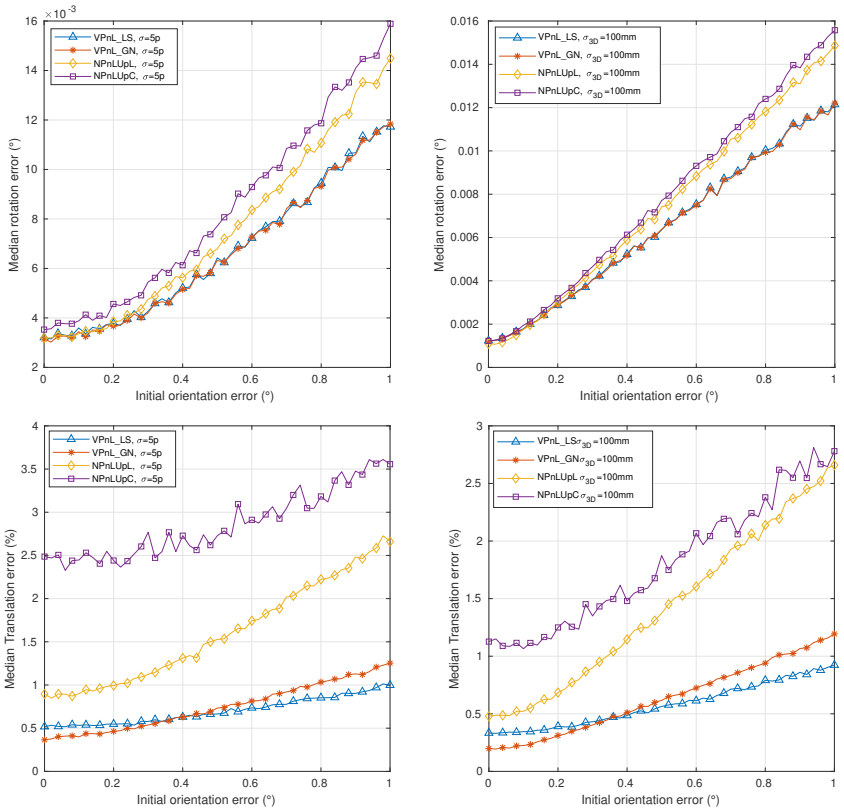

FIGURE 2.4 – Impact de l'erreur d'orientation initiale

Robustesse à l'erreur d'orientation initiale

Pour ce test, on fige les erreurs 2D et 3D respectivement à 5 pixels et 100mm, et on fait varier l'erreur induite dans le vecteur gravité de 0° à 1° par pas de 0.02°. Il est à noter que ASPnL est exclu de ce test, car ne s'appuyant pas sur la direction verticale pour estimer la pose. Pour cette catégorie de bruit, nos deux algorithmes sont définitivement plus performants en terme d'erreur en translation pour les gammes de bruit 2D et 3D sélectionnés. VPnL_GN propose un léger avantage pour un vecteur gravité peu bruité, tandis que VPnL_LS est plus performant avec un bruit plus haut. En terme d'erreur en rotation, les résultats sont similaires pour un bruit faible, mais nos deux algorithmes fournissent de meilleures performances pour un bruit plus élevé. L'optimisation de la rotation ne fournit pas dans ce cas d'avantage.

Impact du nombre de lignes Pour ce test, on fige les valeurs de bruit 2D et 3D respectivement à $\sigma = 10$ pixels et $\sigma_{3D} = 100$mm. On fait varier le nombre de droites considérées par les différents algorithmes de PnL testés de 4 (limite basse de ASPnL) à 40, qui est un nombre de droites cohérent dans une scène urbaine. On évalue alors les performances suivant les métriques

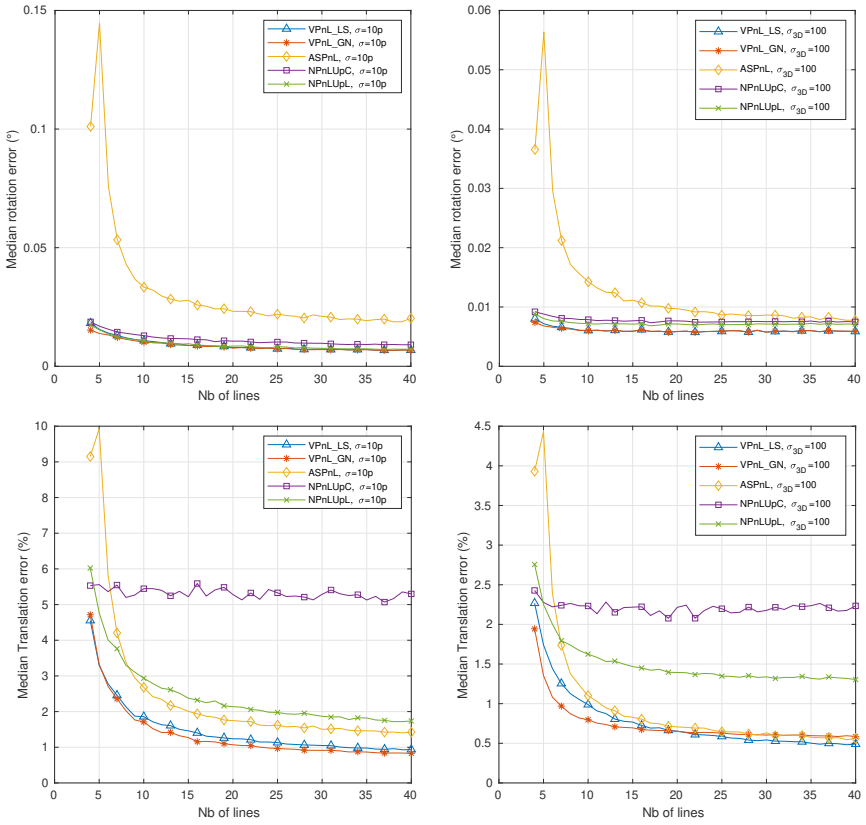

FIGURE 2.5 – Impact du nombre de lignes sur les performances

décrites précédemment. Globalement, les performances de tous les algorithmes s'accroissent à des rythmes relativement similaires, quelles que soient les méthodes considérées. Concernant l'erreur en rotation, nos algorithmes fournissent les meilleurs résultats, quoique la différence de performance avec les méthodes NPnLUpL et NPnLUpC soient marginales. ASPnL fournit en comparaison des résultats nettement dégradés pour le bruit 2D, et rattrapant les performances de NPnLUpL et NPnLUpC seulement pour une quarantaine de droites avec bruit 3D.

Dans le cas de l'erreur en translation, nos deux algorithmes tirent leur épingle du lot, et sont seulement rattrapés par ASPnL, encore une fois aux alentours de la quarantaine de lignes, dans le cas du bruit 3D.

Robustesse aux mauvais appariements

Si la correspondance entre lignes 3D et leur projection sur le plan image est supposée connue, il est néanmoins instructif d'observer le comportement des algorithmes comparés ci-dessus dans le cas où un certain nombre de correspondances sont incorrectes. Pour ce faire, on procède à la génération des droites 2D et 3D de la même manière que présenté sur la figure 2.2. Pour un

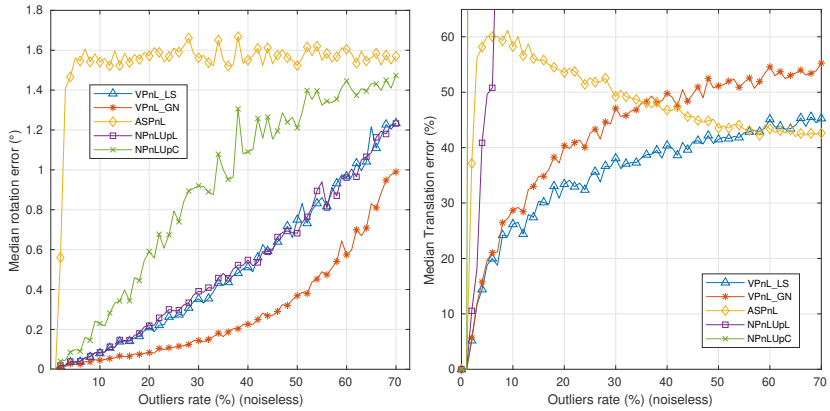

FIGURE 2.6 – Robustesse aux mauvais appariements 2D/3D. Cas sans bruit 2D,3D ni sur la direction verticale

jeu comportant un nombre plus important de lignes, 40 ici, on insère des données aberrantes juste avant la phase d'estimation de pose, en remplaçant une partie des points 3D par des points aléatoires, eux aussi compris dans un rayon de 2m et 20m autour de la caméra. Les données correctement appariées ne subissent aucun bruit 2D, 3D, ni de dégradation de la direction verticale.

En observant les résultats de la figure 2.6, on remarque que même pour un nombre faible de droites mal appairées, l'erreur en translation est très importante. Dès 20% de données aberrantes, donc 8 droites 3D aléatoires pour 32 correctes, l' erreur en translation pour l'algorithme le plus performant, VPnL_LS, est de l'ordre de 33%. Cela correspond à un écart de position moyen de 9.61m et médian de 7.19m. On comprend donc qu'une telle erreur n'est pas compatible avec une application de robotique mobile.

Discussion Cette première expérimentation valide les performances et la robustesse de nos deux algorithmes d'estimation de pose basée ligne, VPnL_LS et VPnL_GN. Que l'on considère un bruit 2D, 3D, un nombre de droites faible ou plus important, ainsi qu'une erreur de mesure initiale du vecteur gravité, nos deux algorithmes proposent des résultats meilleurs ou équivalents aux algorithmes de l'état de l'art. De manière générale, la méthode VPnL_LS est plus performante dans l'estimation de la translation pour un jeu de données bruité. En complément, notre algorithme VPnL_GN fournit des performances équivalentes ou supérieures dans le cadre de l'estimation de l'orientation en conditions similaires. En présence de mauvais appariements de droites, cette fois-ci sans aucun bruit, cette tendance se confirme, et on observe nettement l'intérêt d'une optimisation de la rotation, tandis que l'optimisation de la translation fournit des performances en retrait.

Ces résultats fournissent une base suffisante pour envisager l'expérimentation sur des données réelles.

FIGURE 2.7 – Une caméra infrarouge T40S, de la marque *Vicon*

2.4.3 Jeu de données IRSEEM

Après cette première validation de nos algorithmes en simulation, il convient de valider ces méthodes sur un jeu de données réelles. Dans cette optique, nous avons réalisé l'acquisition d'un jeu de données expérimental au sein du laboratoire de navigation autonome, ou LNA, de l'IRSEEM.

Ce jeu de données est composé d'un ensemble de poses d'une caméra mobile repérées dans un nuage de points, et a été constitué en deux temps. Tout d'abord, une première acquisition de données a été effectuée, permettant le calibrage complet du système. Ce calibrage permet de déterminer les transformations liées aux divers changements de référentiels du jeu de données. Ensuite, une seconde acquisition permet de capturer le jeu de données utilisé cette fois pour la validation de nos deux algorithmes VPnL_LS et VPnL_GN.

Équipement et collecte de données

Système *Vicon* Le LNA est équipé à demeure d'un système de capture de mouvement du fabricant Vicon. Il comprend 20 caméras T40S (cf. figure 2.7) synchronisées permettant une acquisition de la pose d'un système muni d'un motif de marqueurs réfléchissants (cf. figure 2.8), à une fréquence de 100 Hz. Ce système fournit la vérité terrain de l'expérimentation avec une précision millimétrique (Merriaux *et al.* (2017)). Dans la suite de ce document, on appellera le référentiel lié au système d'acquisition *Vicon* "référentiel *Vicon*".

Système LiDAR Leica Les algorithmes d'estimation de pose présentés précédemment nécessitent des droites 3D et leurs correspondances 2D pour fonctionner. Dans ce jeu de données, les droites 3D sont extraites manuellement d'un nuage de points 3D dense acquis grâce à un LiDAR *Leica ScanStation C10*. Ce LiDAR peut reconstituer une carte des distances d'un environnement complexe en effectuant plusieurs acquisitions à différents endroits. Les différents nuages de points acquis sont ensuite alignés avec un algorithme de type ICP (*iterative closest point*, par Chen et Medioni (1992)) et forment ainsi un nuage de points final.

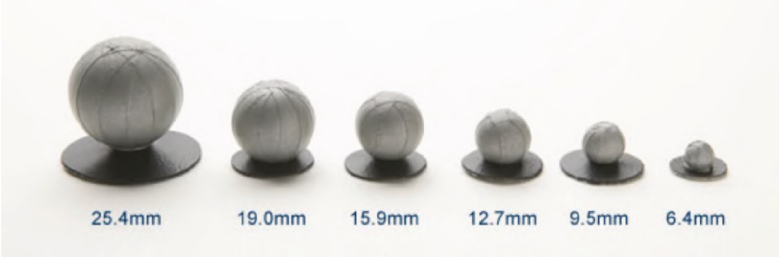

FIGURE 2.8 – Marqueurs du système Vicon de capture de mouvement. Les marqueurs utilisés pour l'expérience sont ceux de 15.9mm. Image extraite de Merriaux (2016)

Le nuage de points final, que l'on simplifie par nuage de points pour le reste de cette section, est exprimé dans un référentiel différent du référentiel *Vicon*. On appelle ce référentiel le référentiel LiDAR. Il est nécessaire de calculer une transformation rigide liant ces deux référentiels afin de pouvoir exprimer les lignes 3D dans le référentiel *Vicon*. Cette transformation est calculée lors d'une phase de calibrage décrite ci-après.

Système caméra Les droites 2D sont extraites manuellement d'un ensemble d'images acquises avec une caméra *Logitech Quickcam 4000*. Ce modèle est une webcam à bas coût, munie d'un capteur de 640 par 480 pixels, telle que l'on peut en trouver couramment dans la grande distribution. Les images acquises par cette caméra comportent des déformations radiales et tangentielles négligeables.

Afin d'obtenir la pose réelle de la caméra, cette dernière est liée rigidement à un support muni de marqueurs Vicon comme le montre la figure 2.10. La pose du support de la caméra est enregistrée de manière synchronisée avec l'acquisition des images. Après une phase de calibrage décrit plus bas, on est capable de calculer la transformation rigide entre la caméra et son support, et donc d'exprimer la pose réelle de la caméra dans le référentiel Vicon (cf. figure 2.11).

L'objectif est de connaître la pose de la caméra dans le référentiel Vicon, pour chaque image acquise. De cette manière, on peut comparer la pose réelle de la caméra avec l'estimation obtenue par notre méthode. La figure 2.11 reprend tous les changements de référentiels.

Le support de la caméra est muni d'un ensemble de marqueurs Vicon. Ces marqueurs permettent d'obtenir la pose du support dans le référentiel Vicon, mais pas directement la pose de la caméra : la transformation rigide entre le centre optique de la caméra et le support est inconnue, mais peut être estimée. Pour la calculer, il est nécessaire de passer par un référentiel intermédiaire. On utilise pour ce faire une mire de calibrage qui comporte un motif d'échiquier, visualisable par la caméra, et un motif de marqueurs Vicon. Cette mire est visible sur la figure 2.9. Cette combinaison d'éléments de plusieurs modalités permet de faire le lien entre les deux référentiels.

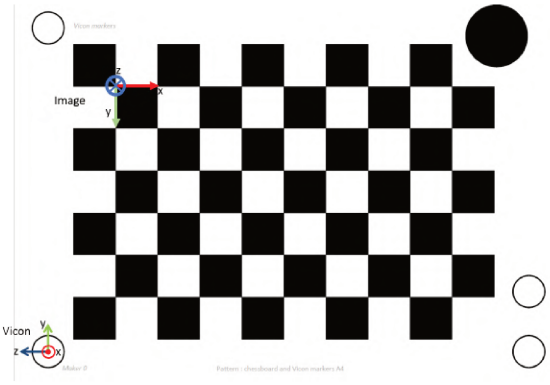

FIGURE 2.9 – Mire de calibrage caméra - Vicon

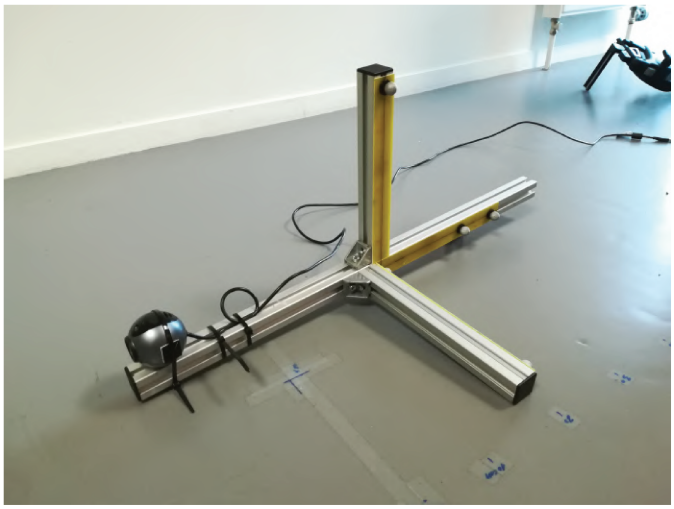

FIGURE 2.10 – Caméra montée sur un support rigide, lui-même muni de marqueurs Vicon

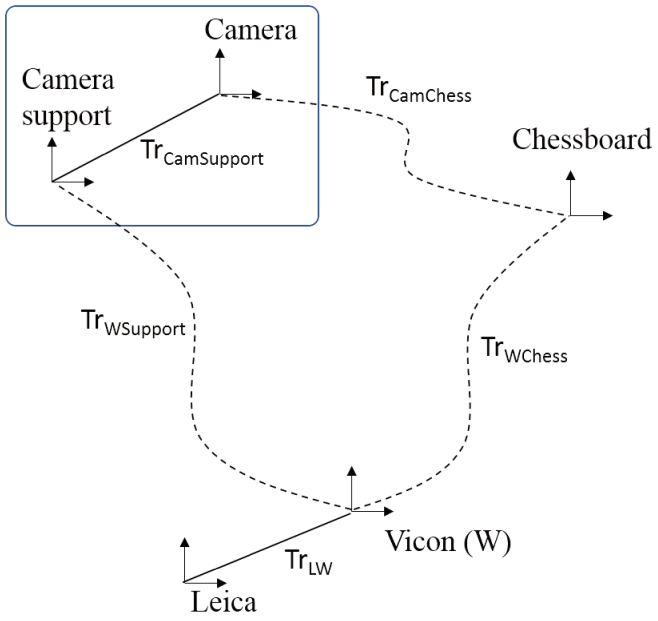

FIGURE 2.11 – Synoptique expérimental du jeu de données IRSEEM

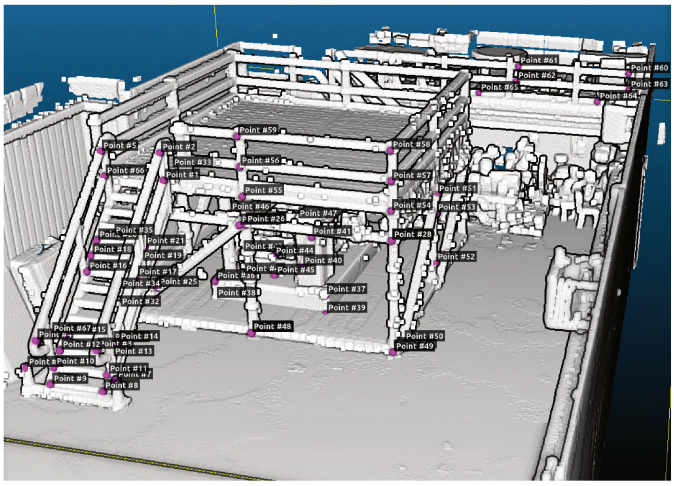

FIGURE 2.12 – Points 3D extraits du nuage de points Leica

Calibrage Vicon LiDAR

La méthode la plus directe pour effectuer ce calibrage consiste à mettre en correspondance des points 3D extraits dans chacune des modalités. Ainsi, pendant l'acquisition du jeu de données, une vingtaine de marqueurs Vicon ont été disposés dans la scène à des positions aisément discriminantes de l'environnement. Connaissant la correspondance entre ces marqueurs et leur position individuelle dans le nuage de points, il est possible d'estimer la transformation rigide entre les référentiels LiDAR et Vicon.

Méthode des moindres carrés La première méthode possible est celle des moindres carrés. Elle consiste à minimiser la somme du carré de la distance entre ces paires de points.

Soient deux jeux de n points $\mathbf{A}$ et $\mathbf{B}$ exprimés en coordonnées homogènes dans deux référentiels respectifs tels que

$$\mathbf{A} = \begin{pmatrix} x_{a1} & \dots & x_{an} \\ y_{a1} & \dots & y_{an} \\ z_{a1} & \dots & z_{an} \\ 1 & \dots & 1 \end{pmatrix} \text{ et } \mathbf{B} = \begin{pmatrix} x_{b1} & \dots & x_{bn} \\ y_{b1} & \dots & y_{bn} \\ z_{b1} & \dots & z_{bn} \\ 1 & \dots & 1 \end{pmatrix} \tag{2.27}$$

avec

$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix} .\mathbf{A} = \boldsymbol{B} \tag{2.28}$$

On note $\mathbf{R}$ et $\mathbf{T}$ respectivement matrice de rotation et vecteur de translation, les paramètres de la transformation de référentiel A vers le référentiel B.

On sait que l'équation (2.28) équivaut à

$$\left[\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix} .\mathbf{A} \right]^T = \mathbf{A}^T . \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix}^T = \mathbf{B}^T . \tag{2.29}$$

Nous pouvons alors formuler l'estimation des paramètres de la transformation au sens des moindres carrés :

$$\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0_{1\times 3}} & 1 \end{pmatrix}^T = (\boldsymbol{A}.\mathbf{A}^T)^{-1}.\mathbf{A}.\mathbf{B}^T . \tag{2.30}$$

Bien que numériquement valide, cette méthode optimise les paramètres un à un sans tenir compte des contraintes entre chacun des paramètres. On peut ainsi obtenir une matrice de rotation pour laquelle la contrainte trigonométrique n'est pas valide (*ie.* $\det(\mathbf{R} \neq 1)$)).

Dans ce cas, il est nécessaire d'orthonormaliser $\mathbf{R}$ en une matrice $\tilde{\boldsymbol{R}}$. Cette opération peut s'effectuer par décomposition en valeurs singulières :

$$\mathbf{R} = \mathbf{U}.\mathbf{\Sigma}.\mathbf{V}^{*T} \text{ et } \tilde{\mathbf{R}} = \mathbf{U}.\mathbf{V}^{*}. \tag{2.31}$$

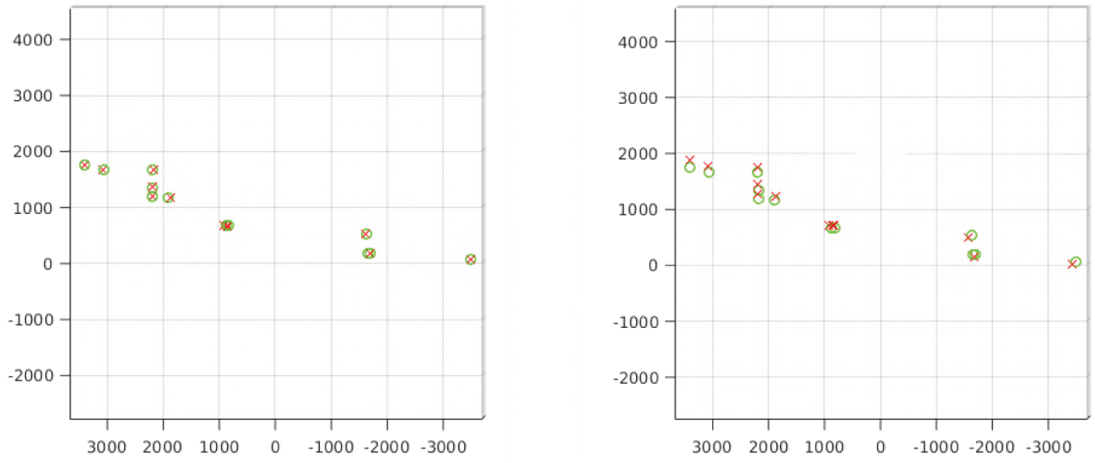

FIGURE 2.13 – Alignement des deux ensembles de points appairés pour le calibrage LiDAR-Vicon par la méthode des moindres carrés avant orthonormalisation (gauche) et après orthonormalisation (droite). Échelle en mm.

Pour une matrice d'entrée $\mathbf{M}_{m\times n}$ (ici, $\mathbf{M} = \mathbf{R}$) à coefficients réels, on note $\mathbf{V}^*$ un ensemble des vecteurs de la base orthonormée de $\mathbb{R}^n$ dits "d'entrée", $\mathbf{U}$ un ensemble des vecteurs de la base orthonormée de $\mathbb{R}^m$ dits "de sortie" et $\mathbf{\Sigma}$ la matrice dont les coefficients diagonaux sont les valeurs singulières de la matrice $\mathbf{M}$.

L'orthonormalisation augmente cependant l'erreur moyenne d'alignement des points, car on modifie directement neuf des 12 paramètres estimés par la méthode des moindres carrés afin de forcer la contrainte trigonométrique sur $\mathbf{R}$.

La figure 2.13 montre les résultats avant et après orthonormalisation de $\mathbf{R}$. Le tableau 2.3 récapitule ces résultats numériquement.

Décomposition en éléments simples Une autre solution pour calculer la transformation du référentiel LiDAR vers le référentiel Vicon passe directement par une décomposition en éléments simples. L'algorithme 2.1 présente le processus.

Soient deux jeux de n points $\mathbf{A}$ et $\mathbf{B}$ exprimés en coordonnées cartésiennes dans deux référentiels respectifs tels que

$$\mathbf{A} = \begin{pmatrix} x_{a1} & \dots & x_{an} \\ y_{a1} & \dots & y_{an} \\ z_{a1} & \dots & z_{an} \end{pmatrix} \text{ et } \mathbf{B} = \begin{pmatrix} x_{b1} & \dots & x_{bn} \\ y_{b1} & \dots & y_{bn} \\ z_{b1} & \dots & z_{bn} \end{pmatrix} \tag{2.32}$$

avec

$$\mathbf{R}.\mathbf{A} + \mathbf{T} = \mathbf{B}. \tag{2.33}$$

Chaque jeu de points est translaté de manière à ce que la coordonnée du point moyen soit celle

du centroïde du jeu considéré. On obtient deux nouveaux jeux de points $\tilde{\mathbf{A}}$ et $\tilde{\mathbf{B}}$. On construit ensuite une matrice $\mathbf{H}$ telle que

$$\mathbf{H} = \tilde{\mathbf{A}}.\tilde{\mathbf{B}}^T \ . \tag{2.34}$$

La rotation $\mathbf{R}$ est récupérée simplement par décomposition en éléments simples (Singular Value Decomposition, ou SVD, en anglais), et l'estimation de la translation est effectuée à partir de $\mathbf{R}$ et des centroïdes de $\mathbf{A}$ et $\mathbf{B}$.

Algorithme 2.1 Estimation de transformation rigide entre deux nuages de points par décomposition en éléments simples

variables :
$\mathbf{A}$ = points Vicon, $\mathbf{B}$ = points LiDAR
fonction : [x y z] $\leftarrow$ centroide($\mathbf{matrice}_{n\times 3}$)
[x y z] = $\sum_{i=1}^{n} \frac{\mathbf{matrice}_{i\times 3}}{n}$
fonction : [$\mathbf{U}$ $\mathbf{S}$ $\mathbf{V}^{*\,T}$] $\leftarrow$ SVD($\mathbf{matrice}_{n\times n}$)

Algorithme :
$\tilde{\mathbf{A}}$ = $\mathbf{A}$ - centroide($\mathbf{A}$)
$\tilde{\mathbf{B}}$ = $\mathbf{B}$ - centroide($\mathbf{B}$)
$\mathbf{H} = \tilde{\mathbf{A}}.\tilde{\mathbf{B}}^T$
$[\mathbf{U}, \mathbf{S}, \mathbf{V}^*\mathbf{T}] = \text{SVD}(\mathbf{H})$
$\mathbf{R} = \mathbf{V}^{*T}.\mathbf{U}$
$\mathbf{T} = -\mathbf{R}.\tilde{\mathbf{A}}$

Alignement par ICP L'algorithme ICP, pour Iterative Closest Point, introduit par Chen et Medioni (1992), est un algorithme construit pour minimiser la distance entre deux nuages de points. Il est particulièrement utile lors de la reconstruction de surface 2D ou 3D à partir de différentes acquisitions. Son fonctionnement simplifié est décrit dans l'algorithme 2.2.

L'implémentation de cet algorithme est reprise directement du travail de Kjer et Wilm (2010).

Comparaison des méthodes d'alignement Vicon-LiDAR Parmi ces trois méthodes d'alignement de nuages de points, les méthodes ICP et décomposition en éléments simples donnent des résultats équivalents, comme le mentionne le tableau 2.3. La méthode des moindres carrés, après orthonormalisation de la matrice de rotation donne une distance d'alignement moyenne très légèrement supérieure aux deux autres méthodes.

La méthode retenue est finalement celle de la décomposition en éléments simples, car pour des résultats équivalents à l'ICP, sa mise en œuvre est moins coûteuse en temps de calcul.

Calibrage caméra - Vicon

Pour le reste de ce document, on note $[\mathbf{R}|\mathbf{T}]^{\mathscr{A}}_{\mathscr{B}}$ l'expression de la pose d'un système $\mathscr{A}$ dans un référentiel $\mathscr{B}$. Cela correspond à la transformation permettant le changement de référentiel

Algorithme 2.2 Iterative Closest Point (ICP)

variables :
A = points Vicon
B = points LiDAR
d = distance entre les A et B
ϵ = critère d'arrêt
Algorithme :
while d>ϵ **do**
    for chaque point de A **do** :
        sélectionner le point le plus proche géométriquement dans B
        [R|T]← Estimer la transformation entre ces deux points
        Appliquer cette transformation **[R|T]** au nuage B
        $d \leftarrow$ calculer une distance entre les deux nuages de points
    end for
end while

Méthode d'alignement (14 paires de points)	Ecart moyen	Temps Processeur
Moindres carrés avec orthonormalisation de R	12.75 mm	3.122ms
Décomposition en éléments simples	11.86 mm	2.525 ms
ICP	11.86mm	4.411 ms

TABLE 2.3 – Erreur d'alignement moyenne (mm) et temps d'exécution en fonction de la méthode. Alignement effectué avec 23 paires de points 3D mises en correspondance manuellement.

$\mathscr{A}$ vers le référentiel $\mathscr{B}$, exprimé $\mathbf{Tr}_{\mathscr{B}\leftarrow\mathscr{A}}$.

On cherche ici la transformation rigide entre le centre optique de la caméra et le support de caméra (raccourci en "support" pour la suite de cette partie).

On ne peut pas calculer directement la transformation caméra-support : la pose du support est connue très précisément dans le référentiel Vicon grâce au motif de marqueurs qu'il comporte, mais la pose de la caméra est en premier lieu inconnue.

Il est nécessaire de passer par un référentiel intermédiaire pour estimer la pose de la caméra.

On utilise dans ce but la mire de calibrage des paramètres intrinsèques de la caméra, munie de marqueurs Vicon et d'un motif imprimé d'échiquier.

On note l'origine de l'échiquier $mire_c$ (cf. figure 2.11), et l'origine du motif de marqueurs Vicon fixés à la mire $mire_v$.

La caméra étant calibrée intrinsèquement, et en connaissant la taille exacte de l'échiquier de la mire, il est possible d'estimer la pose de la caméra par rapport à l'origine de l'échiquier. On note cette pose $[\mathbf{R}|\mathbf{T}]^{camera}_{mire_c}$.

On note la transformation de mire_v vers mire_c : $\mathbf{Tr}_{mire_c\leftarrow mire_v}$. La transformation inverse est notée $\mathbf{Tr}_{mire_v\leftarrow mire_c}$. Cette transformation est estimée depuis l'image de la mire physique acquise par la caméra.

La pose de la caméra dans le référentiel Vicon $[\mathbf{R}|\mathbf{T}]^{camera}_{vicon}$est donc donnée par la relation

$$[\mathbf{R}|\mathbf{T}]^{camera}_{vicon} = \mathbf{Tr}_{vicon\leftarrow mire_v}.\mathbf{Tr}_{mire_v\leftarrow mire_c}.[\mathbf{R}|\mathbf{T}]^{camera}_{mire_c}. \tag{2.35}$$

Pour chaque image, on obtient alors une pose caméra dans le référentiel Vicon. Pour un couple de poses caméra et support synchronisées, on sait que :

$$[\mathbf{R}|\mathbf{T}]^{vicon}_{camera} = \mathbf{Tr}_{camera\leftarrow support}.[\mathbf{R}|\mathbf{T}]^{vicon}_{support} \tag{2.36}$$

$[\mathbf{R}|\mathbf{T}]^{camera}_{vicon}$ et $[\mathbf{R}|\mathbf{T}]^{support}_{vicon}$ étant des transformations connues, on obtient $[\mathbf{R}|\mathbf{T}]^{vicon}_{camera}$ et $[\mathbf{R}|\mathbf{T}]^{vicon}_{support}$ grâce à une inversion matricielle. On peut alors simplement calculer la transformation $\mathbf{Tr}_{camera\leftarrow support}$ avec :

$$\mathbf{Tr}_{camera\leftarrow support} = [\mathbf{R}|\mathbf{T}]^{vicon}_{camera}.[\mathbf{R}|\mathbf{T}]^{support}_{vicon}. \tag{2.37}$$

Cependant, cette méthode n'optimise pas l'estimation de $\mathbf{Tr}_{camera\leftarrow support}$ pour toutes les poses. En effet, les poses $[\mathbf{R}|\mathbf{T}]^{camera}_{vicon}$ dépendent de l'estimation de $[\mathbf{R}|\mathbf{T}]^{camera}_{mire_c}$ qui repose sur la détection du motif d'échiquier et qui est elle-même susceptible au bruit. L'estimation à partir de seulement une pose caméra et support est susceptible d'être loin de la réalité.

Afin d'améliorer la précision de l'estimation de $\mathbf{Tr}_{camera\leftarrow support}$, on doit utiliser une méthode exploitant toutes les poses caméra et support.

Deux options sont alors possibles :

— On peut calculer une transformation $\mathbf{Tr}_{camera\leftarrow support}$ pour chaque pose caméra et support, et calculer une pose médiane ou moyenne.

— Il est aussi possible de formuler le problème sous forme d'optimisation au sens des moindres carrés.

Si l'on considère cette seconde option, on reformule (2.36) de manière à pouvoir utiliser (2.30) :

$$\Big([\mathbf{R}|\mathbf{T}]^{vicon}_{camera}\Big)^T = \Big([\mathbf{R}|\mathbf{T}]^{vicon}_{support}\Big)^T.\Big(\mathbf{Tr}_{camera\leftarrow support}\Big)^T. \tag{2.38}$$

On concatène les $\mathbf{n}$ poses de manière à ce que

$$\begin{aligned} &\mathbf{A} = \Big(([\mathbf{R}|\mathbf{T}]^{vicon}_{support})^T_{\mathbf{1}} \quad ... \quad ([\mathbf{R}|\mathbf{T}]^{vicon}_{support})^T_{\mathbf{n}}\Big) \\ &\text{et} \\ &\mathbf{B} = \Big(([\mathbf{R}|\mathbf{T}]^{vicon}_{camera})^T_{\mathbf{1}} \quad ... \quad ([\mathbf{R}|\mathbf{T}]^{vicon}_{camera})^T_{\mathbf{n}}\Big). \end{aligned} \tag{2.39}$$

Ainsi, de la même manière que dans (2.30), on a finalement

$$\mathbf{Tr}_{camera\leftarrow support} = ((\mathbf{A}^T.\mathbf{A})^{-1}.\mathbf{A}^T.\mathbf{B})^T. \tag{2.40}$$

Expérimentation et résultats A l'issue de la phase de calibrage, on procède à l'acquisition des données qui seront directement utilisées pour l'estimation de pose. Le système caméra est déplacé dans un rayon de 5m autour de l'origine du référentiel Vicon, et orienté suivant des angles importants. Nous sommes donc en possession d'un ensemble d'images, dont la pose est connue dans un nuage de points LiDAR. Sur l'ensemble du jeu de données, on sélectionne un nombre réduit d'images, au nombre de six, avec des configurations angulaires suffisamment différentes pour valider le bon fonctionnement de nos algorithmes avec des directions verticales variées. Il reste cependant une dernière étape avant de procéder à l'estimation de pose : l'extraction de droites.

En partant du nuage de points LiDAR, on extrait des droites manuellement en sélectionnant des points deux par deux, chaque paire définissant une droite distincte, en veillant à ce que les droites résultantes soient facilement identifiables dans les images du jeu de données. Les figures 2.14 et 2.15 illustrent le procédé. On paramétrise ensuite les paires de points en droites en coordonnées de Plücker, telles que décrites dans la section 2.2.

Les droites 3D qui sont extraites du nuage de points sont exprimées de prime-abord dans le référentiel du LiDAR, et il convient de les exprimer dans le référentiel Vicon avant de pouvoir appliquer l'algorithme de PnL à l'aide des paramètres estimés dans la phase de calibrage du système.

De la même manière que pour les droites 3D, des paires de points sont sélectionnées manuellement sur les images du jeu de données, comme l'illustre la figure 2.16, puis paramétrisées en

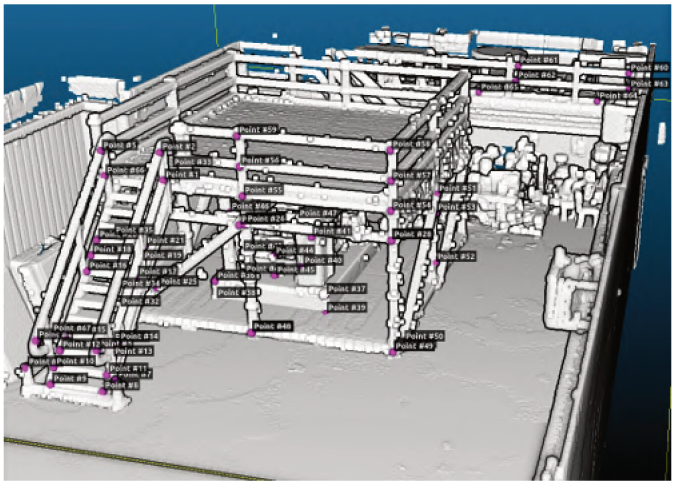

FIGURE 2.14 – Nuage de points dense extrait par un Leica ScanStation C10 et des points extraits manuellement.

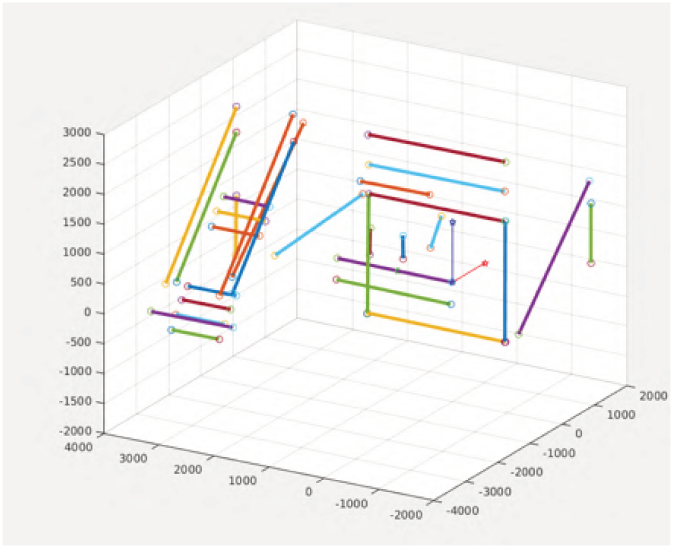

FIGURE 2.15 – Visualisation des droites 3D extraites du nuage de points dense.

FIGURE 2.16 – Image de l'environnement de test, acquise par une caméra *Logitech Quickcam 4000*, avec des droites 2D extraites manuellement (vert).

droites. Cette paramétrisation est effectuée en exprimant les coordonnées des points extraits de l'image en coordonnées métriques à l'aide des paramètres intrinsèques de la caméra.

D'une manière similaire à l'équation 1.4, on réalise la conversion de coordonnées pixeliques vers métriques en effectuant le produit de l'inverse de la matrice des paramètres intrinsèques et de chaque point exprimé en coordonnées pixeliques. Ainsi, pour un point $\mathbf{p_{pixeliques}}=\begin{pmatrix} u & v & 1 \end{pmatrix}^T$, et les paramètres intrinsèques $\mathbf{K}$ d'une caméra tels que définis dans la section 1.3, le point $\mathbf{p_{métrique}} = \begin{pmatrix} x & y & 1 \end{pmatrix}^T$ correspondant exprimé en coordonnées métriques est alors :

$$\mathbf{p_{métrique}} = \mathbf{K^{-1}}.\mathbf{p_{pixelique}} = \begin{pmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} . \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}.$$

Une droite 2D telle qu'elle est obtenue à l'issue d'une projection de droite 3D en coordonnées de Plücker sur un plan, est représentée par le vecteur normal à son plan de projection. Soient deux points 2D $\mathbf{p_s} = \begin{pmatrix} x_s & y_s & 1 \end{pmatrix}^T$ et $\mathbf{p_e} = \begin{pmatrix} x_e & y_e & 1 \end{pmatrix}^T$ exprimés en coordonnées métriques homogènes. On représente la droite $\boldsymbol{l}$ que forment ces deux points par :

$$\mathbf{l} = \frac{\mathbf{p_s} \times \mathbf{p_e}}{||\mathbf{p_s} \times \mathbf{p_e}||} \tag{2.41}$$

avec $\times$ qui représente le produit vectoriel. On note que cette représentation sous forme d'un vecteur de dimension 3 suffit à définir la droite : le plan formé par la normale au plan de projection a une intersection unique avec le plan image, quelle que soit la valeur de la focale de la caméra (cf. figure 2.1). L'appariement des droites 2D et 3D est effectué manuellement, lui aussi.

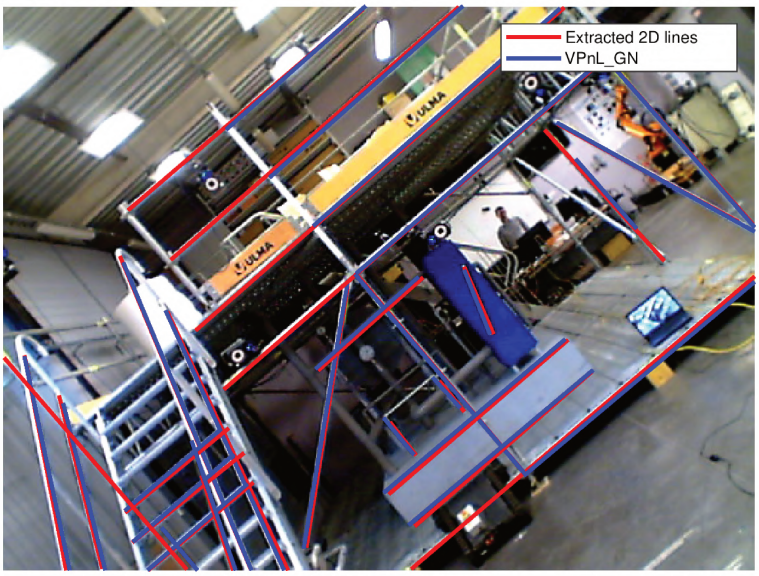

FIGURE 2.17 – Projection des droites 3D extraites du nuage de points sur une image à l'aide des paramètres extrinsèques estimées par la méthode VPnL_GN.

Finalement, on exécute nos deux algorithmes d'estimation pour les six poses sélectionnées, et on compare ces résultats à notre vérité terrain.

On évalue de la même manière les performances des algorithmes ASPnL, NPnLUpL et NPnLUpC, introduits dans la section précédente. Les figures 2.17 et 2.18 présentent en image le résultat de l'estimation de pose à l'aide de notre méthode VPnL_GN. Le tableau 2.4 détaille les résultats numériques de l'exécution de ces algorithmes.

TABLE 2.4 – Valeurs moyennes, médianes et écart type des erreurs d'estimation de position (%) et d'orientation (°) pour les six poses sélectionnées. Les droites 2D et 3D sont extraites et appairées manuellement. La caméra est déplacée dans un rayon de 5m autour de l'origine du référentiel Vicon.

	Erreur d'orientation (°)			Erreur de position (%)		
Méthode	**Med.**	**Moy.**	**Écart type**	**Med.**	**Moy.**	**Écart type**
VPnL_LS	0.247	0.207	0.087	2.728	3.064	1.459
VPnL_GN	0.276	0.294	0.082	2.786	3.109	1.481
NPnLUpL	0.249	0.267	0.187	3.241	3.894	2.760
NPnLUpC	0.278	0.298	0.108	5.281	27.27	50.47
ASPnL	5.494	13.64	21.72	6.767	6.889	2.702

Discussion Cette seconde expérimentation, sur des données réelles cette fois, confirme les performances de nos deux algorithmes d'estimation de pose avec connaissance de la direction verticale, devant celles des algorithmes à l'état de l'art. Néanmoins, le faible nombre de poses

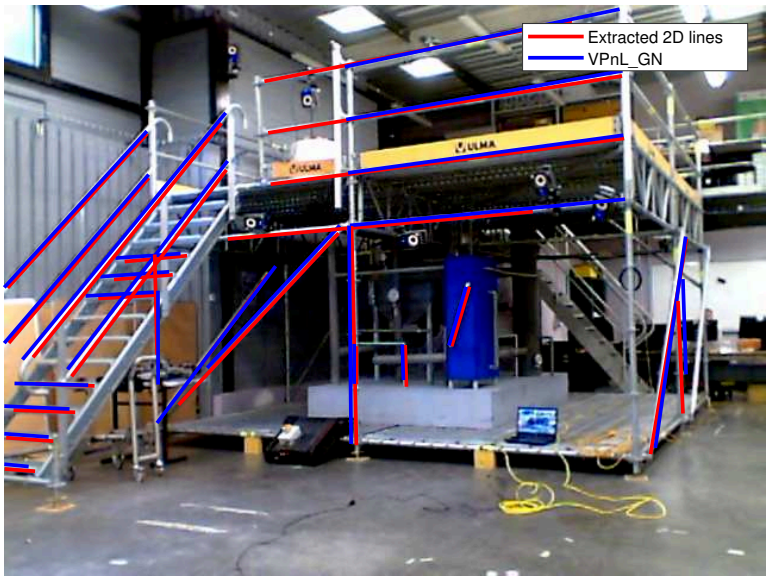

FIGURE 2.18 – Une vue différente utilisée pour la validation de nos algorithmes.

employées pour cette expérimentation ne permet qu'une validation conceptuelle. Une expérimentation sur un plus grand nombre de poses, et sur un jeu de données de référence paraît donc judicieux. Dans cette optique, une expérimentation sur un troisième jeu de données est présentée dans la section suivante.

2.4.4 Jeu de données KITTI

Introduction

Dans cette section, on évalue nos algorithmes d'estimation de pose sur une partie du jeu de données KITTI (Geiger *et al.* (2013)). Ce jeu de données très complet regroupe un grand nombre de séquences dédiées à des applications variées de robotique. Il est acquis par un véhicule se déplaçant dans différentes configurations routières (plusieurs séquences de zones urbaines, périurbaine et grands axes), et est constitué de paires d'images à haute résolution, en couleur et nuances de gris, acquises par un double système de caméras stéréoscopiques. L'acquisition de ces images est synchronisée avec un LiDAR *Velodyne HDL-64-E* qui permet d'associer à chaque image un nuage de points 3D épars de la scène autour du véhicule. La vérité terrain de la pose est fournie par une centrale inertielle *OXTS RT3003* couplée à un système de positionnement satellite. La figure 2.19 présente le synoptique du jeu de données, et les différents référentiels considérés.

Parmi les séquences du jeu de données, nous avons sélectionné un sous ensemble de la séquence 00 du jeu de données d'odométrie et SLAM. Cette sous-séquence est composée de 53 images (aux indices 1223 à 1276) à travers lesquelles le véhicule exécute deux tournants serrés successifs pour

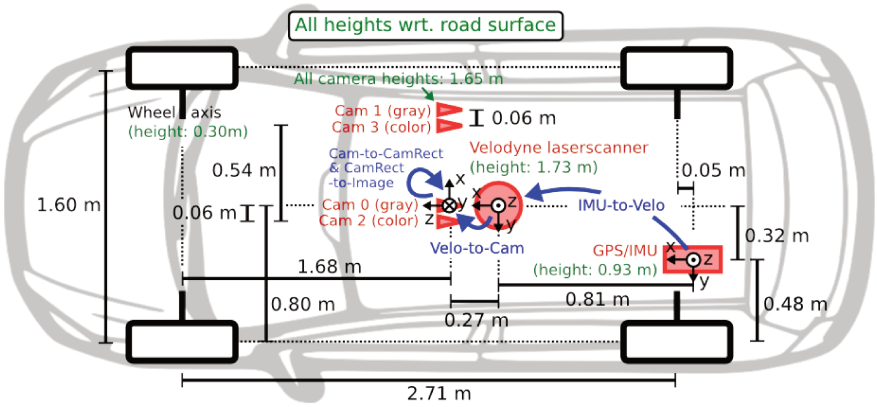

FIGURE 2.19 – Synoptique du jeu de données KITTI, par Geiger *et al.* (2012).

contourner un groupe d'immeubles.

Utilisation du jeu de données Ce jeu fournissant une pose pour le véhicule dans un référentiel global, et un ensemble d'images et nuage de points associés, il est théoriquement possible de recomposer un nuage de points denses, à l'image de celui du jeu de données IRSEEM, simplement à partir de la vérité terrain fournie. La réalité est nettement moins séduisante. Le jeu de données KITTI, bien qu'encore très utilisé aujourd'hui, est réputé pour être prompt aux erreurs dans sa vérité terrain dans les sections urbaines denses comprenant de nombreux virages. Ainsi, l'utilisation seule de la vérité terrain pour recomposer un nuage de points dans la section choisie ne suffit pas. En concaténant simplement les points des différents nuages successifs en prenant en compte les changements de poses donnés par la vérité terrain, de légères erreurs d'orientation et de position s'accumulent, finissant par donner un nuage inexploitable dans le cas de notre application.

Néanmoins, des outils existent pour pallier à ce genre d'erreurs. Particulièrement, les algorithmes de recalage de nuages de points basés ICP sont performants en la matière. La bibliothèque 3DTK, connue sous le nom de SLAM6D (3dt (2019)), opère sur ce principe. Elle est disponible librement et a permis de réaliser ce recalage en prenant simplement les poses de la vérité terrain comme initialisation, fournissant un nuage de points recalé, et une vérité terrain corrigée pour toutes les poses de notre sous-séquence. On peut observer le résultat de ce recalage sur la figure 2.20.

La comparaison entre la trajectoire du jeu de données et la trajectoire corrigée par 3DTK est visible sur la figure 2.22, ainsi que les poses successives estimées par VPnL_GN. On note que ces dernières sont superposées avec les poses 3DTK.

Avec à disposition un nuage de points recalé, un ensemble d'images et les poses associées, on peut procéder à la validation des algorithmes d'estimation de pose. Pour chacune des 53 images, on extrait entre 5 et 8 droites manuellement. Ces droites sont ensuite appariées à leurs

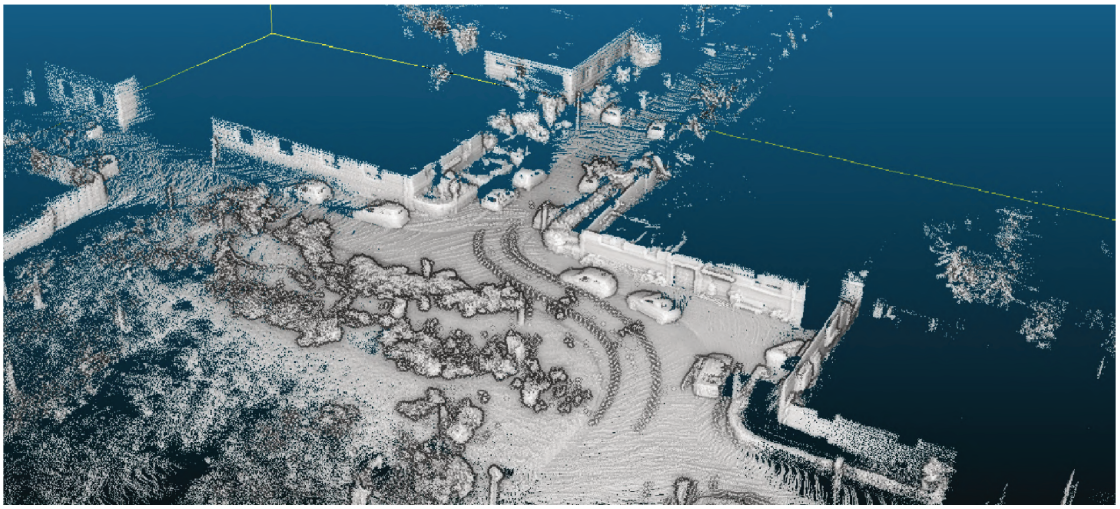

FIGURE 2.20 – Nuage de points issu de la concaténation de nuages successifs à l'aide de la librairie 3DTK.

correspondantes extraites, toujours manuellement, dans le nuage de points recalé. On procède finalement à l'évaluation des méthodes VPnL_LS, VPnL_GN, ASPnL, NPnLUpL et NPnLUpC.

Résultats

La figure 2.21 illustre la projection des droites 3D sur une des images extraites de la séquence. Cette projection est effectuée à l'aide de la vérité terrain 3DTK, et de la pose estimée par VPnL_GN. Sont affichées en superposition les lignes 2D ayant servi à estimer la pose. Le tableau 2.5 présente les résultats numériques de l'expérimentation sur la sous-séquence extraite du jeu de données KITTI. On notera qu'ici, l'erreur en translation est présentée en mètres. On mesure ici la distance euclidienne entre le centre optique réel de la caméra, et la position estimée de ce même point.

Les deux algorithmes NPnLUpL et NPnLUpC sont ici inefficients. ASPnL fournit des résultats avec une très bonne valeur médiane, mais une valeur moyenne et un écart type élevés, qui signifient que lorsqu'une mauvaise estimation de pose survient, l'erreur est très importante. Bien que ce phénomène soit contrôlable à l'aide d'un filtre appliqué entre plusieurs estimations successives, on cherchera dans une application de robotique mobile à obtenir un faible écart type en termes d'erreur. Nos deux algorithmes fournissent ici des performances au dessus du lot, autant en termes d'erreur d'estimation de rotation que de translation. On remarque que VPnL_GN fournit des performances supérieures, notamment en terme d'estimation de la position. Tout comme les performances supérieures d'ASPnL vis-à-vis de NPnLUpL et NPnLUpC, ces résultats correspondent au cas de simulation pour lesquels un bruit particulièrement modéré est présent.

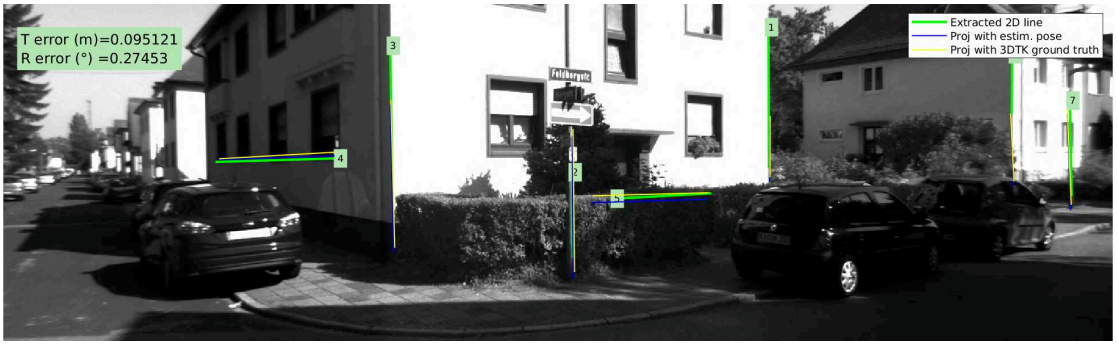

FIGURE 2.21 – Image extraite du jeu de données KITTI . Projection de lignes 3D sur l'image à l'aide de notre méthode de PnL.

Méthode	Erreur Orientation (°)			Erreur Translation (m)		
	Moyenne	Médiane	Écart type	Moyenne	Médiane	Écart type
VPnL_LS	0.62	0.38	0.65	3.19	0.44	10.15
VPnL_GN	0.50	0.34	0.43	0.14	0.13	0.09
ASPnL	20.96	0.87	52.16	3.50	0.19	9.23
NPnLUpL	127.55	148.41	51.21	406.57	30.71	1767
NPnLUpC	94	81.12	45	445.79	163	909.94

TABLE 2.5 – Résultats numériques des algorithmes d'estimation de pose sur le jeu de données KITTI

2.4.5 Discussion

Dans ce chapitre, nous avons tout d'abord introduit un état de l'art de l'estimation de pose basée ligne, puis présenté un algorithme original s'appuyant sur la représentation en coordonnées de Plücker et la connaissance de la direction verticale. Nommé VPnL_LS, cet algorithme a été complété d'une méthode d'optimisation de la pose, appelée VPnL_GN. Cette méthode repose sur l'algorithme de Gauss-Newton pour raffiner successivement les estimations de l'orientation et de la position.

Afin de valider ces méthodes, nous avons proposé une étude de la robustesse de ces méthodes. Nous avons expérimenté tout d'abord en simulation, puis sur un jeu de données réduit réalisé au sein du laboratoire IRSEEM, et finalement sur le célèbre jeu de données KITTI. Les résultats présentés s'accordent à dire que les deux algorithmes introduits fournissent des performances supérieures à celles des algorithmes de l'état de l'art, autant en présence de bruit 2D ou 3D important, que quand des erreurs sont présentes dans la direction verticale fournie. Néanmoins, la sensibilité à la présence de données aberrantes, particulièrement le mauvais appairage de droites 3D et 2D suggère qu'une plus grande robustesse passe par l'ajout d'un algorithme préalable d'appairage. À travers le chapitre 3, nous nous consacrons à cette problématique.

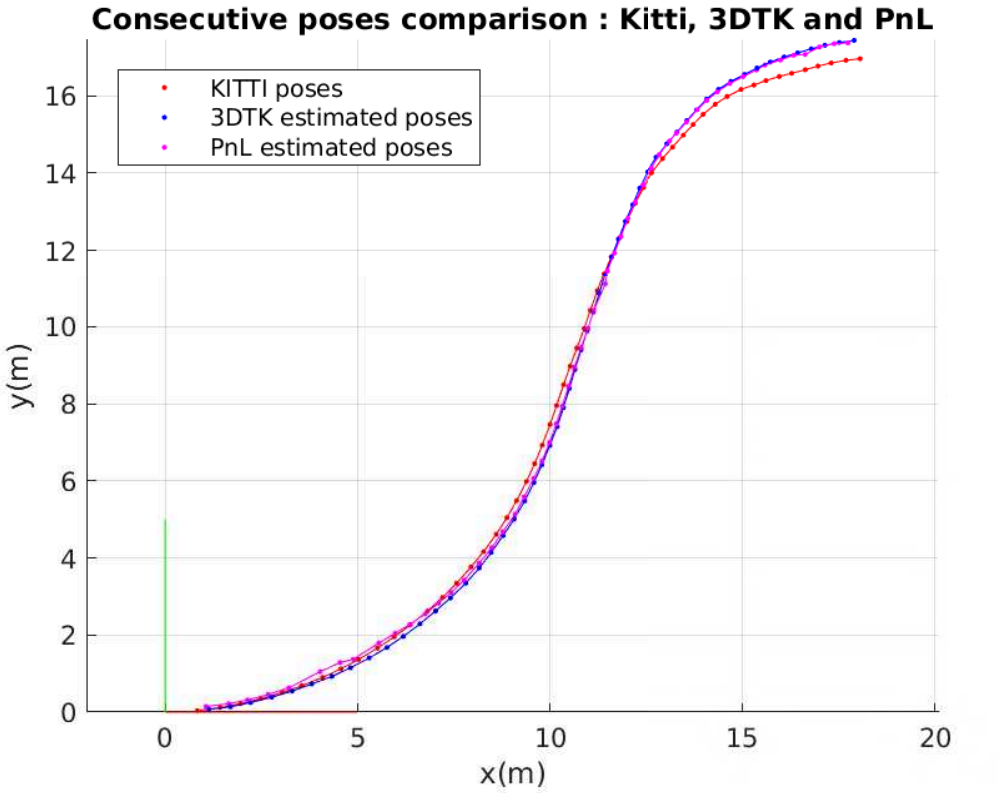

FIGURE 2.22 – Comparaison entre vérité terrain KITTI, 3DTK et pose estimée par VPnL_GN

Chapitre 3

Contribution à la mise en correspondance de droites 2D/3D

3.1 Introduction

Dans le premier chapitre, nous avons vu que le processus de recalage de capteurs implique tout d'abord de retrouver des informations communes entre les capteurs et modalités à recaler. Ces informations peuvent faire intervenir l'intégralité des données issues de chaque modalité, et faire ainsi appel aux méthodes directes de recalage, ou s'appuyer sur des points saillants, appelés amers pour procéder au recalage. Enfin, on peut aussi associer à chaque amer une signature liée à son voisinage, appelée descripteur, et faire appel à des méthodes homonymes.

Nous avons aussi vu que quelle que soit la méthode employée, il est nécessaire d'effectuer une mise en correspondance entre les données des capteurs à recaler. Si le détail de cette étape varie en fonction des méthodes employées, on réalise néanmoins cet appairage en minimisant une distance entre les données considérées, qu'elle soit une norme L1, L2, de Mahalanobis, de Hamming...

À travers le second chapitre, nous avons introduit une méthode originale de recalage de capteurs caméra et LiDAR, introduite sous le terme estimation de pose 2D/3D, et s'appuyant sur une connaissance *a priori* de la direction verticale. Si l'évaluation de la robustesse de cette méthode la place à l'état de l'art lorsque l'appariement des données entre modalités est connu, on observe une sévère dégradation des performances lorsque des paires aberrantes sont introduites. Ce phénomène est aussi observé pour les algorithmes employés dans ce test comparatif. Pour cette raison, nous proposons dans ce chapitre deux contributions à l'appairage de droites 2D/3D, sous forme de deux algorithmes s'appuyant sur respectivement deux et une paire de droites.

3.2 Métriques pour l'appariement d'amers

Afin d'évaluer les performances des algorithmes d'appairage, nous proposons de nous référer à deux métriques bien connues, qui sont le taux de rappel et le taux de précision.

Pour un jeu de données à n droites 2D et m droites 3D, le nombre de combinaisons possibles est donné par

$$\text{nombre de paires possibles} = m.n. \tag{3.1}$$

Un algorithme d'appariement de droites 2D/3D idéal doit être capable d'identifier et de retourner les paires correctes dans un jeu de données, tout en écartant les paires incorrectes, qui sont des correspondances aberrantes. On cherche donc à maximiser les taux de vrais positifs et de vrai négatifs.

Néanmoins, la réalité n'est que rarement idéale, et un certain nombre de paires peuvent être à tort identifiées comme correctement appairées. Il s'agit de faux positifs. De la même manière, des paires correctes peuvent être écartées. Il s'agit alors de faux négatifs. La figure 3.1 propose une représentation visuelle de ces termes.

Le taux de rappel renseigne sur le nombre de paires correctement appairées par rapport au nombre de paires correctes possibles. Il est défini comme

$$\text{Taux de rappel} = \frac{\text{Nombre de vrais positifs retournés}}{\text{Nombre total de paires correctes}}. \tag{3.2}$$

Ainsi, un fort taux de rappel signifie qu'on a identifié un grand nombre de paires correctes. Pourtant, en sélectionnant toutes les paires possibles, on sélectionne forcément toutes les paires correctes, mais en incluant aussi un nombre important de faux positifs. Le taux de rappel n'est donc pas une métrique suffisante pour qualifier un algorithme d'appairage.

Le taux de précision permet de compléter cette première métrique. Il évalue le nombre de paires correctes retournées en fonction du nombre total de paires retournées :

$$\text{Taux de précision} = \frac{\text{Nombre de vrai positifs retournés}}{\text{Nombre de paires retournées}}. \tag{3.3}$$

Pour une recherche donnée, ces deux métriques nous permettent donc de quantifier les performances des algorithmes d'appairage.

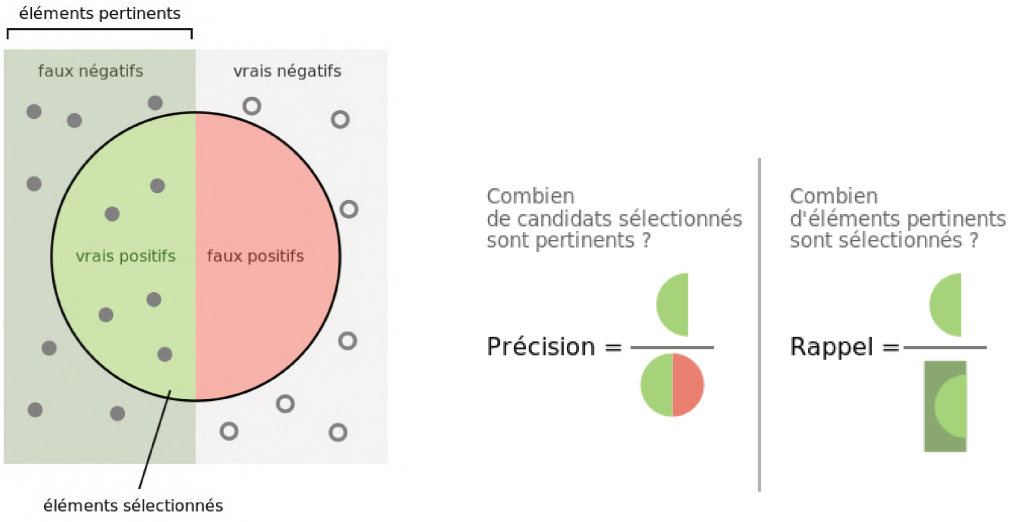

FIGURE 3.1 – Représentation graphique du taux de rappel et de précision, wikipedia.org/wiki/precision_et_rappel

3.3 État de l'art de l'appairage dans le cas de méthodes de PnL

Comme nous avons pu le noter à travers le chapitre 2, dans le cas de méthodes de recalage s'appuyant sur des descripteurs, il est possible de comparer chaque descripteur individuel avec l'ensemble du jeu de descripteurs. On obtient ainsi une distance entre descripteurs, qui permet l'appairage entre descripteurs, et *in fine* un recalage.

Dans le cas de méthodes reposant uniquement sur des amers, cette approche d'appariement individuel n'est pas réalisable. En effet, l'appariement ne peut alors se faire que dans un contexte global : on recherche l'ensemble ou un sous ensemble de paires d'amers qui permet d'estimer les paramètres du recalage minimisant un critère d'erreur, comme le montre la figure 1.24.

3.3.1 Combinatoire de l'appairage

Données aberrantes Supposons que l'on cherche à estimer la pose entre deux capteurs. Pour chacune des modalités, on a réalisé une acquisition de données dont on a extrait respectivement m et n amers, avec $\{m, n\} \in \mathbb{N}$ et $m < n$. Ces amers sont stockés en deux listes nommées $\mathcal{M}$ et $\mathcal{N}$. On suppose ici que chaque amer de $\mathcal{M}$ possède une correspondance valide dans la liste $\mathcal{N}$. $\mathcal{N}$ comportant plus d'éléments, la réciproque n'est donc pas vraie. On a donc m paires correctes existantes, mais inconnues.

À partir de ces deux listes, il est possible de former très exactement $m.n$ paires uniques d'amers. Sur l'ensemble des paires uniques possibles, nous avons donc un taux de paires correctes

tel que

$$\text{Taux de paires valides}=\frac{m}{m.n},$$

ou un taux de paires aberrantes donné par

$$\text{Taux de paires aberrantes} = 1 - \frac{m}{m.n}.$$

On comprend alors que même lorsque $m = n$, et que tous les amers possèdent une correspondance, le taux de paires valides ne cesse de diminuer avec l'augmentation du nombre d'amers. Par exemple, pour un ensemble de 20 amers appairés, 400 combinaisons d'amers en paires sont possibles, donc seulement 5% de paires valides sur l'ensemble des combinaisons possibles. Pour un ensemble de 50 paires valides, on a 2500 combinaisons en paires possibles, donc un taux de 2% de paires valides.

PnL robuste Dans la littérature de l'estimation de pose, un grand nombre de méthodes supposent les correspondances entre amers connues, ou partiellement correctes. Dans ce second cas, ces algorithmes intègrent des modules dédiés au rejet de paires d'amers aberrantes, et sont qualifiés de robustes. Ferraz *et al.* (2014) proposent d'utiliser l'intégralité des amers extraits et de minimiser itérativement une erreur algébrique, en s'inspirant des travaux de Lepetit *et al.* (2009). Cette méthode d'estimation de pose basée points (*Perspective-n-Point*) fonctionne néanmoins en supposant une connaissance *a priori* des correspondances 2D/3D, et ne permet que d'appréhender un taux de paires aberrantes de 50%. En se basant sur des droites, Zhang *et al.* (2012b) utilisent une connaissance approximative de la pose pour procéder à l'appariement d'amers. Leur méthode minimise itérativement une erreur géométrique reposant conjointement sur l'estimation de la rotation et de la translation. Plus récemment, Xu *et al.* (2017) ont proposé des algorithmes de PnL robustes, reformulant partiellement les travaux de Abdel-Aziz *et al.* (2015) (réédition de 1971) et leur formulation de l'algorithme DLT pour la rendre applicable aux droites.

Pourtant, si ces méthodes offrent des performances intéressantes pour des taux de paires aberrantes de l'ordre de 50%, les résultats sont sévèrement dégradés lorsque l'appariement total est à effectuer, ou lorsqu'un grand nombre d'amers sont initialement mal appariés.

Appairage basé sur les sous-ensembles Dans ces circonstances, les méthodes les plus performantes s'avèrent être celles employant seulement un nombre limité d'amers pour effectuer l'appariement. L'idée est alors la suivante. Si l'on sélectionne un nombre réduit de paires d'amers, on maximise les probabilités de n'avoir retenu que des paires valides. A l'aide d'un algorithme d'estimation de pose exploitant ce sous ensemble, on peut estimer la pose complète, ou même seulement une partie de la pose.

Dans le cas où le sous-ensemble sélectionné ne contient donc que des paires valides, la pose estimée sera, sinon exacte, a minima proche de la pose réelle. On peut alors utiliser cette pose pour mesurer une erreur géométrique pour toutes les paires existantes. Toujours dans l'hypothèse

où le sous-ensemble sélectionné est valide, un nombre important de l'ensemble des paires possibles produira une erreur faible, signifiant ainsi que ces paires sont elles-mêmes correctement appairées. On peut alors extraire les paires classifiées comme valides de l'ensemble de toutes les paires possibles, et ensuite procéder à une estimation de pose utilisant les amers nouvellement appariés.

Si le sous-ensemble de paires sélectionnées ne comprend peu ou pas d'amers correctement appariés, la pose estimée permettra toujours de mesurer une erreur pour l'ensemble des paires possibles, mais seul un très faible nombre de paires produira une erreur géométrique faible. Ainsi, on est en mesure d'identifier une combinaison d'amers dont toutes ou une partie des paires sont aberrantes, et de continuer le processus itératif de recherche de combinaisons valides d'amers.

Si la taille maximale du sous-ensemble sélectionné ne dépend que du nombre d'amers à apparier, la taille minimale est elle limitée par la méthode d'appairage. Dans le cas du PnL, un minimum de quatre correspondances d'amers est nécessaire afin d'estimer une pose unique et complète. On parle alors de P4L. Dans le cas où une partie de la pose est supposée connue, comme c'est le cas pour notre algorithme VPnL, un minimum de trois droites est nécessaire.

Deux approches sont alors possibles.

Approche exhaustive On peut tout d'abord explorer exhaustivement toutes les combinaisons existantes de paires d'amers. À l'issue de cette exploration exhaustive, on sélectionnera le sous ensemble ayant généré, lors d'une itération donnée, l'erreur la plus faible pour le plus grand nombre de paires.

Dans le cas de la recherche exhaustive, le nombre minimal de paires à considérer dans l'algorithme permettant d'estimer les paramètres du modèle recherché (ici la pose), est particulièrement important, car il influe considérablement sur la combinatoire du problème. En supposant que notre algorithme de PnL exploite un minimum de k correspondances de droites, et que l'on souhaite donc explorer toutes les possibilités de combinaisons pour k amers 2D et k amers 3D. En sélectionnant k amers dans chaque ensemble $\mathcal{M}$ et $\mathcal{N}$, on a respectivement C_m^k et C_n^k combinaisons (lu respectivement « k parmi m » et « k parmi n ») possibles. La lettre C est ici le symbole de la combinaison, liée à la notion de coefficients binomiaux. De même, il y a respectivement $k!$ arrangements en paire possibles des k amers extraits de chaque ensemble $\mathcal{M}$ et $\mathcal{N}$. Finalement, pour tester exhaustivement les combinaisons de k paires d'amers des ensembles $\mathcal{M}$ et $\mathcal{N}$, on a :

$$\text{nombre de combinaisons de k paires} = k!\mathrm{C}_m^k.C_n^k. \tag{3.4}$$

Pour expliciter numériquement la formule (3.4), on suppose un jeu de données comportant 20 droites 3D et leur projection. Chaque amer possède donc une correspondance. On a alors un taux de paires aberrantes de 95% sur les 200 paires existant entre les deux ensembles. On suppose aussi un algorithme de PnL acceptant une configuration minimale de cinq paires de droites pour fonctionner (*i.e.* P5L). Tester toutes les combinaisons uniques de cinq paires possibles pour ce jeu de données revient à tester 28 844 881 920 combinaisons. Pour le même jeu de données, si l'on réduit le nombre de paires nécessaires à l'algorithme de cinq à quatre, on passe à 563 376 600

Algorithme de PnL	**Nombre de combinaisons de n paires**	**Temps d'exécution (0.01ms par test)**
n=1	200	2 ms
n=2	72 200	722 ms
n=3	7 797 600	13 min
n=4	563 376 600	15 h 38 min
n=5	28 844 881 920	33 jours 9 h 14 min

TABLE 3.1 – Pour un jeu de 20 paires d'amers, nombre de combinaisons possibles de n paires d'amers et temps d'exécution en supposant 0.1ms par passe de PnL

tests nécessaires à l'analyse exhaustive. Passant à trois puis deux, on doit tester respectivement 7 797 600 et 72 200 combinaisons. En posant un temps de calcul plausible de 0.01ms par passe de l'algorithme d'estimation de pose, on peut estimer le temps de calcul dans ces différents cas. Ces résultats sont repris dans le tableau 3.1.

Chiffres à l'appui, on comprend rapidement deux faits. Premièrement, diminuer le nombre de paires d'amers nécessaires au fonctionnement d'un algorithme d'estimation de pose permet de grandement diminuer la combinatoire du problème d'appairage. L'approche exhaustive devient alors viable du point de vue temps de calcul si le nombre de paires utilisées dans l'estimation de pose est suffisamment faible.

Deuxièmement, l'appariement s'appuyant sur des sous-ensembles ne nécessite qu'un nombre réduit d'amers correctement appariés pour apparier un jeu de données complet. Une approche probabiliste du tirage de paires, plutôt qu'une approche exhaustive, permettrait alors de diminuer le nombre de tirages tout en conservant néanmoins une grande probabilité de tirer un nombre réduit de paires valides. Pour cette raison, les approches probabilistes sont aujourd'hui préférées dans ce cas de figure.

Approche probabiliste Dans le cas des approches d'appairage dites probabilistes, on souhaite donc effectuer un nombre de tirages de paires d'amers maximisant la probabilité de tirer une combinaison de paires valides. Cette catégorie d'algorithmes est plus connue sous l'appellation RANdom SAmple Consensus, ou RANSAC (consensus de l'échantillon aléatoire, en anglais), et a été introduite par Fischler et Bolles (1981). On associe l'acronyme RANSAC au nombre d'éléments nécessaires à l'estimation du modèle. Un algorithme RANSAC3 dédié à l'appairage nécessite donc 3 paires d'amers pour fonctionner. Récemment, Xu *et al.* (2017) ont proposé deux algorithmes RANSAC4 et RANSAC3 s'appuyant sur une formulation minimale de leur algorithme ASPnL.

En fonction du taux de paires valides ϵ, et du nombre s de paires d'amers nécessaires à l'estimation de pose, et de la probabilité p d'un tirage valide souhaitée, il est possible de déterminer le nombre W de tirages à effectuer.

En supposant un tirage aléatoire indépendant, ϵ^s exprime la probabilité que les s paires tirées soient valides. Ainsi, $1 - \epsilon^s$ donne la probabilité qu'au moins une des paires soit une paire

Algorithme de PnL	**Nombre de tirages pour une probabilité de réussite de 99%**	
	Taux de paires valides : 2%	**Taux de paires valides : 5%**
n=1	228	90
n=2	11 511	1840
n=3	575 644	36 839
n=4	28 782 311	736 825
n=5	1 439 115 661	14 736 542

TABLE 3.2 – Nombre de tirages nécessaires pour obtenir une probabilité de 99% de tirer un nombre n de paires d'amers valides, pour un jeu de données avec un taux de paires valides de 2% et 5%

aberrante. La probabilité de ne jamais choisir un nombre de paires valides parmi les W tirages effectués est donc de $(1-\epsilon^s)^W$. Si l'on donne p la probabilité de tirer une paire valide parmi tous les tirages, alors la probabilité de ne jamais tirer de paire valide est de $1-p$. On sait donc que $1-p=(1-\epsilon^s)^W$. En appliquant un logarithme sur cette égalité, on obtient donc la formule

$$W=\frac{\log(1-p)}{\log(1-\epsilon^s)}. \tag{3.5}$$

En fixant la valeur p à 99%, on peut donc estimer le nombre de tirages à effectuer en fonction du nombre de paires d'amers nécessaire à l'estimation des paramètres de la pose. Le tableau 3.2 présente les résultats de l'application numérique de l'équation 3.5.

En comparaison avec le concept d'appairage exhaustif, il apparaît évident que les algorithmes basés sur une approche probabiliste permettent de diminuer grandement le temps d'exécution de l'appairage. Néanmoins, l'aspect probabiliste induit une incertitude des résultats, certes maîtrisable, mais pour autant présente.

Que l'on considère une approche probabiliste ou exhaustive, les deux applications numériques des tableaux 3.2 et 3.1 soulignent l'importance de l'apport de formulations minimales de l'estimation de pose. Diminuer le nombre de paires d'amers nécessaires à ce processus permet de diminuer la combinatoire du problème, et donc d'en accélérer la résolution. Cet aspect est d'autant plus important que dans des applications réelles, les correspondances entre les données acquises par des modalités différentes sont rarement connues. De manière générale, l'appariement d'amers et le rejet de données aberrantes comptent parmi les tâches les plus coûteuses en temps de calcul dans le processus d'estimation de pose. La réduction de la charge processeur permet, entre autres, de diminuer l'énergie consommée par le processus complet d'appairage, mais aussi la puissance de calcul à embarquer dans les robots. Ainsi, on ouvre la porte au traitement en ligne des données, c'est-à-dire au traitement embarqué au fur et à mesure que les données capteurs sont acquises. On introduit alors la notion de traitement en temps réel, pour lequel le temps de calcul par tâche est connu. On parle alors de déterminisme du temps traitement de l'information.

3.4 Appariement par RANSAC2 et RANSAC1

Lorsque l'on considère les algorithmes d'estimation de pose basés ligne, la formulation minimale pour estimer une pose unique nécessite quatre paires d'amers, i.e. P4L. Comme cité dans le chapitre 2, Chen (1990) théorise que le P3L, dans sa formulation générale, fournit huit solutions pour la pose. Nous avons aussi vu dans le chapitre 2 que la connaissance *a priori* de la direction verticale permet de réduire la complexité du problème d'estimation de pose. On obtient ainsi une solution unique pour une formulation à trois paires d'amers (P3L).

Nous avons aussi vu dans le chapitre 2 que la première contribution de ce travail de thèse, l'algorithme d'estimation de pose VPnL, propose une formulation minimale à trois droites pour estimer une pose à 4 degrés de liberté en s'appuyant sur une connaissance *a priori* de la direction verticale. De plus, cet algorithme permet aussi d'estimer l'orientation du système pour un minimum de deux droites.

3.4.1 Aspect algorithmique

Algorithme RANSAC2 pour l'appairage de droites C'est en s'appuyant sur cette seconde possibilité qu'est proposée ici la seconde contribution majeure de ce mémoire. L'équation (2.9), qui est rappelée ici

$$(\mathbf{l}_{\boldsymbol{i}}^{\mathscr{C}})^T.\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}.\mathbf{V}_{\boldsymbol{i}}^{\mathscr{M}} = 0,$$

stipule que le vecteur directeur $\mathbf{V}$ d'une droite 3D est perpendiculaire au plan dans lequel est inscrit sa projection $\mathbf{l}$ (cf. figure 2.1). Cette contrainte géométrique, lorsqu'elle est employée pour deux paires de droites, permet d'identifier une solution unique pour l'orientation du système. L'utilisation d'un plus grand nombre de paires d'amers ne permet alors que d'augmenter la précision de cette estimation.

En s'appuyant sur cette équation, on est donc capable d'effectuer l'estimation d'une partie des paramètres extrinsèques du système à partir de deux paires de droites. On peut ensuite employer cette contrainte, en combinaison avec cette estimation préliminaire de l'orientation du système, pour mesurer une erreur géométrique sur l'ensemble de toutes les paires de droites possible. Ainsi, on pose les bases d'un algorithme RANSAC2 que l'on peut alors employer autant au rejet de paires aberrantes qu'à l'appariement d'amers. Ce processus est détaillé dans l'algorithme 3.1.

Comme dans tout algorithme basé RANSAC, on commence par extraire aléatoirement des éléments, ici des paires d'amers, permettant d'estimer un modèle. Une fois l'orientation estimée à l'aide de deux paires de droites, on meure l'erreur géométrique sur l'ensemble des paires possibles à l'aide de la contrainte (2.9). On trie ensuite les paires en fonction de l'erreur associée en constituant ainsi un vecteur d'erreur. Ce vecteur est ensuite séparé en $\frac{m.n}{x+1}$ quantiles, m et n représentant le nombre de droites extraites de chaque modalité, et la valeur de x étant déterminée en fonction du taux de paires valides présentes dans le jeu de données. L'objectif est ici que le premier quantile ne contienne que des paires valides. Finalement, la valeur moyenne du premier

quantile nous renseigne sur la qualité de l'orientation préalablement estimée. Une valeur moyenne élevée indique une estimation erronée de l'orientation, donc des paires d'amers invalides. On doit alors procéder à un nouveau tirage. Dans le cas contraire, si la valeur moyenne de l'erreur du premier quantile est suffisamment faible, cela signifie que l'estimation de l'orientation induit une faible erreur pour un nombre de paires de droites conséquent. Ainsi, on peut raisonnablement supposer que l'orientation estimée est proche de l'orientation réelle du système, et que les paires d'amers du premier quantile sont valides.

Comme dans tous les algorithmes basés RANSAC, un nombre d'itérations maximal est fixé en fonction de l'égalité (3.5).

Algorithme 3.1 Appairage par l'algorithme RANSAC2

procedure APPAIRAGE
  entrées :
  $\mathbf{l_i}$: droite 2D normalisée, $i = 1..n$
  $\mathbf{V_j}$: direction de la droite 3D normalisée, $j = 1..m$
  critère d'arrêt
  N : nombre maximum d'itérations autorisées
  x : nombre d'amers que sépare chaque quantile
  sorties :
  amers appariés
  variables internes :
  $\mathbf{R_{C\leftarrow M}}$: matrice de rotation du référentiel monde $\rightarrow$ caméra
  $\boldsymbol{\varepsilon}$: vecteur d'erreur tel que ${\mathbf{l_i}}^T.\mathbf{R_{C\leftarrow M}}.\mathbf{V_j}$
  ϵ : premier quantile de $\boldsymbol{\varepsilon}$
  Algorithme :
  do
    Sélection aléatoire de deux paires 2D-3D
    Estimation de $\mathbf{R_{C\leftarrow M}}$ pour deux paires de droites
    Calculer $\boldsymbol{\varepsilon} = {\mathbf{l_i}}^T.\mathbf{R_{C\leftarrow M}}.\mathbf{V_j}$, $\forall\ i = 1..n,\ j = 1..m$
    Échantillonner le vecteur d'erreur $\boldsymbol{\varepsilon}$ en $\dfrac{m.n}{x+1}$ quantiles
    Extraire ϵ
  while $\epsilon >$ critère d'arrêt ET nombre d'itérations $< N$
  Retourner amers appariés pour lesquels $\boldsymbol{\varepsilon}_i <$ critère d'arrêt 23
end procedure

Algorithme RANSAC1 pour l'appairage de droites Comme nous venons de le voir, la contrainte formulée dans l'équation (2.9) permet de résoudre de manière unique l'orientation du système pour deux paires de droites. Pourtant, il est possible de réduire encore le nombre de droites nécessaire à la résolution de l'orientation. On combine, pour ce faire l'équation (2.9) et la contrainte trigonométrique en un système d'équations tel que

$$\begin{cases} (\mathbf{l_i})^T.\mathbf{R}_{\mathscr{C}\leftarrow\mathscr{M}}.\mathbf{V_j} = 0 \\ c_z^2 + s_z^2 = 1 \end{cases}, \quad (3.6)$$

avec c_z et s_z les sinus et cosinus de l'angle Ψ à déterminer. La résolution de ce système nous donne alors deux solutions pour l'angle Ψ. Pour chacune de ces solutions, on applique le critère d'erreur de l'équation (2.9) à l'ensemble des paires possibles. On obtient ainsi deux vecteurs d'erreur qui peuvent être concaténés pour former un unique vecteur dont la taille est alors le double du nombre de paires possibles. La procédure est ensuite la même que pour l'algorithme RANSAC2, et est détaillée dans l'algorithme 3.2. Nous pouvons donc réaliser un appariement de droites par RANSAC1.

On note que d'un point de vue calculatoire, on doit appliquer le critère d'erreur sur un nombre plus important de paires d'amers que pour l'algorithme RANSAC2. Il est en effet nécessaire de considérer les deux solutions de l'orientation dans le calcul de notre vecteur d'erreur $\boldsymbol{\epsilon}$. Néanmoins, cette opération est peu coûteuse, et l'aspect combinatoire présenté dans le tableau 3.2 et l'équation (3.5) justifie largement ce surcoût calculatoire : on diminue grandement le nombre de tirages de paires à réaliser. Si elle n'est pas ici développée, l'approche exhaustive devient parfaitement justifiable dans le cas d'une estimation de l'orientation à une seule paire de droites.

Algorithme 3.2 Appairage par l'algorithme RANSAC1

procedure APPAIRAGE
  entrées :
  $\mathbf{l_i}$: droite 2D normalisée, $i = 1..n$
  $\mathbf{V_j}$: direction de la droite 3D normalisée, $j = 1..m$
  critère d'arrêt
  N : nombre maximum d'itérations autorisées
  x : nombre d'amers que sépare chaque quantile
  sorties :
  amers appariés
  variables internes :
  $\mathbf{R_{C\leftarrow M}}$: matrice de rotation du référentiel monde $\rightarrow$ caméra
  $\boldsymbol{\varepsilon_1}$: vecteur d'erreur tel que $\mathbf{l_i}^T.\mathbf{R_{C\leftarrow M1}}.\mathbf{V_j}$
  $\boldsymbol{\varepsilon_2}$: vecteur d'erreur tel que $\mathbf{l_i}^T.\mathbf{R_{C\leftarrow M2}}.\mathbf{V_j}$
  $\boldsymbol{\varepsilon}$: vecteur d'erreur issu de la concaténation de $\boldsymbol{\varepsilon_1}$ et $\boldsymbol{\varepsilon_2}$
  ϵ : premier quantile de $\boldsymbol{\varepsilon}$
  Algorithme :
  do
    Sélection aléatoire de deux paires 2D-3D
    Estimation de $\mathbf{R_{C\leftarrow M1}}$ et $\mathbf{R_{C\leftarrow M2}}$ pour une paire de droite
    Calculer $\boldsymbol{\varepsilon_1} = \mathbf{l_i}^T.\mathbf{R_{C\leftarrow M1}}.\mathbf{V_j}$ et $\boldsymbol{\varepsilon_2} = \mathbf{l_i}^T.\mathbf{R_{C\leftarrow M2}}.\mathbf{V_j}$, $\forall\ i = 1..n$, $j = 1..m$
    Concaténer $\boldsymbol{\varepsilon_1}$ et $\boldsymbol{\varepsilon_2}$ en $\boldsymbol{\varepsilon}$
    Échantillonner le vecteur d'erreur $\boldsymbol{\varepsilon}$ en $\dfrac{m.n}{x+1}$ quantiles
    Extraire ϵ
  while $\epsilon >$ critère d'arrêt ET nombre d'itérations $< N$
  Retourner amers appariés pour lesquels $\boldsymbol{\varepsilon}_i <$ critère d'arrêt 23
end procedure

3.5 Étude de la robustesse : Jeu de données de simulation

3.5.1 Introduction

Comme dans le cas des algorithmes d'estimation de pose proposés précédemment, on cherche à valider le concept de ces algorithmes RANSAC2 et RANSAC1. Dans un premier temps, on évalue nos deux algorithmes sur un jeu de données de simulation.

Tout comme dans le cas de l'analyse de la robustesse des méthodes VPnL_LS et VPnL_GN, nous cherchons ici à confronter nos résultats à des algorithmes à l'état de l'art. Dans la littérature consacrée à l'estimation de pose basée ligne, l'appariement des amers est considéré comme déjà réalisé. Les méthodes dite robustes se contentent alors généralement d'isoler les paires d'amers valides des paires aberrantes, et d'estimer ensuite une pose. Si ces algorithmes ne sont, par nature, pas conçus pour effectuer un appariement d'amers, la logique veut qu'ils soient compatibles avec cette tâche. L'appariement n'est, *in fine*, qu'un cas extrême du rejet de paires aberrantes où, parmi l'ensemble de toutes les paires considérées, un très faible taux est valide. On considère dans cette section les algorithmes RANSAC3 et RANSAC4, ainsi que RLPnL_Enull. Tous trois sont tirés des travaux de Xu *et al.* (2017), dont on s'appuiera sur l'implémentation directe pour ces tests.

RLPnL_Enull est une adaptation des travaux de Lepetit *et al.* (2009). RANSAC3 s'appuie sur une formulation polynomiale du P3L, pour estimer la pose. Nous comparons donc dans cette section nos deux méthodes d'appariement d'amers RANSAC1 et RANSAC2, ainsi que les trois algorithmes sus-cités.

Dans un premier temps, on étudie les performances de ces algorithmes dans une tâche de rejet de paires aberrantes. L'objectif est ici d'employer les algorithmes RANSAC3 et RLPnL_Enull dans des conditions qui leur sont favorables, et de déterminer de la sorte si ces algorithmes sont compatibles avec la tâche plus ardue qu'est l'appairage. Ainsi, pour un jeu de données pré-appairé, un certain nombre de correspondances aberrantes sont insérées. On évalue la robustesse en bruitant les amers. L'évaluation des performances est donnée en terme de taux de précision et de rappel, comme définis dans la section 3.2 de ce chapitre, ainsi qu'en taux d'erreur en translation, et en erreur d'orientation, comme nous les avons définis dans le chapitre précédent. L'erreur en rotation est définie par l'équation 2.26. Dans cette catégorie de tests, on validera le concept de nos méthodes dans un environnement de simulation non bruité, puis on validera la robustesse en insérant du bruit 2D, 3D et sur la direction verticale, à l'image de l'évaluation réalisée dans le chapitre 2.

Dans un second temps, on évalue les performances de nos algorithmes dans une tâche d'appariement pure : pour un nombre donné de droites 3D et leurs combinaisons, on considère toutes les combinaisons de paires possibles, et on tâche de retrouver les paires valides. Pour un nombre initial de 50 droites 3D et leur 50 correspondances 2D, on diminue progressivement le nombre de correspondances, tout en réalisant l'appairage.

Génération de données Le processus de génération de droites est ici identique à celui décrit sur la figure 2.2, et utilisé pour la validation des algorithmes VPnL_LS et VPnL_GN. On simule à nouveau une caméra virtuelle de 640 par 480 pixels, de distance focale 655 pixels, déplacée aléatoirement à chaque test d'une distance de 2 à 20 mètres autour de l'origine du référentiel monde. La caméra est ensuite orientée aléatoirement entre 0 rad et 2π rad suivant les trois axes de rotation. On génère 50 droites 3D et leur projection à l'aide des paramètres cités précédemment. Le bruit 2D et 3D ajouté est un bruit gaussien introduit dans les coordonnées des points 2D et 3D définissant les droites du jeu de données.

3.5.2 Évaluation de la robustesse pour le rejet de paires aberrantes

Quand on considère le problème de rejet de paires aberrantes avec un pré-appariement, on se trouve alors en présence de cinquante paires de droites, dont seulement une partie sont des correspondances aberrantes. Dans ce contexte, on fait varier le taux de paires invalides de 0 à 80%, soit jusqu'à 40 paires invalides sur un total de 50.

Absence de bruit La figure 3.2 présente les résultats du rejet de paires aberrantes lorsque les amers valides ne sont pas bruités. L'objectif de ce premier test est d'effectuer une première validation de concept de nos deux algorithmes RANSAC1 et RANSAC2. On propose ici des conditions avantageuses, permettant aussi de valider l'implémentation fournie pour RLPnL_Enull et RANSAC3.

On remarque que nos deux algorithmes montrent des performances supérieures à RANSAC3 et RLPnL_Enull dans le rejet de paires aberrantes, avec un taux de précision et de rappel de 100%, pour une rapidité d'exécution respectivement de l'ordre de 1ms et 3ms pour RANSAC2 et RANSAC1. Les erreurs en translation et rotation de nos deux algorithmes sont ici nulles, quel que soit le taux de paires aberrantes.

En ces circonstances, on constate que le taux de précision de RANSAC3 se dégrade brutalement à partir d'un taux de 50% de paires sans correspondances.

Les performances de RLPnL_Enull, remarquables en temps d'exécution, sont en revanche nettement dégradées en terme d'estimation de pose et de rejet de paires aberrantes. On note toutefois qu'ici, une erreur en translation et rotation nulle n'est que peu significative : un rejet efficace de paires aberrantes ne laisse que des paires valides et non bruitées. Employés dans un algorithme d'estimation de pose, ces amers n'induisent alors pas d'erreur qui pourrait se répercuter sur l'estimation de la translation et de la rotation.

Robustesse au bruit 2D et 3D À l'issue de cette première validation, on cherche à vérifier la robustesse de nos méthodes face un du bruit 2D et 3D important. Toujours pour un jeu de 50 paires d'amers, on fixe le taux de paires aberrantes à 50%, et on ajoute un bruit 2D gaussien variant de 0 à 5 pixels. Dans un second cas de test, on conserve un taux de paires aberrantes de 50%, et on ajoute progressivement un bruit 3D gaussien jusqu'à 500mm sur les points définissant

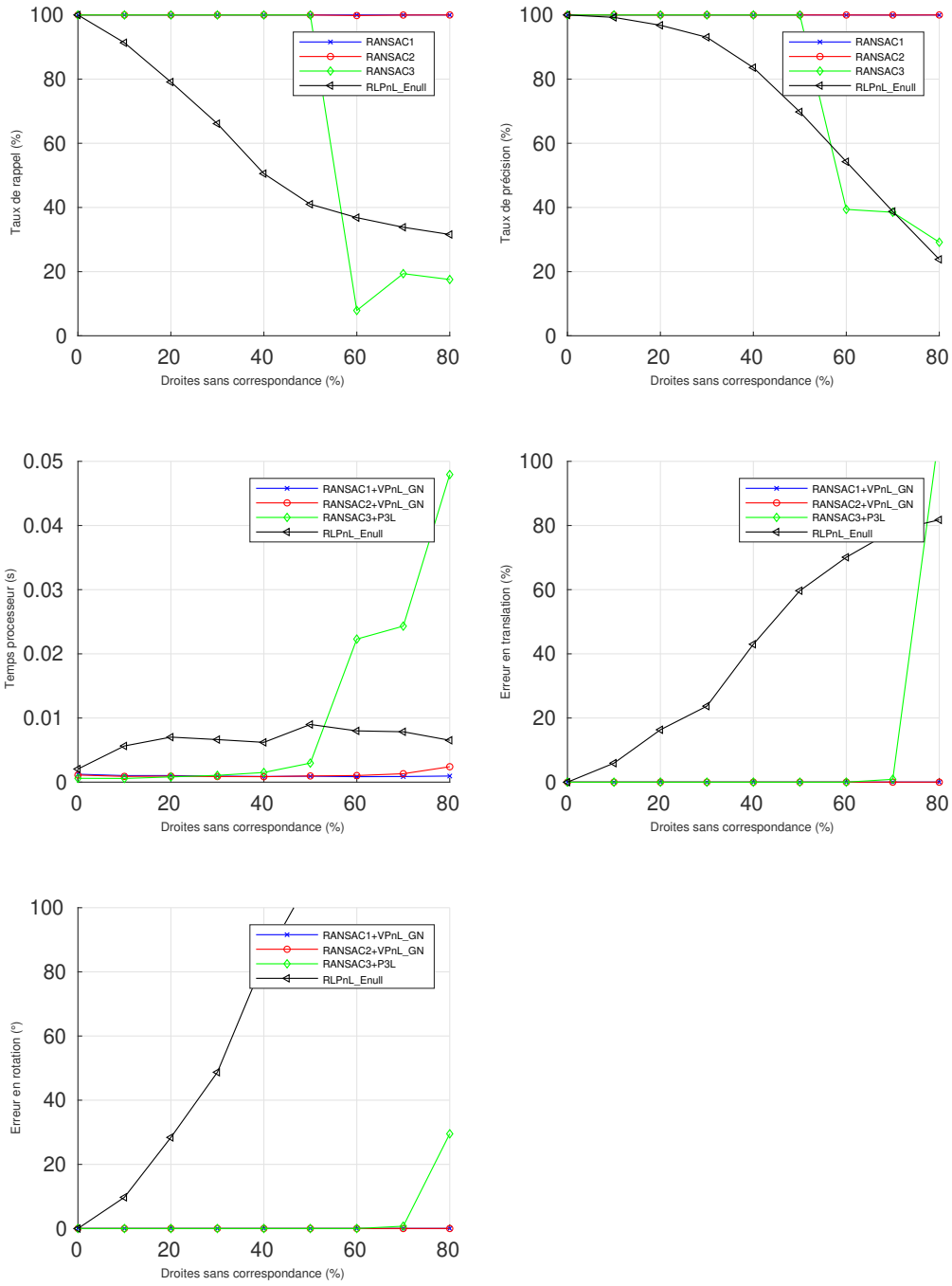

FIGURE 3.2 – Rejet de paires aberrantes : Validation des algorithmes pour un jeu de données non bruité, valeurs moyennes sur 1000 lancers

les droites. Dans un souci d'équité, on rajoute aussi un bruit de 0.1° sur la direction verticale. Les figure 3.3 et 3.4 montrent respectivement les résultats des tests en présence de bruit 2D et 3D.

Que l'on se place dans le cas du bruit 2D ou 3D, nos deux algorithmes fournissent des performances supérieures pour les métriques choisies ici.

Le taux de rappel de RANSAC1 et RANSAC2 oscille entre 40% et 60%, tandis que la précision est comprise entre 98% et 100%. On note dans les deux cas de sévères dégradations des performances de l'algorithme RANSAC3+P3L en terme d'erreur en rotation et translation, avec respectivement plus de 55° et 150% en l'absence de paires aberrantes. Dans le cas de l'estimation de pose en présence d'amers bruités, on observe néanmoins une erreur en rotation et translation pour les algorithmes RANSAC1+VPnL_GN et RANSAC2+VPnL_GN. Pour RANSAC1+VPnL_GN, l'erreur de rotation reste relativement constante, aux alentours de 1° quel que soit le taux de paires aberrantes. L'erreur en translation de nos deux méthodes reste non nulle. On note jusqu'à 6% pour RANSAC1 et 8% pour RANSAC2 à 5 pixels de bruit 2D, malgré un excellent taux de précision dans le rejet de paires aberrantes. En effet, l'introduction de bruit diminue la précision de l'estimation de pose, malgré l'optimisation de la pose incluse dans l'algorithme VPnL_GN.

Soumis à un bruit 2D et 3D important, nos deux algorithmes RANSAC1 et RANSAC2 , combinés à notre méthode d'estimation de pose VPnL_GN, fournissent des performances nettement supérieures à celles des algorithmes de l'état de l'art, pour une tâche de rejet de paires aberrantes.

3.5.3 Évaluation de la robustesse pour l'appariement de droites

Lorsque l'on se place dans le cas d'une tâche d'appairage, l'équation 3.1 nous indique que l'on est en présence de 2500 combinaisons de paires uniques, dont on va chercher à identifier les 50 valides. On calcule donc ici un ratio de 50 paires valides pour 2500 combinaisons possibles, soit un taux initial de paires valides de 2%. De par les résultats dégradés dont fait montre la combinaison RANSAC3+P3L lors de la simple tâche de rejet de paire aberrantes, et compte tenu du temps de calcul par passe estimé à partir des tableaux 3.2 et 3.1, cet algorithme n'est pas ici considéré pour la tâche d'appariement.

Robustesse au bruit 2D et 3D

On évalue maintenant la robustesse de nos algorithmes RANSAC1 et RANSAC2 au bruit 2D et 3D lors de l'appariement de droites 3D. Pour ce faire, on fait varier dans un premier temps un bruit 3D de 0mm à 500 mm, tout en conservant un bruit 2D nul. Comme dans le cas de l'évaluation des performances en cas de rejet d'amers mal appariés, on insère une erreur de 0.1° dans le vecteur gravité. Pour chaque pas de bruit, 1000 tests sont effectués, et les valeurs moyennes des taux de rappel et de taux de précision, tels que définis dans la section 3.2, sont

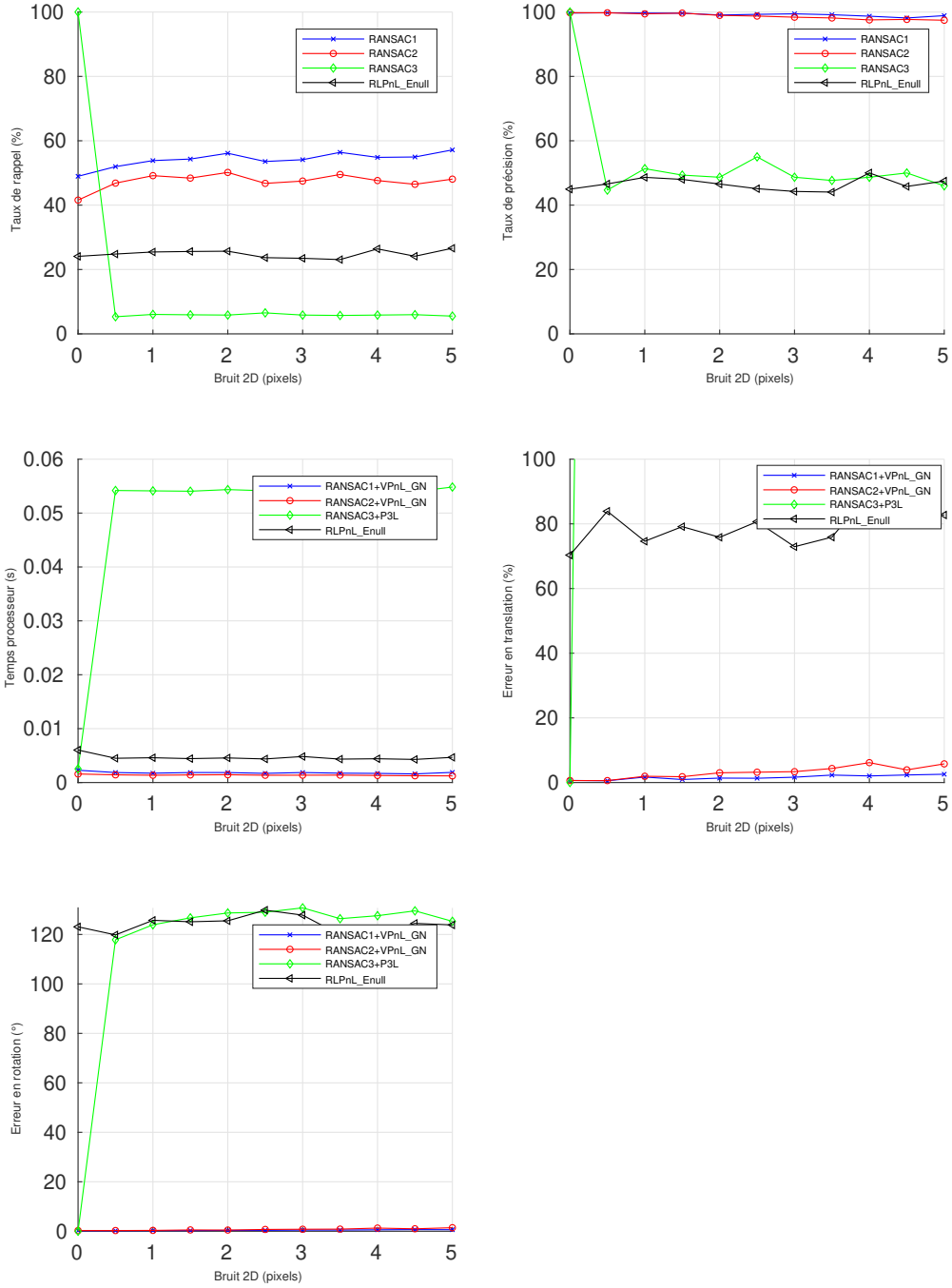

FIGURE 3.3 – Rejet de paires aberrantes : Bruit 2D croissant, erreur de la direction verticale de 0.1°, bruit 3D nul, valeurs moyennes sur 1000 lancers par point, 40% de paires aberrantes.

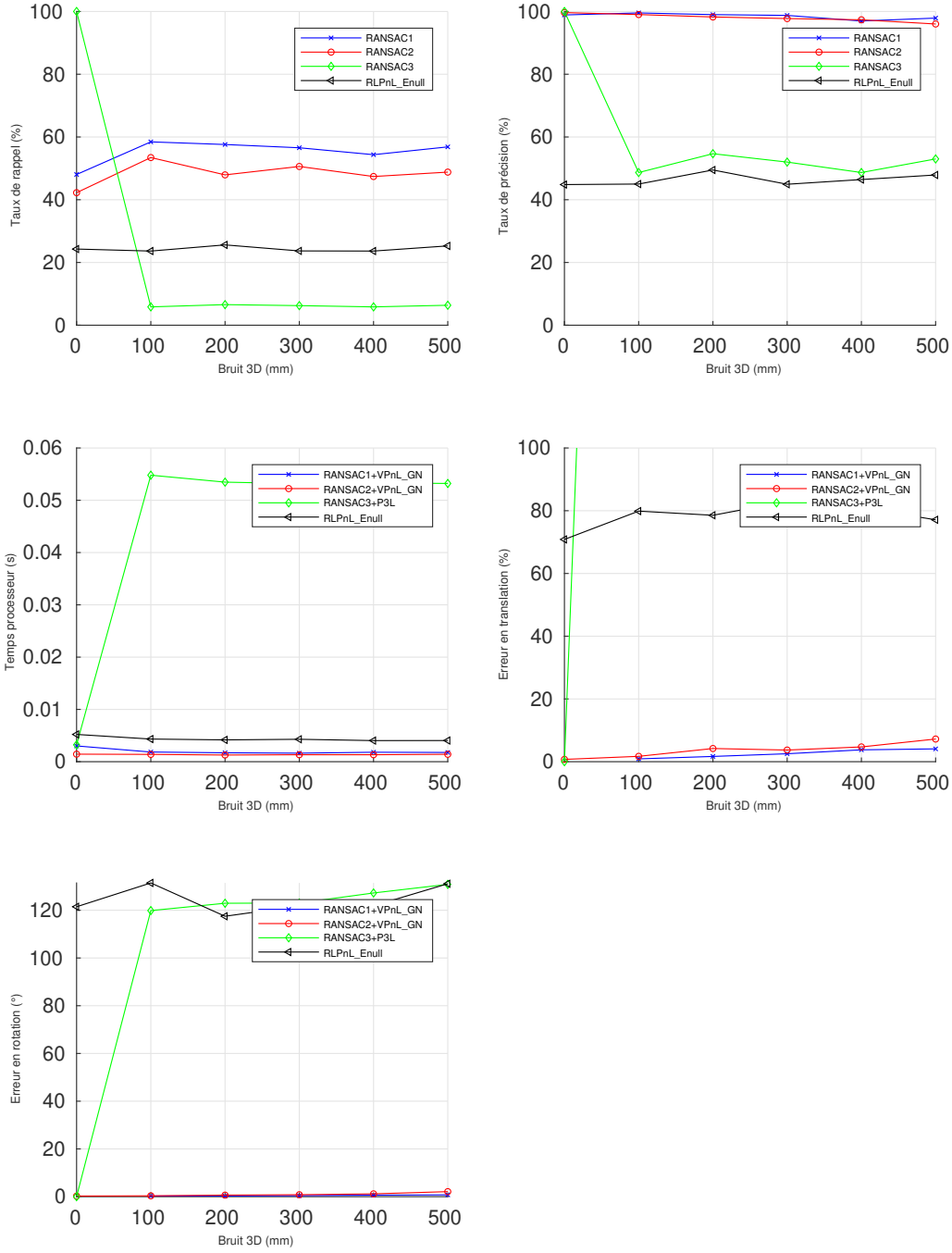

FIGURE 3.4 – Rejet de paires aberrantes : Bruit 3D croissant, erreur de la direction verticale de 0.1°, bruit 2D nul, valeurs moyennes sur 1000 lancers par point, 40% de paires aberrantes.

utilisés permettent d'évaluer les performances des deux algorithmes.On fait ensuite varier le bruit 2D de 0 à 5 pixels, cette fois-ci en l'absence de bruit 3D. La direction verticale sur laquelle s'appuient nos algorithmes reste soumise à une erreur de 0.1°. Les résultats de ce test sont visibles sur les figures 3.5 et 3.6.

L'objectif de l'appariement d'amers est avant tout de former le plus grand nombre de paires d'amers valides pour un jeu de données. En ce sens, on observe tout d'abord les résultats en terme de taux de rappel et de précision. Sur ces deux aspects, RANSAC1 et RANSAC2 fournissent des performances assez similaires.

On note une précision de l'ordre de 99% en l'absence de bruit, qui décroît avec l'introduction d'un bruit de plus en plus important pour atteindre 46% avec 5 pixels et 20% pour 500mm. En comparaison, RLPnL_Enull reste constant avec une précision de 6%, quel que soit le bruit. A l'issue de chaque exécution, nos deux algorithmes proposent entre 20 et 40 paires d'amers, contre une moyenne aux alentours de 800 paires pour RLPnL_Enull. Cette différence explique le très faible taux de précision de RLPnL_Enull, malgré un taux de rappel équivalent ou supérieur à celui de RANSAC1 et RANSAC2.

Si l'appariement est ici évalué, la finalité de ces algorithme est l'estimation de pose. En cela, RANSAC2+VPnL_GN fournit les moins bons résultats pour un fort bruit, avec des erreurs en translation pouvant atteindre les 73% pour 500mm de bruit 3D. En comparaison, dans des circonstances identiques, RANSAC1+VPnL_GN a des performances similaires à RLPnL_Enull, aux alentours de 14% d'erreur. Il en va de même pour les erreurs d'estimation de l'orientation, où RANSAC1+VPnL_GN donne des résultats similaires à RLPnL_Enull, proches de 1.3° en cas de fort bruit, largement en dessous de RANSAC2+VPnL_GN.

Sans surprise, le temps d'exécution de RANSAC2 est nettement supérieur à celui de RANSAC1 et RLPL_Enull. Il est à noter que malgré un taux de précision particulièrement faible dans une tâche d'appariement d'amers, l'algorithme RLPnL_Enull fournit une estimation de pose plausible, en comparaison directe avec RANSAC1+VPnL_GN.

3.5.4 Jeu de données IRSEEM

Si les tests sur le jeu de données de simulation permettent de valider l'aspect théorique des algorithmes RANSAC1 et RANSAC2, il convient néanmoins d'expérimenter aussi sur un jeu de données réelles. On s'appuie ici à nouveau sur le jeu de données IRSEEM, présenté dans le chapitre 2, pour valider nos algorithmes dans ces circonstances. De la même manière que pour les données simulées, on étudie le comportement de nos deux algorithmes tout d'abord dans une tâche de rejet de paires aberrantes, puis d'appariement de droites. On sélectionne pour ce faire une image du jeu de données et la pose associée. De cette image sont extraites 25 droites 2D, dont la correspondance est connue dans le nuage de points représentant la scène d'expérimentation. L'image sélectionnée correspond à un placement de la caméra à 4.3m de l'origine du repère Vicon.

Notre jeu de données n'étant pas parfait, on cherche tout d'abord à quantifier les erreurs liées à l'expérimentation. On mesure donc les erreurs en translation et orientation entre la vérité

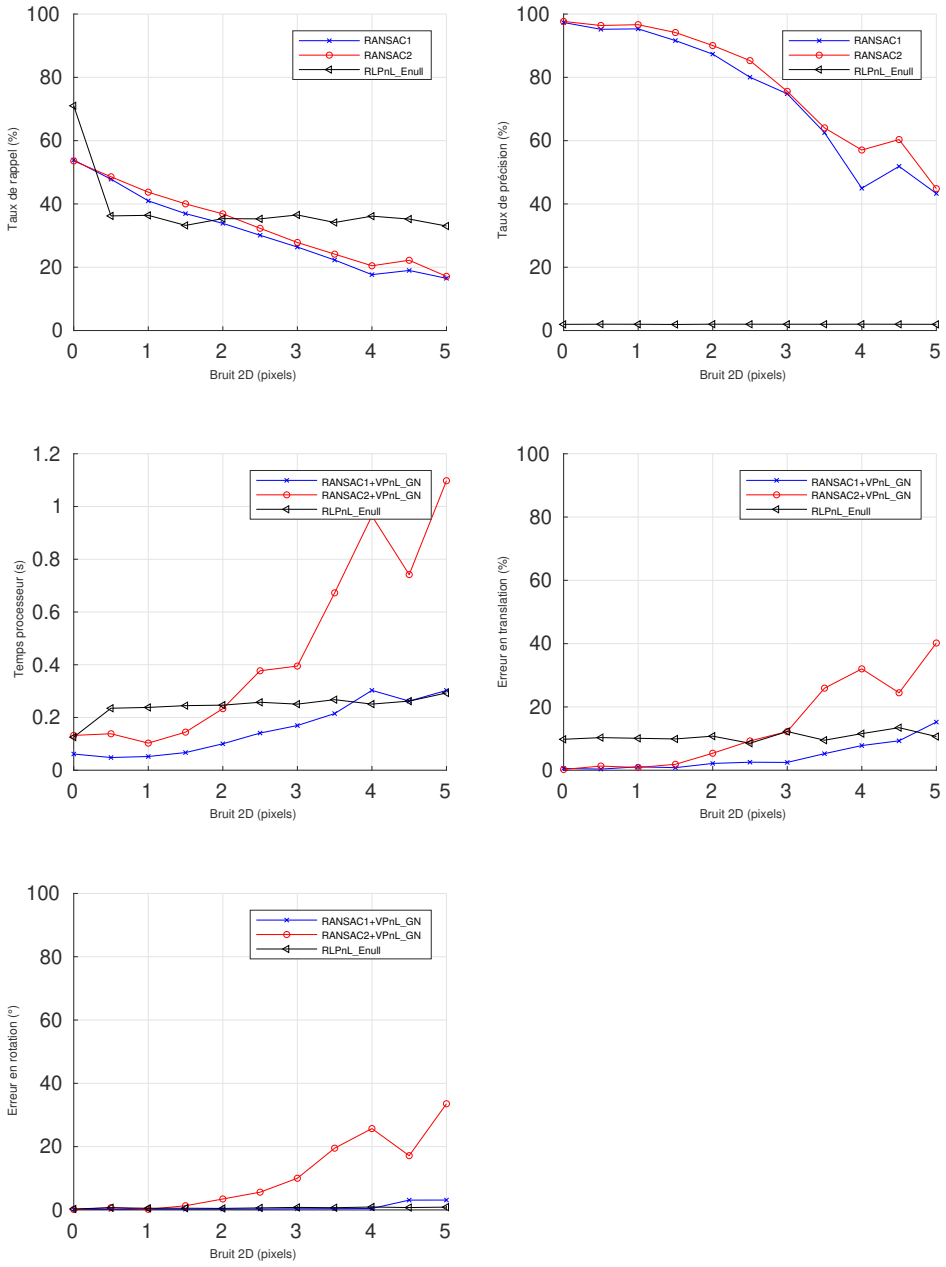

FIGURE 3.5 – Appariement de 50 droites 2D et 50 droites 3D en présence de bruit 2D gaussien croissant, bruit de 0.1° introduit dans la direction verticale. Valeurs moyennes sur 1000 lancers

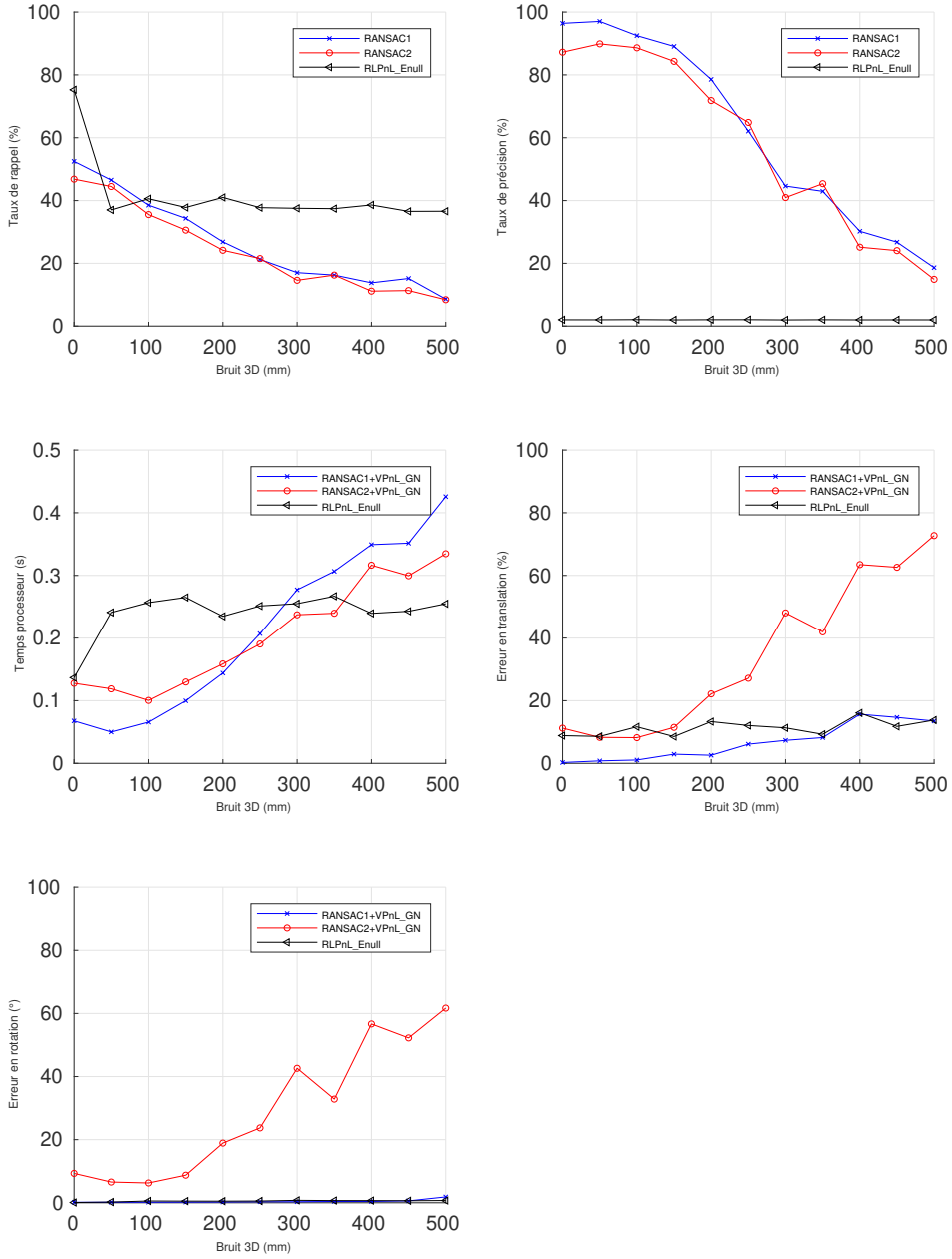

FIGURE 3.6 – Résultats de l'appariement de 50 paires de droites, avec un bruit 3D. Valeurs moyennes sur 1000 lancers, bruit de 0.1° sur la direction verticale

Méthode	Erreur en translation (%)	Erreur en orientation (°)
VPnL_GN	4.95	0.24
ASPnL	3.41	0.95
RLPnL_Enull	2.30	0.77

TABLE 3.3 – Écart entre vérité terrain et estimation de pose pour l'image sélectionnée dans le jeu de données IRSEEM.

terrain et les poses retournées par les algorithmes VPnL_GN, RLPnL_Enull et ASPnL avant tout ajout de paires aberrantes. Ces résultats sont compilés dans le tableau 3.3, et fournissent un élément de comparaison avec les erreurs qui seront mesurées à l'issu de l'insertion de paires aberrantes ou de l'appariement.

Rejet de paires aberrantes Au cours de 10000 itérations, on introduit 50% de paires aberrantes de manière aléatoire en remplaçant 50% des droites 3D par des droites aléatoires. On réalise pour chaque test un rejet de paires aberrantes ainsi qu'une estimation de pose par les méthodes RANSAC2+VPnL_GN, RLPnL_Enull, RANSAC3+P3L.

On y observe dans le tableau 3.4 les valeurs médianes des taux de précision et rappel des algorithmes d'appairage, ainsi que les erreurs en translation et rotation. Dans ce cas particulier, les valeurs moyennes sont écartées car peu représentatives.

En effet, nous avons présenté dans le second chapitre que certaines configurations spatiales ne permettent pas d'estimer une pose complète, particulièrement lorsque l'on est en présence de droites uniquement parallèles. En remplaçant 50% des droites 3D de ce cas de test, cette configuration peut survenir, et induire alors une pose aberrante malgré un taux de rappel et de précision excellents. On se retrouve alors avec des valeurs de translation estimées induisant des erreurs particulièrement élevées, et augmentant fatalement la valeur moyenne. Afin de quantifier ces d'occurrences, on présente dans ce tableau les taux moyens de validité des rotation et translation, qui correspondent au taux de rotation et translation estimées dont les erreurs sont inférieures respectivement à 2° et 5%.

Employés dans des tâches de rejet de paires aberrantes, nos algorithmes RANSAC1 et RANSAC2 démontrent tous deux une précision médiane de 100%, pour un taux de rappel de 61.53%. Les erreurs médianes en rotation et translation de ces deux algorithmes sont particulièrement basses en comparaison avec les algorithmes de l'état de l'art, avec des valeurs de en dessous de 0.40° et 1.60%. L taux de validité en rotation indique qu'une orientation est correctement recouvrée par RANSAC1 et RANSAC2 pour dans respectivement 93.5% et 92.4% des cas. En revanche, le taux de validité de l'estimation de la translation indique qu'une position valide ne peut être estimée que dans 75% des cas, lorsque l'on est en présence d'un fort taux de paires aberrantes.

Ces résultats, placent les performances de nos algorithmes largement au dessus des méthodes concurrentes de l'état de l'art. La configuration géométrique de ce jeu de données a néanmoins un

Méthode	Erreur Rot. (°)	Erreur Trans. (%)	Temps d'exéc. (ms)	Préc. (%)	Rap. (%)	Taux de validité rot.(%)	Taux de validité trans. (%)
RANSAC1 + VPnL_GN	0.3482	1.3585	0.88	100	61.53	93.5	76.61
RANSAC2 + VPnL_GN	0.30487	1.6032	0.88	100	61.53	92.4	73.94
RLPnL_Enull	66.141	145.54	3.22	66.6	53.84	23.09	16.53
RANSAC3 + P3L	142.1	319.21	33.1	87.5	15.38	0.48	0.21

TABLE 3.4 – Valeurs médianes des erreur en rotation et translation, temps d'exécution, taux de précision et taux de rappel. Taux de validité moyen des rotations et translations. Mesures sur 1000 tests et 50% de données aberrantes. La caméra est située à 4.3m de l'origine du référentiel Vicon.

fort impact sur le taux de validité en translation. Le grand nombre de droites parallèles, combiné au taux de paires aberrantes important dégradent les performances dans le cas de l'estimation de la translation.

Appariement de droites Dans ce second test sur le jeu de données IRSEEM, on suppose les correspondances entre droites 2D et 3D inconnues. On doit alors tâcher d'identifier ces correspondances au sein d'un grand nombre de paires possibles. Pour ce cas de test, 8 droites 3D sont remplacées par des droites aberrantes, induisant un taux de droites sans correspondances de 32%. Tout comme dans l'évaluation des performances en présence d'un appariement imparfait, on évalue ici les erreurs médianes en rotation, translation et temps d'exécution. On présente de même les taux médians de précision et rappel, ainsi que les taux de validité moyens en rotation et translation. Ces résultats sont regroupés dans le tableau (3.5).

On note cette fois une nette dégradation des résultats. La configuration géométrique comportant un grand nombre de droites parallèles du jeu de données IRSEEM induit des taux de validité moyens en rotation particulièrement faibles pour les algorithmes RLPnL_Enull (0%) et RANSAC3+P3L (0%). Si les performances de RANSAC1 et RANSAC2 sont ici supérieures, les taux restent néanmoins faibles avec respectivement 30.4% et 23.0%. Les taux moyens de translation sont nuls pour nos deux algorithmes, avec des erreurs au delà de 200%.

Discussion À l'issue de cette première expérimentation sur des données réelles, nous constatons que nos deux algorithmes RANSAC1 et RANSAC2 fournissent des performances dépassant celles des algorithmes de l'état de l'art en terme de rejet de paires aberrantes.

Dans le cas d'une tâche d'appariement de droites, nous constatons cependant une dégradation des performances. La configuration géométrique du jeu de données IRSEEM comporte un grand nombre de droites parallèles, que nous avons identifié plus tôt comme problématiques car ne permettant pas de discriminer efficacement les paires d'amers valides des paires aberrantes. Si ce fait s'est avéré peu significatif dans le rejet simple de droites mal appariées, on note un échec de l'appariement dans la majorité des cas, et ce pour la totalité des algorithmes considérés dans ce test. Seul RANSAC1+VPnL_GN permet de retrouver une orientation avec moins de 5° d'erreur, seulement dans 30% des cas.

Ces résultats démontrent la complexité du processus d'appariement lorsqu'un grand nombre de droites d'orientation similaires sont présentes dans le jeu de données. Une expérimentation sur un jeu de données plus varié est donc nécessaire pour estimer les performances de nos deux algorithmes.

3.5.5 Jeu de données KITTI

Pour valider nos deux algorithmes sur un jeu de données plus varié, on emploie à nouveau le jeu de données KITTI et la séquence utilisée dans le chapitre 2. Pour rappel, cette sous-séquence est constituées de 53 images dont on a extrait dans chacune entre 5 et 7 droites dont

Méthode	Erreur Rot. (°)	Erreur Trans. (%)	Temps d'exéc. (ms)	Préc. (%)	Rap. (%)	Taux de validité rot.(%)	Taux de validité trans. (%)
RANSAC1 + VPnL_GN	5.60	213	11.57	5.88	5.88	30.4	0
RANSAC2 + VPnL_GN	86.98	204	2.02	5.88	5.88	23.0	0
RLPnL_Enull	69.22	141.77	14.13	5.82	47.05	0	0.4
RANSAC3 + P3L	135.49	205	133.33	0	0.00	1.2	8.25

TABLE 3.5 – Valeurs médianes des erreur en rotation et translation, temps d'exécution et taux de précision et rappel. Taux moyen de validité en rotation et translation. Résultats d'appariement pour 1000 tests et 32% de données aberrantes. La caméra est située à 4.3m de l'origine du référentiel Vicon.

Méthode	Erreur Trans. (m)	Erreur Rot. (°)	Temps d'exéc. (ms)	Taux de rappel (%)	Taux de précis.(%)
RLPnL_Enull	19.126	123.79	1.3	46.92	68.07
RANSAC1 + VPnL_GN	0.085	0.058	0.81	94.19	100
RANSAC2 + VPnL_GN	0.089	0.058	0.74	94.29	100
RANSAC3 +ASPnL	1983.3	126.06	20.1	26.34	73.58

TABLE 3.6 – Rejet de paires aberrantes sur une sous-séquence du jeu de données KITTI. Valeurs moyennes pour les erreurs en rotation et translation, le temps d'exécution et les taux de rappel et précision sur 53 poses

la correspondance est connue dans un nuage de points de la scène. Dans le nuage de points, on a extrait un total de 17 droites tout au long de la trajectoire du véhicule.

Rejet de paires aberrantes Dans un premier temps, on se propose de mesurer ici la capacité des algorithmes RANSAC1 et RANSAC2 à effectuer un rejet de paires aberrantes. On insère donc dans les droites 2D et 3D des droites sans correspondances. Pour chaque image, on est alors en présence d'entre 5 et 7 droites 2D possédant une correspondance 3D, et 5 droites 2D sans correspondances. Cela équivaut alors à un taux de paires aberrantes entre 41% et 50% en fonction des images.

Nos deux algorithmes RANSAC1+VPnL_GN et RANSAC2+VPnL_GN sont finalement appliqués à ces données, et les résultats sont comparés à RLPnL_Enull et RANSAC3+P3L en terme de taux de rappel et précision, erreur en translation et rotation, et temps d'exécution. L'erreur en translation est ici donnée en mètres, et correspond à la distance euclidienne entre la position estimée du véhicule à sa position réelle. A nouveau, les poses utilisées comme vérité terrain sont celle fournies par la librairie 3DTK.

La figure 3.7 reprend les résultats de ce test. Les performances présentées ici sont conformes aux résultats des données de simulation : RANSAC3 et RLPnL_Enull fournissent des taux de rappel et précision trop faibles pour permettre une estimation de pose précise. Si RLPnL_Enull rivalise avec RANSAC1 et RANCAC2 en terme de temps d'exécution (1ms en moyenne contre 0.8ms pour RANSAC1 et RANSAC2), les performances en estimation de pose sont très dégradées. Le tableau 3.6 reprend les résultats numériques de cette expérimentation.

Ce premier test confirme les performances de RANSAC1 et RANSAC2 en tant qu'algorithmes de rejet de paires aberrantes d'amers. Combinés à l'algorithme VPnL_GN, ils permettent alors une estimation de pose d'une précision nettement supérieure aux algorithmes à l'état de l'art, tout en garantissant un temps d'exécution inférieur. On notera néanmoins que la connaissance *a priori* de la direction verticale fournit ici un net avantage quant à l'estimation de pose. En effet, les algorithmes RLPnL_Enull et ASPnL estiment une pose à six degrés de liberté, contre seulement quatre dans notre cas.

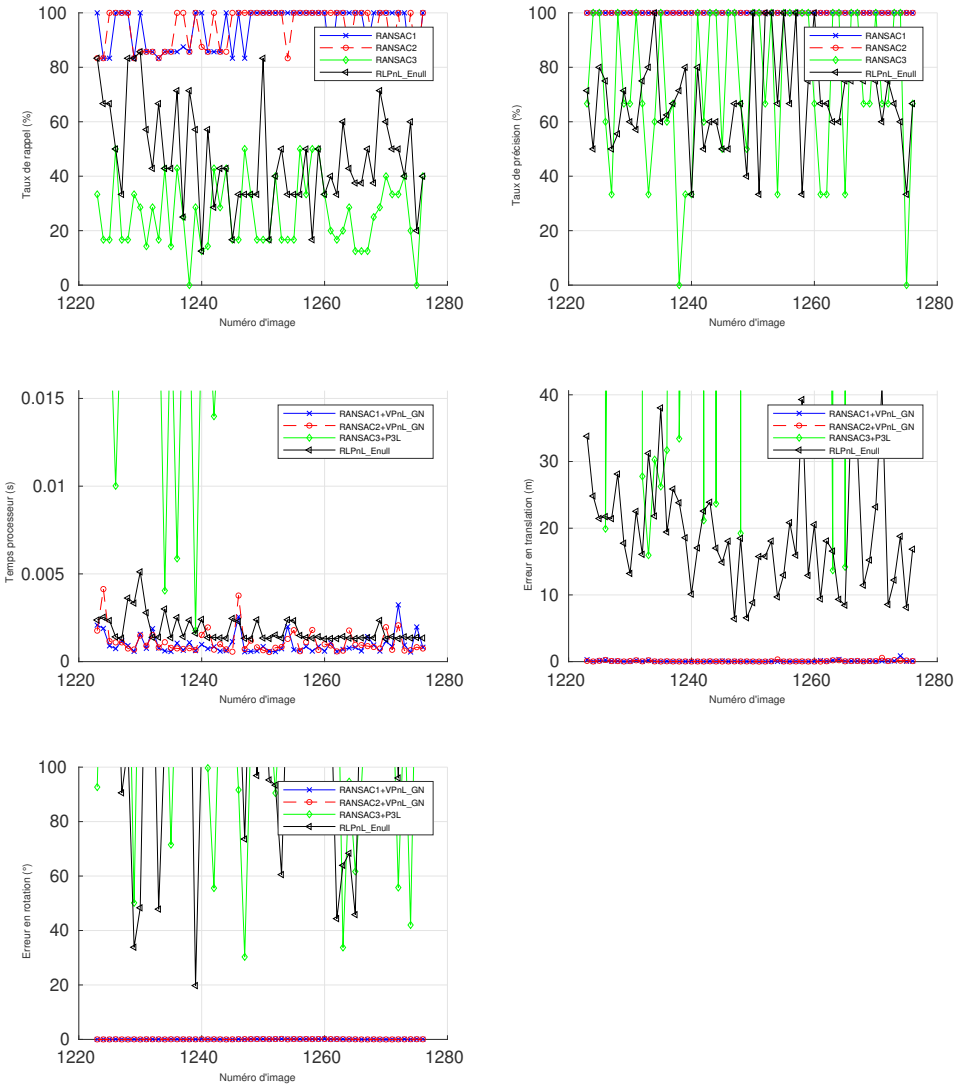

FIGURE 3.7 – Résultat du rejet de paires aberrantes sur le jeu de données KITTI. Taux de rappel et précision, erreur en rotation et translation, et temps d'exécution

Méthode	Erreur Trans. (m)	Erreur Rot. (°)	Temps d'exéc. (ms)	Taux de rappel (%)	Taux de précis.(%)
RANSAC1 + VPnL_GN	4.28	0.048	3.9	87.35	94.36
RANSAC2 + VPnL_GN	5.20	0.055	13.0	81.66	89.70

TABLE 3.7 – Résultats numériques de l'appariement sur KITTI. Séquence de 53 images

Appariement Enfin, on cherche à évaluer les performances de RANSAC1 et RANSAC2 lorsque l'on souhaite réaliser l'appariement de droites. Pour le jeu de données KITTI, le faible nombre de droites disponibles par image ne pose pas un problème d'une complexité combinatoire suffisante. En réalisant l'appariement des droites des images avec leurs correspondances 3D, on se trouve en présence d'entre 25 et 49 combinaisons possibles par pose. Dans l'optique de complexifier le problème d'appariement, on ne considère plus seulement les correspondances 3D de chaque droite, mais la totalité des droites 3D extraites sur la trajectoire du véhicule pour notre sous-séquence. Ainsi, chaque droite 2D doit être appairée parmi un jeu de 17 droites 3D. D'un nombre d'amers sans correspondances nul, on se place ici dans le cas où seulement 30 à 40% des amers peuvent former une paire valide. On augmente aussi de cette manière la combinatoire du problème. De 25 à 49 combinaisons possibles d'amers par pose, on a entre 85 et 119 possibilités à exploiter. On analyse finalement les résultats de l'appariement à l'aide des mêmes métriques que précédemment, présentés sur la figure 3.8 et dans le tableau 3.7. Par souci de lisibilité, et compte tenu des résultats pour la simple tâche de rejet de paires aberrantes, on ne comparera pas ici nos résultats avec les algorithmes RANSAC3 et RLPnL_Enull.

Sauf exception ponctuelle, les taux de rappel et de précision de nos deux algorithmes restent relativement élevés, avec des valeurs respectives moyennes de 87% et 94% pour RANSAC1. Concernant RANSAC2, le taux de rappel moyen est de 81% et la précision de 89%. S'il augmente en comparaison avec les tests de rejet de paires aberrantes, le temps d'exécution de l'appariement reste néanmoins bas avec des valeurs respectives de 3.9ms et 13.0ms pour RANSAC1 et RANSAC2.

Les erreurs en rotation et translation sont particulièrement intéressantes. Pour cette sous-séquence, on observe une forte erreur en translation, respectivement de 4.28m et 5.20m pour RANSAC1 et RANSAC2. En parallèle, l'erreur d'estimation de l'orientation reste très basse, de l'ordre de 0.05 degrés dans les deux cas. Cet écart de performance entre rotation et précision est caractéristique des choix algorithmiques pris pour ces algorithmes.

RANSAC1 et RANSAC2 s'appuient seulement sur l'orientation pour effectuer l'appariement. Ainsi, le facteur optimisé par l'appariement est l'erreur d'orientation. Dans cette sous-séquence, on a extrait un grand nombre de droites coplanaires et perpendiculaires, que l'estimation de l'orientation seule ne permet pas de discriminer systématiquement dans le cas d'une tâche d'appairage. La contrainte géométrique de l'équation 2.26 employée ici optimise donc l'orientation, au détriment de l'erreur en translation.

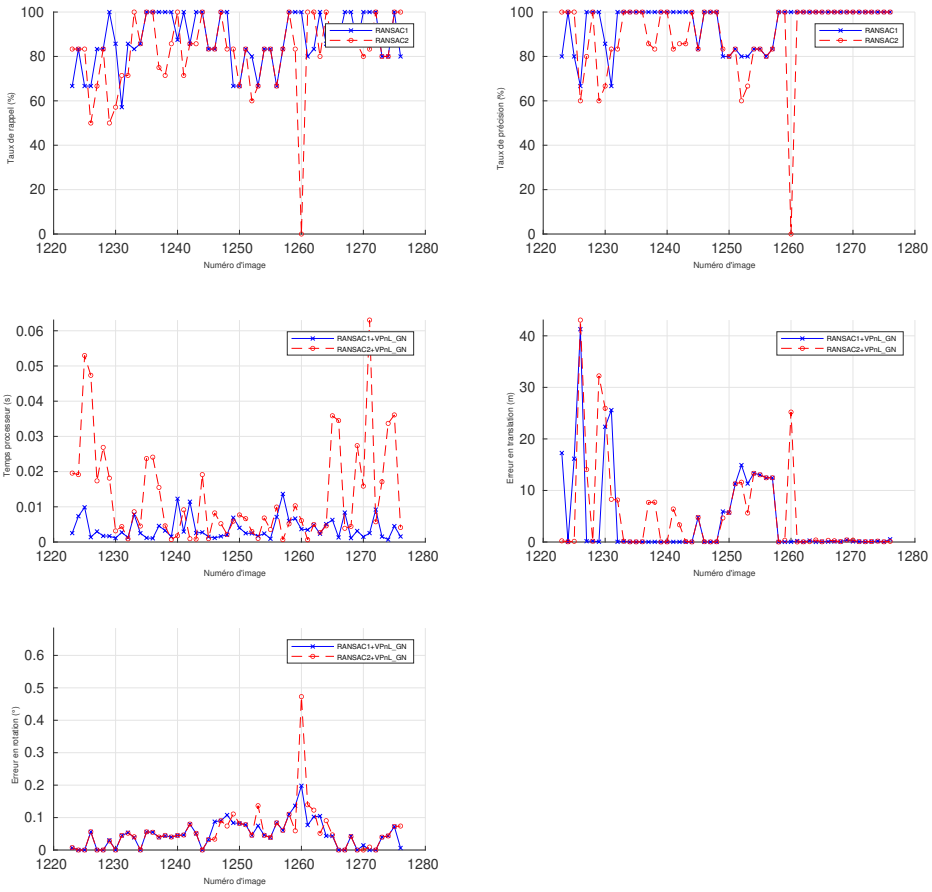

FIGURE 3.8 – Évaluation des performances pour une tâche d'appariement sur une sous-séquence du jeu de données KITTI. Taux de rappel et précision, erreur d'estimation de l'orientation et de la position, et temps d'exécution des méthodes RANSAC1 et RANSAC2

Malgré tout, les taux de rappel et de précision élevés indiquent qu'une large majorité des paires d'amers constituées sont valides. RANSAC1 et RANSAC2 permettent donc d'effectuer un premier appairage, qui pourrait ensuite être affiné à l'aide d'une autre contrainte optimisant cette fois la translation. On se retrouverait alors dans le cas d'un pré-appairage avec un nombre faible de paires aberrantes à écarter avant une estimation de pose subséquente. Nos deux algorithmes, et particulièrement RANSAC1, permettent de diminuer grandement la combinatoire du problème. Leur rapidité d'exécution, démontrée précédemment, permet donc d'effectuer un appariement sinon parfait, a minima pertinent, et d'employer ensuite un algorithme plus coûteux pour le rejet final de paires aberrantes.

3.5.6 Conclusion

Dans ce chapitre, nous avons introduit la problématique ouverte qu'est l'appariement de droites 2D et 3D. Nous avons proposé deux algorithmes basés RANSAC utilisant la connaissance *a priori* de la direction verticale pour effectuer une rejet de paires aberrantes, ou un appariement à l'aide de seulement deux ou une paires d'amers. A travers des tests en simulation et sur des données réelles, nous avons évalué le comportement de ces algorithmes, nommés RANSAC1 et RANSAC2.

Employés en combinaison avec un algorithme d'estimation de pose, tel VPnL_GN, nous avons démontré que ces méthodes permettent d'éliminer efficacement des mauvaises correspondances de droites dans un jeu de données pré-apparié y compris en présence d'un bruit modéré. L'évaluation de la robustesse développée dans ce chapitre place les deux contributions RANSAC1 et RANSAC2 devant les méthodes à l'état de l'art en terme de taux précision et de rappel de paires valides, mais aussi de temps d'exécution pour les tâches de rejet de paires aberrantes.
Lors d'une tâche d'appariement de lignes, nous avons identifié la nécessité d'une certaine variété dans l'orientation des droites 3D, afin de permettre un appairage efficace et précis, et ce quels que soient les algorithmes considérés dans nos tests. Ce point s'est avéré particulièrement important dans les tests effectués sur le jeu de données IRSEEM. Nous avons en effet observé une nette dégradation des performances de tous les algorithmes comparés lorsqu'un grand nombre de droites parallèles est considéré pour l'appariement.

Avec une configuration géométrique des droites plus variée, l'expérimentation sur le jeu de données KITTI nous permet de tirer des conclusions différentes. Si le taux de droites correctement appariées est ici faible, les importants taux de rappel suggèrent l'emploi de RANSAC1 et RANSAC2 comme algorithmes de pré-appariement. Une combinaison avec un algorithme de rejet de paires aberrantes minimisant conjointement les erreurs d'orientation et de translation permettrait alors un appariement performant.

Conclusion et perspectives

À travers cette thèse, nous avons abordé la problématique d'estimation de pose robuste multimodale dans le cas d'une complémentarité caméra et LiDAR.

Dans un premier chapitre, nous avons introduit les concepts liés à la multimodalité. Ce faisant, nous avons présenté les principaux capteurs employés en robotique mobile moderne, en portant une attention particulière au capteur caméra et à la formation des images, au cœur de cette problématique. Nous avons ensuite proposé un état de l'art articulé autour des méthodes de recalage de capteurs, que nous avons séparé en deux principales catégories. Nous avons considéré les méthodes dites directes, s'appuyant sur l'intégralité des données fournies, et les méthodes s'appuyant sur les éléments saillants, aussi appelés amers. Pour cette seconde catégorie, nous avons présenté la distinction entre méthodes reposant seulement sur l'aspect géométrique des amers, et les algorithmes s'appuyant sur la description de ces points d'intérêt et de leur voisinage.

Dans un second chapitre, nous avons développé un état de l'art des méthodes d'estimation de pose reposant sur les correspondances entre droites 2D et 3D. Après avoir introduit les méthodes dites générales, estimant une pose complète, nous avons présenté les algorithmes reposant sur une connaissance de la direction verticale pour réduire la complexité du problème. Nous avons alors introduit la première contribution majeure de ce travail de thèse. Cette contribution prend la forme d'une méthode originale d'estimation de pose s'appuyant sur une formulation des droites en coordonnées de Plücker et reposant sur la direction verticale, nommée VPnL_LS. En complément, nous avons présenté une méthode améliorant la précision de la pose à partir d'une adaptation de l'algorithme d'optimisation de Gauss-Newton, appelée VPnL_GN. Ces méthodes ont fait l'objet de publications dans un article de conférence et un journal. Afin de valider expérimentalement ces méthodes, nous avons proposé une étude exhaustive de la robustesse à l'aide de données simulées, en induisant un bruit sur la direction verticale et la définition des droites 2D et 3D, mais aussi sur des données réelles. Nous avons ainsi mesuré les performances en se confrontant à trois jeux de données, l'un simulé, le second constitué pour l'occasion au sein du laboratoire IRSEEM, et le troisième extrait du célèbre jeu de données public KITTI. Les résultats présentés confirment la supériorité de nos deux méthodes en comparaison aux algorithmes de l'état de l'art, y compris dans des cas où l'erreur issue des capteurs est particulièrement importante. Pourtant, ces expérimentations sont réalisées dans le cas théorique de l'estimation de pose, où chaque correspondance entre amers est connue. En pratique, une partie de ces correspondances

est *a minima* incorrecte. Dans les cas les plus réalistes, l'appariement d'amers est à effectuer avant toute estimation de pose. Nous avons ainsi évalué l'impact de mauvais appariements sur nos méthodes et les algorithmes de l'état de l'art. Les résultats à l'issue de l'expérimentation dans cette configuration plus réaliste ont alors démontré l'intérêt d'une approche robuste permettant d'écarter les paires aberrantes.

Au sein du troisième chapitre, nous avons développé la problématique de robustesse de l'estimation de pose en nous concentrant sur le rejet de mauvaises correspondances entre amers 2D et amers 3D, mais aussi en nous consacrant à la problématique d'appariement de droites. Dans cette optique, nous avons dans un premier temps introduit les métriques spécifiques au problème, puis présenté un bref état de l'art sur le rejet de paires de droites aberrantes. En développant l'aspect combinatoire du problème d'appairage, nous avons démontré la nécessité d'une approche diminuant le nombre de correspondances nécessaires à l'appariement. Dans cette optique, nous avons proposé deux algorithmes d'appariement de droites, pouvant aussi être employés pour le rejet de paires d'amers aberrantes. Basés sur la méthode RANSAC, ces algorithmes emploient respectivement deux et une seule paire(s) de droite(s) et la connaissance *a priori* de la direction verticale pour écarter les paires invalides. Cette opération est réalisée en optimisant un critère géométrique basé uniquement sur l'orientation du système. Nommés en conséquence RANSAC2 et RANSAC1, ces algorithmes ont fait l'objet d'une communication en conférence d'une publication dans un journal. Nous avons évalué les performances de ces algorithmes successivement dans des tâches d'appariement et de rejet de paires aberrantes d'amer à l'aide d'un jeu de données simulées, et des jeux de données IRSEEM et KITTI. Dans cette première tâche, les résultats placent nos algorithmes devant les méthodes de l'état de l'art, pour l'ensemble des métriques employées. Concernant l'appariement de droites, les résultats sont de nature plus mitigée. En simulation, l'algorithme RANSAC1 fournit de meilleurs résultats que les algorithmes de l'état de l'art, tandis que RANSAC2 donne des performances équivalentes. L'expérimentation sur des données réelles permet cependant de présenter les faiblesses de ces algorithmes. En s'appuyant uniquement sur l'orientation des droites, certaines configurations géométriques spécifiques, particulièrement sur le jeu de données IRSEEM, mettent en défaut le processus d'appariement de droites. En effet, l'orientation seule des droites ne permet pas de discriminer efficacement les droites 3D parallèles. Plus rares dans la séquence KITTI choisie, ces configurations sont néanmoins présentes et induisent des erreurs importantes dans l'estimation de pose subséquente à l'appariement, particulièrement sur la position du véhicule. En l'état actuel, les résultats confirment les performances supérieures de RANSAC1 et RANSAC2 dans les tâches de rejet de paires aberrantes, mais ne permettent pas de considérer ces algorithmes comme seuls acteurs d'un appariement.

La question de l'amélioration de la robustesse des méthodes d'estimation de pose multimodale demeure une problématique complexe. Au cours de ces travaux de thèse, nous avons considéré ce problème par une approche purement géométrique, et en considérant une extraction manuelle d'amers dans des images et nuages de points. De nombreuses pistes d'amélioration ou de compléments aux méthodes proposées sont envisageables.

En se consacrant pleinement à l'estimation de pose, nous avons exclu la première étape essentielle au recalage de capteurs. En effet, l'extraction manuelle de droites n'est pas souhaitable dans une application de robotique autonome. Si la détection de droites dans une image est un sujet de recherche particulièrement étudié, le pendant dans les nuages de points est nettement plus anecdotique dans la littérature. Les solutions existantes pour ce second cas sont pour la plupart relativement coûteuses en calcul, de par le volume de données important que génèrent les capteurs LiDAR. Une première perspective d'amélioration des travaux de recherche présentés dans ce mémoire réside donc dans une contribution à l'extraction de droites 3D d'un nuage de points.

Si les algorithmes d'estimation de pose introduits dans ce mémoire se concentrent exclusivement sur l'utilisation de droites, certaines méthodes hybrides employant une combinaison de droites et de points présentent des résultats notables. Il en effet difficile d'extraire des points possédant une correspondance entre des modalités LiDAR et caméra. L'emploi de ces éléments, en combinaison avec des droites, pourrait fournir une opportunité supplémentaire d'améliorer la robustesse de l'appariement.

La précision des deux algorithmes RANSAC1 et RANSAC2 introduits dans ce mémoire peut s'avérer insuffisante pour réaliser un appariement fiable pour des configurations géométriques de droites spécifiques. Néanmoins, la rapidité d'exécution de ces deux méthodes présentées permettent d'envisager sérieusement une méthode d'appariement hybride, reposant sur un pré-appariement par RANSAC1. Cet appairage préalable pourrait ensuite être complété et affiné par des méthodes reposant à la fois sur une optimisation de l'orientation et de la translation pour rejeter les correspondances aberrantes de droites. Le développement d'une telle méthode serait une contribution intéressante à la littérature.

Bibliographie

The 3dtk toolkit (slam6d), 2019. URL `https://sourceforge.net/projects/slam6d/`.

YI Abdel-Aziz, HM Karara et Michael Hauck : Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. *Photogrammetric Engineering & Remote Sensing*, 81(2):103–107, 2015.

Cuneyt Akinlar et Cihan Topal : Edlines : A real-time line segment detector with a false detection control. *Pattern Recognition Letters*, 32(13):1633–1642, 2011.

Cenek Albl, Zuzana Kukelova et Tomas Pajdla : Rolling shutter absolute pose problem with known vertical direction. *In Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3355–3363, 2016.

Adnan Ansar et Konstantinos Daniilidis : Linear pose estimation from points or lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):578–589, 2003.

Paul E Anuta : Spatial registration of multispectral and multitemporal digital imagery using fast fourier transform techniques. *IEEE transactions on Geoscience Electronics*, 8(4):353–368, 1970.

Daniel I Barnea et Harvey F Silverman : A class of algorithms for fast digital image registration. *IEEE transactions on Computers*, 100(2):179–186, 1972.

Adrien Bartoli et Peter Sturm : The 3d line motion matrix and alignment of line reconstructions. *In IEEE International Conference on Computer Vision and Pattern Recognition*, 2001.

Herbert Bay, Tinne Tuytelaars et Luc Van Gool : Surf : Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

Ilker Bekmezci, Ozgur Koray Sahingoz et Ş̧amil Temel : Flying ad-hoc networks (fanets) : A survey. *Ad Hoc Networks*, 11:1254–1270, 2013.

Jon Louis Bentley : Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Rikard Berthilsson : Affine correlation. *In Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No. 98EX170)*, volume 2, pages 1458–1460. IEEE, 1998.

Paul J Besl et Neil D McKay : Method for registration of 3-d shapes. *In Sensor fusion IV : control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

Rémi Boutteau : *Reconstruction tridimensionnelle de l'environnement d'un robot mobile, à partir d'informations de vision omnidirectionnelle, pour la préparation d'interventions.* Thèse de doctorat, 2010.

Ronald Newbold Bracewell et Ronald N Bracewell : *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

Pascal Brand, Roger Mohr et Philippe Bobet : Distorsions optiques : Correction dans un modele projectif. 1993.

Lisa Gottesfeld Brown : A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376, 1992.

J Brian Burns, Allen R Hanson et Edward M Riseman : Extracting straight lines. *IEEE transactions on pattern analysis and machine intelligence*, (4):425–455, 1986.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid et John J Leonard : Past, present, and future of simultaneous localization and mapping : Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

Michael Calonder, Vincent Lepetit, Christoph Strecha et Pascal Fua : Brief : Binary robust independent elementary features. *In European conference on computer vision*, pages 778–792. Springer, 2010.

Dylan Campbell et Lars Petersson : Gogma : Globally-optimal gaussian mixture alignment. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5685–5694, 2016.

John Canny : A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

Luciano Cantelli, M Mangiameli, C Donato Melita et Giovanni Muscato : Uav/ugv cooperation for surveying operations in humanitarian demining. *In 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2013.

TS Chan et Raymond KK Yip : Line detection algorithm. *In Proceedings of 13th International Conference on Pattern Recognition*, volume 2, pages 126–130. IEEE, 1996.

Homer H Chen : Pose determination from line-to-plane correspondences : existence condition and closed-form solutions. *In IEEE International Conference on Computer Vision*, pages 374–378, 1990.

Qin-sheng Chen, Michel Defrise et Frank Deconinck : Symmetric phase-only matched filtering of fourier-mellin transforms for image registration and recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):1156–1168, 1994.

Yang Chen et Gérard Medioni : Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

G. Christie, A. Shoemaker, K. Kochersberger, P. Tokebar, L. McLean et A. Leonessa : Radiation search operations using scene understanding with autonomous uav and ugv. août 2016. URL `https://arxiv.org/pdf/1609.00017.pdf`.

Tingting Cui, Shunping Ji, Jie Shan, Jianya Gong et Kejian Liu : Line-based registration of panoramic images and lidar point clouds for mobile mapping. *Sensors*, 17(1):70, 2016.

Philip David, Daniel DeMenthon, Ramani Duraiswami et Hanan Samet : Simultaneous pose and correspondence determination using line features. *In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2003.

E De Castro et C Morandi : Registration of translated and rotated images using finite fourier transforms. *IEEE Transactions on pattern analysis and machine intelligence*, (5):700–703, 1987.

Agnès Desolneux, Lionel Moisan et Jean-Michel Morel : Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.

Agnes Desolneux, Lionel Moisan et Jean-Michel Morel : *From gestalt theory to image analysis : a probabilistic approach*, volume 34. Springer Science & Business Media, 2007.

Michel Dhome, Marc Richetin, J-T Lapreste et Gerard Rives : Determination of the attitude of 3d objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, 1989.

Michael Dille, Ben Grocholsky, Stephen Thomas Nuske, Mark Moseleyy et Sanjiv Singh : Air-ground collaborative surveillance with human-portable hardware. 2011.

edmundoptics.fr : Spectre électromagnétique. `edmundoptics.fr/resources/application-notes/imaging/what-is-swir/`, 2019.

A Etemadi : Robust segmentation of edge data. *In 1992 International Conference on Image Processing and its Applications*, pages 311–314. IET, 1992.

Y Feng, A Schlichting et C Brenner : 3d feature point extraction from lidar data using a neural network. *In 23rd International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences Congress, ISPRS 2016, 12–19 July 2016, Prague, Czech Republic*. Göttingen : Copernicus GmbH, 2016.

Leandro AF Fernandes et Manuel M Oliveira : Real-time line detection through an improved hough transform voting scheme. *Pattern recognition*, 41(1):299–314, 2008.

Luis Ferraz, Xavier Binefa et Francesc Moreno-Noguer : Very fast solution to the pnp problem with algebraic outlier rejection. *In IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014.

Martin A Fischler et Robert C Bolles : Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Hassan Foroosh, Josiane B Zerubia et Marc Berthod : Extension of phase correlation to subpixel registration. *IEEE transactions on image processing*, 11(3):188–200, 2002.

Christian Forster, Simon Lynen, Laurent Kneip et Roland Siegwart : Collaborative visual slam with multiple mavs. 2012.

Vincent Fremont, Philippe Bonnifait *et al.* : Extrinsic calibration between a multi-layer lidar and a camera. *In Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 214–219. IEEE, 2008.

C Galambos, J Kittler et J Matas : Gradient based progressive probabilistic hough transform. *IEE Proceedings-Vision, Image and Signal Processing*, 148(3):158–165, 2001.

Abel Gawel, Titus Cieslewski, Renaud Dubé, Mike Bosse, Roland Siegwart et Juan Nieto : Structure-based vision-laser matching. *In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 182–188. IEEE, 2016.

Andreas Geiger, Philip Lenz, Christoph Stiller et Raquel Urtasun : Vision meets robotics : The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

Andreas Geiger, Philip Lenz et Raquel Urtasun : Are we ready for autonomous driving ? the kitti vision benchmark suite. *In Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Gene H Golub et Christian Reinsch : Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

N Hanaizumi et S Fujimur : An automated method for registration of satellite remote sensing images. *In Proceedings of IGARSS'93-IEEE International Geoscience and Remote Sensing Symposium*, pages 1348–1350. IEEE, 1993.

Chris Harris et Mike Stephens : A combined corner and edge detector. 1988.

Richard Hartley et Andrew Zisserman : *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Richard I Hartley et Fredrik Kahl : Global optimization through rotation space search. *International Journal of Computer Vision*, 82(1):64–79, 2009.

Jared Heinly, Enrique Dunn et Jan-Michael Frahm : Comparative evaluation of binary features. *In European Conference on Computer Vision*, pages 759–773. Springer, 2012.

Joel A Hesch et Stergios I Roumeliotis : A direct least-squares (dls) method for pnp. *In IEEE International Conference on Computer Vision*, pages 383–390, 2011.

Nora Horanyi et Zoltan Kato : Generalized pose estimation from line correspondences with known vertical direction. *In International Conference on 3D Vision (3DV)*, pages 244–253, 2017a.

Nora Horanyi et Zoltan Kato : Multiview absolute pose using 3d–2d perspective line correspondences and vertical direction. *In IEEE International Conference on Computer Vision Workshop*, pages 2472–2480, 2017b.

Harold Hotelling : Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Paul VC Hough : Method and means for recognizing complex patterns, décembre 18 1962. US Patent 3,069,654.

Daniel P Huttenlocher, William J Rucklidge et Gregory A Klanderman : Comparing images using the hausdorff distance under translation. *In Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 654–656. IEEE, 1992.

Roman Käslin, Péter Fankhauser, Elena Stumm, Zachary Taylor, Elias Mueggler, Jeffrey Delmerico, Davide Scaramuzza, Roland Siegwart et Marco Hutter : Collaborative localization of aerial and ground robots through elevation maps. *In Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*, pages 284–290. IEEE, 2016.

Ismail Khalid Kazmi, Lihua You et Jian Jun Zhang : A survey of 2d and 3d shape descriptors. *In 2013 10th International Conference Computer Graphics, Imaging and Visualization*, pages 1–10. IEEE, 2013.

Been Kim, Michael Kaess, Luke Fletcher, John Leonard, Abraham Bachrach, Nicholas Roy et Seth Teller : Multiple relative pose graphs for robust cooperative mapping. *In Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3185–3192. IEEE, 2010.

Nahum Kiryati, Yuval Eldar et Alfred M Bruckstein : A probabilistic hough transform. *Pattern recognition*, 24(4):303–316, 1991.

Hans Martin Kjer et Jakob Wilm : Evaluation of surface registration algorithms for pet motion correction. B.S. thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2010.

Solomon Kullback : *Information theory and statistics*. Courier Corporation, 1997.

Rakesh Kumar et Allen R Hanson : Robust methods for estimating pose and a sensitivity analysis. *CVGIP : Image understanding*, 60(3):313–342, 1994.

Louis Lecrosnier, Rémi Boutteau, Pascal Vasseur, Xavier Savatier et Friedrich Fraundorfer : Camera pose estimation based on pnl with a known vertical direction. *IEEE Robotics and Automation Letters*, 4(4):3852–3859, 2019.

Vincent Lepetit, Francesc Moreno-Noguer et Pascal Fua : Epnp : An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009.

Chao Li, Zhong Wang et Lin Li : An improved ht algorithm on straight line detection based on freeman chain code. *In 2009 2nd International Congress on Image and Signal Processing*, pages 1–4. IEEE, 2009.

Yangbin Lin, Cheng Wang, Jun Cheng, Bili Chen, Fukai Jia, Zhonggui Chen et Jonathan Li : Line segment extraction for large scale unorganized point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 102:172–183, 2015.

Tony Lindeberg : Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

Yuncai Liu, Thomas S. Huang et Olivier D. Faugeras : Determination of camera location from 2-d to 3-d line and point correspondences. *IEEE Transactions on pattern analysis and machine intelligence*, 12(1):28–37, 1990.

David G Lowe : Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal et Paul Suetens : Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16(2):187–198, 1997.

Walter F Mascarenhas : The divergence of the bfgs and gauss newton methods. *Mathematical Programming*, 147(1-2):253–276, 2014.

Jiri Matas, Charles Galambos et Josef Kittler : Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

Mathworks : kdtree, 2019. URL \url{https://fr.mathworks.com/help/stats/classification-using-nearest-neighbors.html}.

Pierre Merriaux : *Contribution à la localisation robuste embarquée pour la navigation autonome.* Thèse de doctorat, normandie université, 2016.

Pierre Merriaux, Yohan Dupuis, Rémi Boutteau, Pascal Vasseur et Xavier Savatier : A study of vicon system positioning performance. *Sensors*, 17(7):1591, 2017.

Krystian Mikolajczyk et Cordelia Schmid : A performance evaluation of local descriptors. 2005.

F Mirzaei et S Roumeliotis : Globally optimal pose estimation from line correspondences. *In IEEE International Conference on Robotics and Automation*, pages 5581–5588, 2011.

Hans P Moravec : Obstacle avoidance and navigation in the real world by a seeing robot rover. Rapport technique, Stanford Univ CA Dept of Computer Science, 1980.

Fernando Israel Ireta Munoz : *Global pose estimation and tracking for RGB-D localization and 3D mapping.* Thèse de doctorat, 2018.

Carl Olsson, Fredrik Kahl et Magnus Oskarsson : Branch-and-bound methods for euclidean registration problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31 (5):783–794, 2008.

Gaurav Pandey, James R McBride, Silvio Savarese et Ryan M Eustice : Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *Journal of Field Robotics*, 32(5):696–722, 2015.

Chavdar Papazov et Darius Burschka : Stochastic global optimization for robust point set registration. *Computer Vision and Image Understanding*, 115(12):1598–1609, 2011.

Alvaro Parra Bustos, Tat-Jun Chin et David Suter : Fast rotation search with stereographic projections for 3d registration. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3930–3937, 2014.

Nicolas Pinchon, Olivier Cassignol, Adrien Nicolas, Frédéric Bernardin, Patrick Leduc, Jean-Philippe Tarel, Roland Brémond, Emmanuel Bercier et Johann Brunet : All-weather vision for automotive safety : which spectral band ? *In International Forum on Advanced Microsystems for Automotive Applications*, pages 3–15. Springer, 2018.

Josien PW Pluim, JB Antoine Maintz et Max A Viergever : Mutual information matching in multiresolution contexts. *Image and Vision Computing*, 19(1-2):45–52, 2001.

Wiliam K Pratt : Digital image processing, new-york. *NY : John Wiley and Sons*, 1991.

Judith MS Prewitt : Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

Bronislav Přibyl, Pavel Zemčík et Martin Čadík : Camera pose estimation from lines using plucker coordinates. *arXiv :1608.02824*, 2016.

Bronislav Přibyl, Pavel Zemčík et Martin Čadík : Absolute pose estimation from line correspondences using direct linear transformation. *Computer Vision and Image Understanding*, 2017.

B Srinivasa Reddy et Biswanath N Chatterji : An fft-based technique for translation, rotation, and scale-invariant image registration. *IEEE transactions on image processing*, 5(8):1266–1271, 1996.

Nicola Ritter, Robyn Owens, James Cooper, Robert H Eikelboom et Paul P Van Saarloos : Registration of stereo and temporal images of the retina. *IEEE Transactions on medical imaging*, 18(5):404–418, 1999.

R Tyrrell Rockafellar et Roger J-B Wets : *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

Edward Rosten et Tom Drummond : Machine learning for high-speed corner detection. *In European conference on computer vision*, pages 430–443. Springer, 2006.

Ethan Rublee, Vincent Rabaud, Kurt Konolige et Gary R Bradski : Orb : An efficient alternative to sift or surf. *In ICCV*, volume 11, page 2. Citeseer, 2011.

Yohann Salaün, Renaud Marlet et Pascal Monasse : Multiscale line segment detector for robust and accurate sfm. *In Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2000–2005. IEEE, 2016.

Davide Scaramuzza et Friedrich Fraundorfer : Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

Davide Scaramuzza, Ahad Harati et Roland Siegwart : Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes. *In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 4164–4169. IEEE, 2007.

Aleksandr Segal, Dirk Haehnel et Sebastian Thrun : Generalized-icp. *In Robotics : science and systems*, volume 2, page 435. Seattle, WA, 2009.

Jacopo Serafin, Edwin Olson et Giorgio Grisetti : Fast and robust 3d feature extraction from sparse point clouds. *In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4105–4112. IEEE, 2016.

Jianbo Shi *et al.* : Good features to track. *In 1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE, 1994.

Manuel Silva, Ricardo Ferreira et José Gaspar : Camera calibration using a color-depth camera : Points and lines based dlt including radial distortion. *In Workshop Color-Depth Camera Fusion in Robotics, held with IROS*, 2012.

Aleksandra Sima et Simon Buckley : Optimizing sift for matching of short wave infrared and visible wavelength images. *Remote Sensing*, 5(5):2037–2056, 2013.

Andrew Simper : Correcting general band-to-band misregistrations. *In Proceedings of 3rd IEEE International Conference on Image Processing*, volume 2, pages 597–600. IEEE, 1996.

Stephen M Smith et Michael J Brady : Susan : A new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

Irwin Sobel et Gary Feldman : A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

Colin Studholme, Derek LG Hill et David J Hawkes : An overlap invariant entropy measure of 3d medical image alignment. *Pattern recognition*, 32(1):71–86, 1999.

Philippe Thévenaz et Michael Unser : An efficient mutual information optimizer for multiresolution image registration. *In Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, volume 1, pages 833–837. IEEE, 1998.

Tinne Tuytelaars, Krystian Mikolajczyk *et al.* : Local invariant feature detectors : a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008.

Rafael Grompone Von Gioi, Jeremie Jakubowicz, Jean-Michel Morel et Gregory Randall : Lsd : A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2010.

Dejan V Vranic et Dietmar Saupe : *3D model retrieval.* Thèse de doctorat, University of Leipzig PhD thesis, 2004.

William M Wells III, Paul Viola, Hideki Atsumi, Shin Nakajima et Ron Kikinis : Multimodal volume registration by maximization of mutual information. *Medical image analysis*, 1 (1):35–51, 1996.

Wikipedia : Ccd-sensor — wikipedia, die freie enzyklopädie, 2019. URL `\url{https://de.wikipedia.org/w/index.php?title=CCD-Sensor&oldid=191952909}`. [Online ; Stand 19. Oktober 2019].

Wikipedia contributors : Active-pixel sensor — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Active-pixel_sensor&oldid=920630908`, 2019a. [Online ; accessed 19-October-2019].

Wikipedia contributors : Aperture — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Aperture&oldid=915252157`, 2019b. [Online ; accessed 17-October-2019].

Wikipedia contributors : Spherical aberration — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Spherical_aberration&oldid=912217022`, 2019c. [Online ; accessed 17-October-2019].

Wikipédia : Scale-invariant feature transform — wikipédia, l'encyclopédie libre, 2018. URL `\url{http://fr.wikipedia.org/w/index.php?title=Scale-invariant_feature_transform&oldid=153753145}`. [En ligne ; Page disponible le 7-novembre-2018].

Wikipédia : Aberration chromatique — wikipédia, l'encyclopédie libre, 2019a. URL `\url{http://fr.wikipedia.org/w/index.php?title=Aberration_chromatique&oldid=161451978}`. [En ligne ; Page disponible le 1-août-2019].

Wikipédia : Détection de contours — wikipédia, l'encyclopédie libre, 2019b. URL `\url{http://fr.wikipedia.org/w/index.php?title=D%C3%A9tection_de_contours&oldid=156146784}`. [En ligne ; Page disponible le 25-janvier-2019].

Wikipédia : Lidar — wikipédia, l'encyclopédie libre, 2019c. URL `\url{http://fr.wikipedia.org/w/index.php?title=Lidar&oldid=162806650}`. [En ligne ; Page disponible le 19-septembre-2019].

George Wolberg et Siavash Zokai : Image registration for perspective deformation recovery. *In Automatic Target Recognition X*, volume 4050, pages 259–270. International Society for Optics and Photonics, 2000a.

George Wolberg et Siavash Zokai : Robust image registration using log-polar transform. *In Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 1, pages 493–496. IEEE, 2000b.

Ryan W Wolcott et Ryan M Eustice : Visual localization within lidar maps for automated urban driving. *In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 176–183. IEEE, 2014.

Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss et Wolfram Burgard : Octomap : A probabilistic, flexible, and compact 3d map representation for robotic systems. *In Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.

Chi Xu, Lilian Zhang, Li Cheng et Reinhard Koch : Pose estimation from line correspondences : A complete analysis and a series of solutions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1209–1222, 2017.

Lei Xu, Erkki Oja et Pekka Kultanen : A new curve detection method : randomized hough transform (rht). *Pattern recognition letters*, 11(5):331–338, 1990.

Jiaolong Yang, Hongdong Li et Yunde Jia : Go-icp : Solving 3d registration efficiently and globally optimally. *In Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2013.

Andy Zeng, Shuran Song, Matthias Niessner, Matthew Fisher, Jianxiong Xiao et Thomas Funkhouser : 3dmatch : Learning local geometric descriptors from rgb-d reconstructions. 2016.

Lilian Zhang, Chi Xu, Kok-Meng Lee et Reinhard Koch : Robust and efficient pose estimation from line correspondences. *In Asian Conference on Computer Vision*, pages 217–230. Springer, 2012a.

Xiaohu Zhang, Zheng Zhang, You Li, Xianwei Zhu, Qifeng Yu et Jianliang Ou : Robust camera pose estimation from unknown or known line correspondences. *Applied optics*, 51 (7):936–948, 2012b.

Yueqiang Zhang, Xin Li, Haibo Liu et Yang Shang : Probabilistic approach for maximum likelihood estimation of pose using lines. *IET Computer Vision*, 10(6):475–482, 2016.

Zhiqiang Zheng, Han Wang et Eam Khwang Teoh : Analysis of gray level corner detection. *Pattern Recognition Letters*, 20(2):149–162, 1999.

Barbara Zitova et Jan Flusser : Image registration methods : a survey. *Image and vision computing*, 21(11):977–1000, 2003.

Liste des publications

Journaux

— Louis Lecrosnier, Rémi Boutteau, Pascal Vasseur, Xavier Savatier et Friedrich Fraundorfer. « Camera Pose Estimation Based on PnL With a Known Vertical Direction ». *IEEE Robotics and Automation Letters*, IEEE 2019, 4 (4), pp.3852-3859. ⟨10.1109/LRA.2019.2929982⟩

Conférences

— Louis Lecrosnier, Rémi Boutteau, Pascal Vasseur, Xavier Savatier et Friedrich Fraundorfer. « Camera Pose Estimation Based on PnL With a Known Vertical Direction ». *IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS 2019*

— Louis Lecrosnier, Rémi Boutteau, Pascal Vasseur, Xavier Savatier et Friedrich Fraundorfer. « Vision based vehicle relocalization in 3D line-feature map using Perspective-n-Line with a known vertical direction », *IEEE Intelligent Transportation Systems Conference - ITSC 2019*

Estimation de pose multimodale - Approche robuste par les droites 2D et 3D

Résumé : Avec la complexification des tâches confiées aux robots, la perception de l'environnement, aussi appelée perception de scène, doit se faire de manière plus complète. L'emploi simultané de différents types de capteurs, ou multimodalité, est l'un des leviers employés à cet effet. L'exploitation de données issues de modalités différentes nécessite généralement de connaître la pose, c'est-à-dire l'orientation et la position, de chaque capteur relativement aux autres.

Les travaux présentés dans cette thèse se concentrent sur la problématique d'estimation de pose robuste dans le cas d'une multimodalité incluant des capteurs de type caméra et LiDAR. Deux contributions majeures sont présentées dans ce manuscrit. On présente dans un premier temps un algorithme d'estimation de pose original s'appuyant sur la correspondances entre droites 2D et 3D, ainsi qu'une connaissance a priori de la direction verticale. Dans l'optique d'améliorer la robustesse de cet algorithme, une deuxième contribution repose sur un algorithme d'appariement de droites et de rejet de paires aberrantes basé RANSAC. Cette méthode fonctionne à l'aide de deux ou d'une seule paire de droites, diminuant le coût en calcul du problème.

Les résultats obtenus sur des jeux de données simulées et réelles démontrent une amélioration des performances en comparaison avec les méthodes de l'état de l'art.

Mots clés : Estimation de pose robuste, multimodalité vision-LiDAR, robotique mobile, appariement d'amers géométriques, Perspective-n-Line

Multimodal pose estimation - Robust approach using 2D and 3D lines

Abstract : Camera pose estimation consists in determining the position and the orientation of a camera with respect to a reference frame. In the context of mobile robotics, multimodality, *i.e.* the use of various sensor types, is often a requirement to solve complex tasks. However, knowing the orientation and position, *i.e.* the pose, of each sensor regarding a common frame is generally necessary to benefit multimodality.

In this context, we present two major contributions with this PhD thesis. First, we introduce a pose estimation algorithm relying on 2D and 3D line and a known vertical direction. Secondly, we present two outliers rejection and line pairing methods based on the well known RANSAC algorithm. Our methods make use of the vertical direction to reduce the number of lines required to 2 and 1, i.e. RANSAC2 and RANSAC1.

A robustness evaluation of our contributions is performed on simulated and real data. We show state of the art results.

Keywords : Robust pose estimation, LiDAR-vision multimodality, mobile robotics, geometric feature pairing, Perspective-n-Line;