LiDAR-based point clouds registration for localization in indoor environments

Ketty Favre

▶ **To cite this version:**

Ketty Favre. LiDAR-based point clouds registration for localization in indoor environments. Signal and Image Processing. Université Rennes 1, 2021. English. NNT : 2021REN1S059 . tel-03522998

HAL Id: tel-03522998

https://theses.hal.science/tel-03522998v1

Submitted on 12 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par
Ketty FAVRE

LiDAR-based point clouds registration for localization in indoor environments

Thèse présentée et soutenue à l'INSA Rennes, le 7 Octobre 2021
Unité de recherche : IETR (Institut d'Electronique et des Technologies du numéRique)

Rapporteurs avant soutenance :

Rémi BOUTTEAU	Professeur des Universités, Université de Rouen Normandie
Guillaume CARON	Maître de Conférences HDR, Université de Picardie Jules Verne

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Paul CHECCHIN	Professeur des Universités, Université de Clermont Auvergne
Examinateurs :	Rémi BOUTTEAU	Professeur des Universités, Université de Rouen Normandie
	Guillaume CARON	Maître de Conférences HDR, Université de Picardie Jules Verne
	Paul CHECCHIN	Professeur des Universités, Université de Clermont Auvergne
	François POMERLEAU	Professor, Université Laval
	Myriam SERVIERES	Maitre de Conférences HDR, Centrale Nantes
Dir. de thèse :	Luce MORIN	Professeur des Universités, INSA Rennes
Co-dir. de thèse :	Eric MARCHAND	Professeur des Universités, Université de Rennes 1
Encadrante de thèse :	Muriel PRESSIGOUT	Maître de Conférences, INSA Rennes

REMERCIEMENTS

Je tiens tout d'abord à remercier mon équipe d'encadrement, Luce Morin, Eric Marchand et Muriel Pressigout, qui m'ont aidée et guidée durant cette thèse. Je les remercie pour leurs (nombreuses) remarques méticuleuses et toujours constructives, ainsi que pour la confiance qu'ils m'ont témoignée au cours de ces trois années.

J'adresse mes remerciements à l'ensemble de l'équipe VAADER de l'IETR, au sein de laquelle ma thèse s'est déroulée. J'en retiens des échanges scientifiques toujours très enrichissant, mais également (surtout) des repas de Noël et des barbecues bien animés ! J'ai particulièrement apprécié organiser et participer à des sessions quiz et jeux de société avec vous, dommage qu'une certaine pandémie ait freiné notre élan.

J'ai une pensée plus particulière pour mes compagnons doctorants : nos thèses respectives se sont déroulées dans des conditions bien étranges. Il n'a pas toujours été facile de garder le cap, mais ce fut un plaisir de partager ce long et périlleux voyage avec eux. Me voilà maintenant arrivée à destination et c'est en grande partie à eux que je le dois.

Je remercie également mes amis qui ont su comprendre quand on pouvait me parler de la thèse et quand il valait mieux, parfois, me parler plutôt de la pluie et du beau temps (et du dernier chapitre sorti de Shingeki no kyojin)... Alors merci Olivier, Romain, Quentin, Vincent (- H), Vincent (Pripri), Alex, François, Judith, d'avoir contribué indirectement à mes travaux. Un grand merci à Julien pour son soutien quotidien. Toujours confiant dans mes capacités, sans lui cette thèse se serait déroulée avec beaucoup plus de stress et d'angoisses.

Ces remerciements ne peuvent s'achever sans une énorme pensée pour mes parents. Ils m'ont appris l'importance du travail et des études et m'ont soutenue jusqu'au bout. Ce manuscrit leur est dédié.

Table of Contents

List of Figures

List of Tables

RÉSUMÉ EN FRANÇAIS

Pour l'être humain, il est facile de se situer dans son environnement. Grâce à ce que nous percevons avec nos yeux, le cerveau humain est capable de décoder les informations qu'il voit, pour comprendre immédiatement où il se trouve. Cette notion n'est pas directement transposable aux entités artificielles. Des capteurs ont été créés pour reproduire la notion de vision humaine pour les plateformes robotiques, tels que des caméras, des radars ou encore des LiDARs.

Ces capteurs vont permettre de générer des données brutes directement liées au milieu environnant, dans leur repère local. L'objectif est d'estimer les mouvements relatifs de l'entité portant le capteur afin de la placer dans un repère global, autrement dit, d'estimer sa pose dans ce repère. Pour ce faire, des primitives, le plus souvent géométriques, sont utilisées. D'une capture à l'autre, les primitives pourront être identifiées et mises en correspondance. Ce processus de mise en correspondance permet d'établir une fonction dite de coût (par exemple une erreur ou une distance à minimiser) qui varie en fonction des paramètres qui représentent le mouvement relatif entre les captures (la transformation). Lorsque la fonction de coût atteint son minimum, les paramètres de transformation estimés sont optimaux. Ce type de problème d'estimation de pose, également appelé problème de recalage, a été largement étudié et est encore plus source d'émulation de nos jours. Les études diffèrent en fonction de la diversité des capteurs ou des applications, et l'expansion de la qualité de la technologie qui y est relative en fait un sujet encore plus dynamique.

Ainsi, de nombreux types de données permettant de répondre au problème de localisation peuvent être trouvés, en fonction du capteur dont elles sont issues. Le type de capteur va également déterminer le type de transformation permettant de relier les données. Dans cette thèse, le choix a été fait d'utiliser une configuration utilisant uniquement un capteur LiDAR afin de résoudre le problème de recalage. Un LiDAR génère des données sous la forme d'un nuage de points. Les nuages de points peuvent être en deux ou trois dimensions, dans notre cas, nous nous sommes concentrés sur les applications 3D. La transformation liant les nuages de points est dite rigide, ce qui signifie que seules les translations et rotations relatives entre les scans sont affectées et doivent être estimées.

L'objectif global de cette thèse est d'améliorer les méthodes de recalage basées unique-

ment sur des données LiDAR. Nous nous sommes concentrés sur la façon de tirer profit des environnements structurés pour fournir des algorithmes qui permettent un compromis entre la précision, la robustesse et le temps de calcul en réduisant la dimensionnalité du problème de recalage.

0.1 Contexte

0.1.1 Capteur LiDAR et applications

Description

Comme nous le disions précédemment, de nombreux types de capteurs permettant de résoudre la problématique de recalage existent, chacun avec leurs avantages et désavantages. Parmi ceux là on retrouve les caméras, très utilisées grâce à la grande quantité d'information qu'elles fournissent grâce aux images, et les capteurs Light Detection And Ranging (LiDAR) qui vont nous intéresser plus particulièrement dans cette thèse. Bien que plus chers que les caméras, les LiDARs sont plus précis et ne sont pas sensibles au changement de luminosité ambiante. Ils permettent également des acquisitions à 360°.

Un capteur LiDAR va permettre d'acquérir la distance entre lui et l'objet que son laser frappe. Son fonctionnement est donc très proche de celui d'un radar, mais ce sont ici les ondes lumineuses qui sont traitées. On retrouve plusieurs types de LiDAR, nos travaux se concentrent sur les LiDAR 3D. Les données générées sont, en général, un jeu de points 3D, aussi appelé nuage de points 3D. Un exemple simplifié de capture de nuage de points est donné en Figure 1.

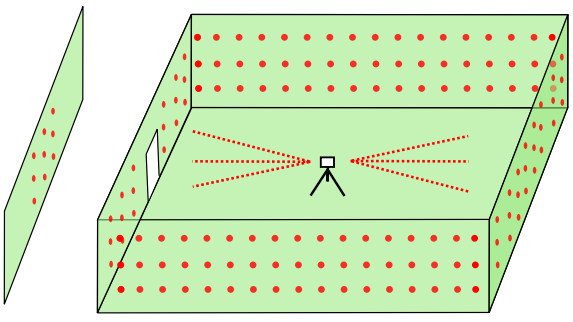


Figure 1 – Représentation d'un LiDAR 3D faisant une acquisition d'un nuage de points dans une salle. Ici, le LiDAR possède trois rayons et tourne sur lui-même. Un nuage de points 3D est généré (points rouges). Une porte est ouverte, on voit ainsi que des points sont capturés à l'extérieur de cette salle.

Applications

De nos jours on retrouve les capteurs de type LiDAR dans toutes sortes d'applications. Par exemple, ils sont aujourd'hui utilisés dans le domaine du génie civil afin de faire de la modélisation 3D de bâtiments. La Cathédrale de Notre-Dame, avant son incendie en 2019, avait été totalement modélisée. Ces modèles peuvent aujourd'hui être utilisés dans sa reconstruction. On note également des utilisations aériennes en archéologie. Ces dernières ont pu permettre la découverte des vestiges d'une ville Maya au Guatemala en 2020. On retrouve également aujourd'hui les LiDARs sur des véhicules autonomes.

0.1.2 Formulation de l'ICP

Nos travaux présentent des algorithmes permettant le recalage de nuages de points 3D. L'une des approches les plus populaires pour recaler des nuages de points 3D est l'algorithme Iterative Closest Point (ICP). Cette section présente un état de l'art global de cette méthode.

La première formulation de l'ICP nous est donnée dans [Besl et al. 1992]. Il permet de calculer la transformation rigide (rotation et translation) qui relie un nuage de points 3D source et un nuage de points 3D cible. Pour ce faire, chaque point du nuage source est apparié avec son plus proche voisin dans le nuage cible. Ensuite, la transformation rigide 3D qui minimise la distance entre les points appariés est estimée. Cette opération est réalisée selon un schéma itératif jusqu'à ce que l'erreur résiduelle atteigne le seuil souhaité.

Le recalage de type ICP peut être divisé en deux catégories : le recalage des caractéristiques locales et celui des caractéristiques globales. Les premières approches de l'algorithme ICP utilisaient principalement les caractéristiques locales. [Besl et al. 1992] utilise la distance point-à-point et [Chen et al. 1992] la distance point-à-plan, qui s'avère être plus robuste et converge plus rapidement que la distance point-à-point. Une résolution linéaire de la minimisation de la distance point-à-plan peut être trouvée dans [Low 2004] en utilisant l'approximation des petits angles. Une résolution non linéaire de Levenberg-Marquardt utilisant des estimateurs robustes est proposée dans [Fitzgibbon 2003]. Les méthodes de recalage local peuvent également extraire des points clés des nuages de points pour établir des correspondances. Dans [Magnusson et al. 2015], Normal Distribution Transform (NDT) prend en compte les structures de surface locales autour de chaque point. Dans l'approche d'ICP généralisée (GICP (Generalized-ICP)) proposée dans [Segal et al. 2009], le voisinage local des points est utilisé afin d'assimiler cette structure à des patchs planaires. De même que pour les approches point-à-plan, les normales locales du nuage de points cible sont prises en compte ainsi que celles du nuage de points source. Ces méthodes offrent des performances satisfaisantes mais prennent généralement

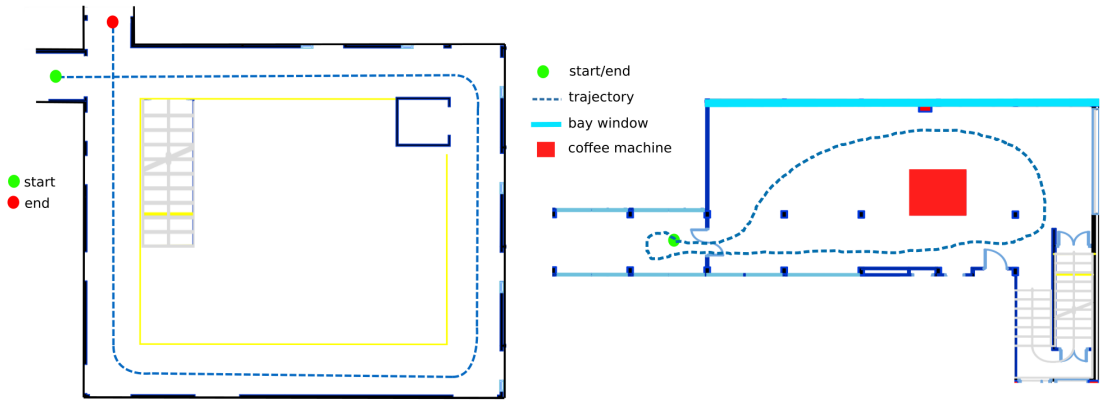
beaucoup de temps lorsqu'il s'agit de recaler de grands nuages de points.

Une façon de surmonter le problème du nombre de points est d'utiliser des caractéristiques globales, telles que les plans. Dans ces approches, la première étape consiste à segmenter les nuages de points en patchs planaires [Zhou et al. 2016][Chen et al. 2020][Zong et al. 2019], puis à utiliser les caractéristiques planaires pour établir les correspondances. Pour faire correspondre les patchs planaires, on peut trouver plusieurs approches. Dans le cas de [Grant et al. 2019], le débit d'acquisition des données est supposé élevé, ce qui entraîne un faible mouvement relatif d'un scan à l'autre. Ainsi, pour déterminer les correspondances de plans, les plans proches les uns des autres et presque parallèles sont appariés. Cette approche donne de bons résultats dans ce type de scénario, mais peine à estimer une trajectoire correcte en dehors de faibles déplacement entre deux scans. Dans [Chen et al. 2020], un descripteur plan/ligne est proposé pour établir les correspondances de structure. Il conduit à un recalage grossier efficace. Dans [Zong et al. 2019], les formes, les surfaces et les normales des patchs planaires sont considérées pour trouver les meilleures correspondances. Dans [Pathak et al. 2009], une cartographie 3D basée sur les segments de plan est proposée.

0.2 LOOP'IN, un jeu de données permettant l'évaluation de la fermeture de boucle

La première contribution de cette thèse est un jeu de données appelé LOOP'IN [1]. Il s'agit d'un jeu de données composé de deux longues séquences capturées grâce à un LiDAR VLP-16 Puck Hi-Res monté sur une plateforme mobile poussée par un opérateur. Il est conçu pour évaluer la capacité des algorithmes de recalage de nuages de points à fermer des boucles avec des données réelles, dans un environnement intérieur. Comme nous n'avions pas la possibilité de mesurer précisément la réalité terrain des poses de notre capteur, nous avons ajouté des boucles dans les trajectoires, ainsi, il est possible de vérifier si l'algorithme a subi une dérive ou s'il est capable de fermer correctement la boucle. Ce jeu de données est utilisé sur tous les algorithmes issus de cette thèse.

1. https://github.com/kfavre/LOOP-IN_dataset.



(a) Boucle *Balcony* (b) Boucle *Coffee Hall*

Figure 2 – jeu de données LOOP'IN

0.3 Recalage multi-résolution de nuages de points 3D

Dans cette seconde contribution, nous présentons un algorithme de recalage basé sur un processus multi-résolution appelé Gauss-Newton based Multi-Resolution ICP (GNMR-ICP).

0.3.1 Méthode proposée

Afin d'assurer une minimisation rapide et robuste pour estimer les paramètres de transformation, nous minimisons la distance point-à-plan en utilisant une méthode de Gauss-Newton. Pour réduire l'influence des valeurs aberrantes (causées par le processus d'appariement des points), des fonctions robustes de type M-estimateurs sont ajoutées dans le processus de minimisation. Le processus multi-résolution est possible grâce à la représentation du nuage de points sous forme d'octree. La structure de l'algorithme est fournie en Figure 3.

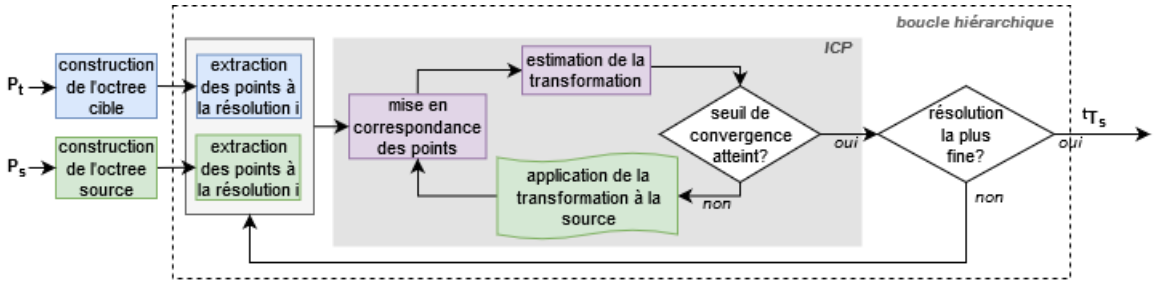


Figure 3 – Structure de GNMR-ICP

0.3.2 Expériences

Des expériences sont menées sur le jeu de données de référence Autonomous Systems Lab (ASL)[Pomerleau et al. 2012] pour évaluer la robustesse et la précision de la méthode proposée. Ce jeu de données est fourni avec la vérité terrain des poses du capteur, nous permettant de calculer la distance Euclidienne ainsi que la distance géodésique entre la pose que nous estimons et la pose réelle. Si la distance Euclidienne est inférieur à 0.1m et la distance géodésique à 2.5° le recalage est considéré réussi [Magnusson et al. 2015]. Ces résultats sont visibles dans la Table 1. Elle montre que GNMR-ICP est plus performant que son équivalent réalisant une minimisation par une méthode directe utilisant l'approximation des petits angles (noté SA-ICP), indépendamment des M-estimateurs choisis (Cauchy, Huber ou Tukey).

Table 1 – Pourcentage (%) de recalages réussis sur les séquences du jeu de données ASL.

Séquence	**SA-ICP**	**GNMR-ICP**$_{Huber}$	**GNMR-ICP**$_{Tukey}$	**GNMR-ICP**$_{Cauchy}$
Apartment	43	82	88	84
ETH	94	100	100	100
Gazebo	87	97	94	97
Mountain	73	97	93	93
Stairs	87	100	93	100
Wood	84	100	93	100

Une étude sur le nombre de niveaux de résolution à utiliser dans le processus multi-résolution montre également que plus le nombre de niveaux utilisés est grand, plus les résultats générés sont précis. Cela entraîne également, pour les environnements structurés une diminution du temps de calcul.

0.4 Recalage basé plan de nuages de points 3D

L'idée de la troisième contribution est d'exploiter les structures planaires identifiées dans la seconde. Ainsi, un algorithme appelé New Accurate Plane-based ICP (NAP-ICP) conçu pour recaler les plans est proposé.

0.4.1 Méthode proposée

Tout d'abord, il est nécessaire d'établir les correspondances entre les plans. Pour ce faire nous avons conçu une fonction de score basée sur des caractéristiques pondérées de correspondances de plans pour résoudre ce problème. Pour assurer une estimation robuste, une minimisation de la distance plan à plan en deux étapes est effectuée : tout d'abord, une minimisation par une méthode directe est effectuée en utilisant un processus RANdom SAmple Consensus (RANSAC) pour initialiser l'estimation et identifier les valeurs aberrantes. Ensuite, une minimisation itérative de type Gauss-Newton est effectuée sur la distance plan à plan afin d'affiner l'estimation. Un recalage supplémentaire point-à-plan est effectué à la fin du processus pour rendre l'estimation aussi précise que possible. La structure globale de NAP-ICP est présentée en Figure 4.

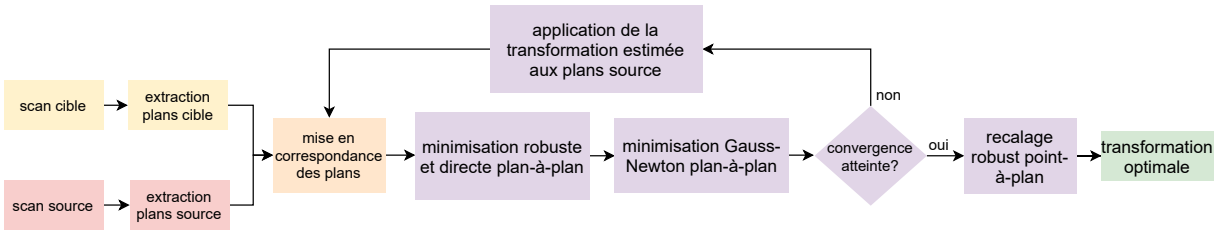


Figure 4 – Structure de NAP-ICP.

0.4.2 Expériences

Les expériences montreront l'importance et l'impact sur la précision de l'algorithme NAP-ICP de l'étape du RANSAC et du recalage point-à-plan supplémentaire. Par exemple en Figure 5, on voit que NAP-ICP en combinant recalage plan-à-plan et point-à-plan est capable d'estimer une trajectoire comparable avec celle de la vérité terrain, alors que sans l'étape du point-à-plan l'algorithme est moins précis.

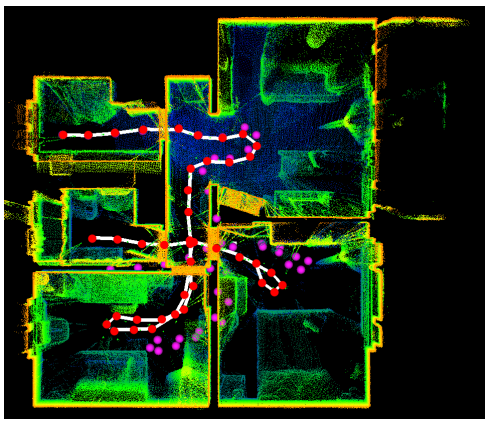


Figure 5 – Vue du dessus de la construction de carte 3D de la séquence *Apartment* en utilisant NAP-ICP. Les points de la carte sont colorés selon leur hauteur (le bleu correspond à la valeur la plus faible et le jaune la plus forte. Afin de rendre la lecture de la figure plus aisée, seule la construction de carte générée par l'algorithme NAP-ICP (complet). Dans cette même intention, le plafond a été retiré du nuage de point affiché ici. En blanc : la trajectoire réalité terrain. - En violet : la trajectoire calculée avec le recalage plan-à-plan **uniquement**. - En rouge : la trajectoire calculée avec l'algorithme **complet** de NAP-ICP (recalage plan-à-plan suivi du point-à-plan.)

La précision et la robustesse de la méthode proposée sont ensuite évaluées et comparées aux algorithmes de l'état de l'art grâce à la méthode présentée précédemment en section 0.3.2 se basant sur le jeu de données ASL. NAP-ICP montre qu'il est capable de surpasser les algorithmes de l'état de l'art dans ce contexte (100% de recalages valides sur l'ensemble des séquences *Apartment*, *ETH* et *Stairs*). Enfin, grâce à l'expérience sur le jeu de données LOOP'IN (présenté en section 0.2), nous démontrons que NAP-ICP est capable de fermer des boucles et aussi de rétablir la trajectoire estimée quand une erreur est faite lors d'un recalage sur une longue séquence.

0.5 Mise en correspondance de plans par apprentissage pour du recalage plan à plan

Le fonctionnement de la fonction de score de la contribution précédente a mis en évidence que les caractéristiques des paires de plans que nous avons choisies pour établir les correspondances entre les plans étaient pertinentes car le comportement de l'algorithme était celui attendu. Nous avons donc choisi pour notre quatrième contribution de proposer une méthode

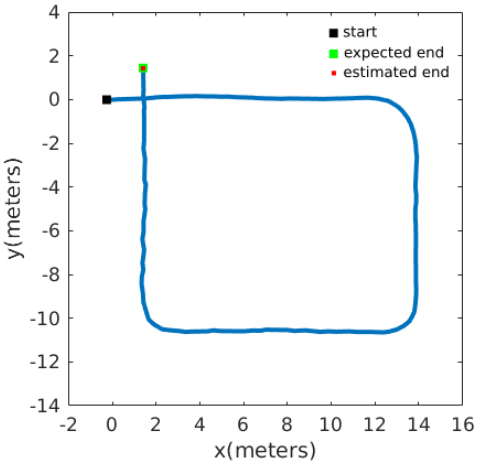
de recalage plan à plan, appelée Plane-based Accurate Registration ICP (PAR-ICP), dont la mise en correspondance est faite par apprentissage, en se basant sur ces caractéristiques de paires de plans.

0.5.1 Méthode proposée

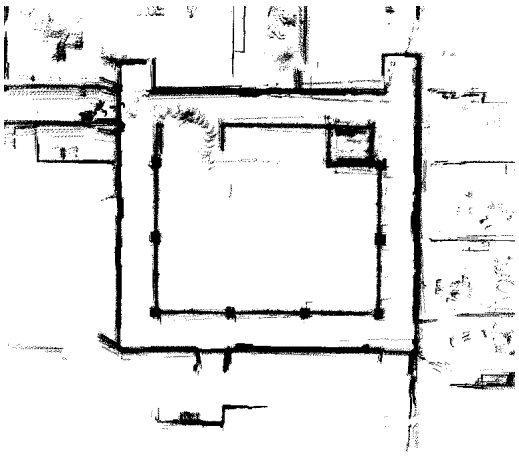
Le classifieur utilisé est basé sur un Random Forest entraîné grâce aux caractéristiques de paires de plans dans NAP-ICP. Les données utilisées pour l'apprentissage proviennent du jeu de données ASL incluant la vérité terrain de la pose du capteur, permettant d'identifier facilement les plans correspondants, et d'étiqueter les données à la main. Après entraînement, il faut noter que la précision de la classification sur le jeu de données de test n'est pas parfaite, mais assez bonne pour qu'elle soit gérée par l'algorithme RANSAC utilisé dans l'étape de minimisation.

0.5.2 Expériences

Les expériences montre qu'PAR-ICP est capable de recaler avec succès les séquences intérieures du jeu de données de données ASL, avec la méthode décrite en section 0.3.2. L'algorithme est également capable de fermer des boucles sur de longues séquences et de fournir des cartes incrémentales géométriquement cohérentes via des expériences sur le jeu de données LOOP'IN comme on peut le voir en Figure 6.



(a) Boucle *Balcony*



(b) Carte incrémentale de la séquence *Balcony*

Figure 6 – Trajectoire dans le plan xy calculée avec PAR-ICP et carte incrémentale de la séquence *Balcony* du jeu de données LOOP'IN. Les axes sont en mètres.

0.6 Conclusion et perspectives

Dans cette thèse nous avons abordé le sujet du recalage de nuages de points 3D. Notre objectif était d'améliorer les méthodes basées uniquement sur des données LiDAR. Nous avons fourni des algorithmes tirant parti des environnements intérieurs souvent très structurés, dans une volonté de proposer des méthodes permettant un compromis entre précision, robustesse et temps de calcul en réduisant la dimensionnalité du problème posé par le recalage.

La première contribution de cette thèse présente un dataset, LOOP'IN, créé pour évaluer la capacité des algorithmes de recalage à fermer des boucles. Les deux séquences proposées sont longues, permettant de vérifier si l'algorithme est sujet à une dérive.

Dans la seconde contribution nous avons proposé un algorithme multi-résolution nommé GNMR-ICP. Afin d'assurer une minimisation rapide et robuste, nous avons choisi de minimiser la distance point-à-plan en utilisant une méthode de Gauss-Newton. Des M-estimateurs ont été ajouté dans le processus afin de réduire l'impact des valeurs aberrantes lors de la minimisation.

Les expériences sur le jeu de données ASL ont montré que GNMR-ICP est plus précis et plus robuste que son équivalent se basant sur l'approximation des petits angles. Nous avons également noté que plus il y a de niveaux de recalage dans le processus multi-résolution, meilleurs étaient les résultats en termes de précision en plus de réduire les temps de calcul

dans les environnements très structurés.

Dans la troisième contribution, nous tirons parti des plans présents dans les environnements intérieurs. Nous avons créé un algorithme appelé NAP-ICP destiné à recaler les plans. Afin de s'assurer que le processus de minimisation soit robuste, il est effectué en deux étapes: d'abord, une minimisation directe est faite sur les plans, permettant d'identifier et de rejeter les valeurs aberrantes. Cette minimisation de la distance plan-à-plan est précédée par une mise en correspondance effectuée grâce à une fonction de score conçue dans ce but. La fonction permet de donner un score aux paires de plans qui permettent de savoir si oui ou non deux plans sont appariés correctement. Ensuite, une minimisation utilisant un Gauss-Newton est faite sur les plans restants afin de raffiner l'estimation. L'utilisation de plans a une tendance à lisser les données, donc, afin de rendre l'algorithme plus précis, un recalage point-à-plan supplémentaire est effectué en fin de processus.

NAP-ICP a été capable de recaler avec succès 100% des séquences intérieures du jeu de données ASL (selon les seuils choisis) et a fourni de meilleurs résultats que les méthodes de l'état de l'art évaluées. L'algorithme a également montré, sur LOOP'IN, qu'il était capable de fermer les boucles sur une longue séquence et qu'il pouvait rattraper une erreur faite pendant l'estimation.

La quatrième contribution avait pour but d'améliorer la fonction de score de NAP-ICP. C'est pourquoi nous avons proposé une méthode plan-à-plan, PAR-ICP, où la mise en correspondance s'effectue à l'aide d'une méthode d'apprentissage. Un classifieur de type Random Forest a été entraîné afin de classifier si deux plans étaient des correspondants ou non. Afin de l'entraîner, nous avons utiliser les mêmes caractéristiques que la fonction de score de NAP-ICP ainsi que les nuages de points du jeu de données ASL.

Les expériences ont montré que PAR-ICP était capable de recaler avec succès l'ensemble des séquences intérieures du jeu de données ASL (selon les seuils choisis). PAR-ICP a aussi permis de générer des cartes incrémentales cohérentes sur le jeu de données LOOP'IN et a pu fermer les boucles comme espéré.

Le contexte global de cette thèse était de résoudre le problème de recalage à des fins de localisation. Cela implique l'utilisation de plateformes mobiles et donc de systèmes embarqués. Avec un travail sur l'architecture et l'optimisation de nos algorithmes on peut imaginer les porter sur de telles plateformes.

Une suite logique de nos travaux serait de les intégrer dans un algorithmes de Simultaneous Localization And Mapping (SLAM) permettant de générer des cartes 3D tout en les mettant à jour, ou encore d'utiliser directement en entrée de l'algorithme une carte ou un modèle d'un bâtiment et de les corriger au fur et à mesure des déplacements de la plateforme.

Aussi, il serait intéressant de les tester sur des jeux de données urbains et structurés, ce qui impliquerait des nuages de points plus grands et des mouvements relatifs plus élevés entre deux nuages. Cela permettrait d'évaluer la robustesse de nos algorithmes à d'importantes différences de volume et à différents types de capteurs.

INTRODUCTION

For human beings, it is quite easy to locate themselves in their environment. Thanks to the surroundings perceived by our eyes, the human brain is able to decode the information it sees, to immediately understand where it stands. This notion is not directly transferable for artificial entities. Sensors, such as cameras, radars or even LiDARs, have been created to reproduce the notion of human vision for robotic platforms.

Those sensors allow to generate raw data directly related to the surrounding environment, in their local frame. In order to be able to locate itself in a global frame, the robot aim is to estimate its relative motion within the global frame. To do so, primitives, more often geometrical, are used. Hopefully, from one capture to another, the primitives can be identified and matched. This matching process gives the possibility to establish a so-called cost function that will vary in function of the parameters that represent the relative motion between captures. The goal will be to estimate the transformation parameters that will best align the primitives. When the cost function reaches its minimum, the estimated parameters of the transformation are optimal, the primitives are aligned in the best way they could.

This type of pose estimation problem, also called registration problem, was widely studied in the last decades and is still an open issue. Studies differ according to the diversity of sensors or applications, and the expansion of the technology quality makes the subject even more dynamic. Thus, many types of data allowing to answer the localization problem can be found, according to the sensor they are issued from. The type of sensor will also determine the type of transformation linking the data. In this thesis, the choice was made to use only a LiDAR sensor in order to solve the registration problem. A LiDAR generates data in the form of a point cloud. The point clouds can be of two or three dimensions, in our case, we focused on the 3D case. The transformation linking the point clouds are rigid, which means only the relative 3D translation and 3D rotation between point clouds are affected and need to be estimated.

The overall goal of this thesis is to improve registration methods based only on LiDAR data. We focused on how to take advantage of structured environments to provide algorithms that allow a trade off between accuracy, robustness and computation time by reducing the dimensionality of the registration problem.

Context

This thesis was carried out in the VAADER (Video Analysis and Architecture Design for Embedded Resources) team of the IETR (Institut d'Electronique et des Technologies du numéRique) laboratory in Rennes, and was funded by the French Ministry of Higher Education and Research.

Contributions

LOOP'IN, a loop-closure evaluation dataset.

The first contribution of this thesis is a challenging dataset called LOOP'IN. It is a dataset captured thanks to a VLP-16 Puck Hi-Res LiDAR. It is designed to evaluate the ability of point cloud registration algorithms to close loops with real data, in indoor environments. LOOP'IN comes with two long sequences that both include loops. This dataset will be used along with other to assess all the algorithms issued from this thesis.

Multi-resolution registration of 3D point clouds.

In the second contribution, we will present a registration algorithm based on a multi-resolution scheme called Gauss-Newton based Multi-Resolution ICP (GNMR-ICP). To ensure a fast and robust minimization to estimate the transformation parameters, we will minimize the point-to-plane distance using a Gauss-Newton method. To reduce the influence of outliers (caused by the point matching process), we will show the efficiency of the addition of robust M-estimators functions in the minimization process.

Experiments are held on a benchmark dataset to assess the robustness and accuracy of the proposed method. They show that GNMR-ICP performs better than its closed-form equivalent using the small angle approximation, regardless the chosen M-estimator function. They also outline that using more levels of resolution in the multi-resolution process generates more accurate results as well as reducing computation time. We will also see that time reduction applies more obviously in man-made environments.

Plane-based registration of 3D point clouds.

The idea of the third contribution is to exploit the planar structures identified in the second contribution. Thus, an algorithm called New Accurate Plane-based ICP (NAP-ICP) designed to register planes is proposed. First, after planar patches are segmented, plane correspondences need to be established, which is not a trivial task. So we designed a score function based on weighted plane correspondence features to solve this problem. To ensure a robust estimation, a two-step plane-to-plane distance minimization is performed: first, a closed-form minimization is made using a RANdom SAmple Consensus (RANSAC) process to initialize the estimation and to identify the inliers. Then an iterative Gauss-Newton minimization on the plane-to-plane distance is performed to refine the estimation. An additional point-to-plane registration is performed at the end of the process to make the estimation as precise as possible.

Experiments have shown the importance and the impact of the RANSAC step and the point-to-plane additional registration step on NAP-ICP's process accuracy. The accuracy and robustness of the proposed method have then been assessed and compared to state of the art algorithms. NAP-ICP is able to outperform the state of the art algorithms in this context. Finally, thanks to experiments on LOOP'IN dataset, we have demonstrated that NAP-ICP is able to close loops and also to recover from errors made in the estimation during the registration of long sequences.

Learning-based plane matching for plane-to-plane registration.

NAP-ICP approach enables to find plane matches using a score function based on plane correspondence features. However the weights of the score function have been chosen empirically and we need to make this part more robust. We will show how this weighting function can be replaced by a learning-based method designed to classify whether planes are true or wrong matches. This classifier is based on a Random Forest classification trained thanks to the identified features. The data used for training comes from a dataset including the ground truth of the sensor pose, allowing to easily identify the plane matches, and to label the data.

The experiment have shown how this new method Plane-based Accurate Registration ICP (PAR-ICP) is able to successfully register the indoor sequences of the benchmark dataset. The algorithm also proves that it is able to close loops on long sequences and that it is able to provide geometrically consistent incremental maps.

Outline of this thesis

The manuscript is structured as follows.

In Chapter 1, the background of this thesis is presented. A description of the Light Detection And Ranging (LiDAR) and its applications are given. Then, the data representation is presented and the 3D primitives used in the manuscript are detailed. Finally, the theoretical foundations needed to express the pose estimation problem we are aiming to solve in this manuscript are summarized.

In Chapter 2, a global state of the art of the registration of 3D point clouds is given. A focus on the well-known Iterative Closest Point (ICP) algorithm is made. A section is dedicated to Simultaneous Localization And Mapping (SLAM) applications of 3D point clouds.

Chapter 3 presents an overview of the datasets created to evaluate point cloud registration algorithms properties. A section details the two datasets used all along this manuscript, including the one designed during this thesis, LOOP'IN.

In Chapter 4, a robust multi-resolution registration algorithm, called GNMR-ICP, is presented. The algorithm is fully detailed and its robustness, accuracy and computation time are evaluated thanks to a series of experiments.

Chapter 5 depicts a plane-based algorithm, NAP-ICP, designed to reduce the dimensionality of the registration problem while keeping the estimation accurate. The impact of each step of the algorithm on accuracy is presented in addition to a comparison with state of the art algorithms.

In Chapter 6, a variation of the plane-based registration algorithm presented in chapter 4 is given, PAR-ICP. Its focus is on the plane matching technique. A method based on learning to build plane-to-plane correspondences is presented. Once more, experiments evaluating accuracy and robustness are held.

Finally the conclusions and perspectives are given in a last chapter.

Chapter 1

Background

The problem addressed in this thesis is formulated as an optimization problem on the data provided by a Light Detection And Ranging (LiDAR). It is then necessary to introduce the type of data we will use as well as the optimization method we intend to use. To that end, a first section introduces the description and applications of a LiDAR, the primary sensor considered in this work. Then, the representations of the data provided by the LiDAR sensor are presented. After, we consider the 3D primitives used in our contributions. Next, the transformation and parameters estimation notions, describing the 3D registration problem, are introduced. Finally, as we are dealing with data acquired from a sensor, we need to address the outlier issue.

Please note that the notations used in the manuscript are summed up in Appendix A.

1.1 LiDAR sensor

In robotics, navigation applications can be handled by different types of sensors. They all come with advantages and disadvantages. For instance, cameras are widely used because of the significant amount of information (for instance, color) that can be generated through images, moreover, it is a rather cheap technology. However, cameras are very sensitive to sudden light changes or direct sunlight. We can also find some types of cameras that can directly provide depth information through structured infrared light such as a Kinect v1. In sunlight, it is not possible to use infrared information as it is drowned in the one caused by the sun. Thus, using a Kinect v1 outdoor would not be possible. LiDAR sensors are not sensitive to ambient lightning. However, a LiDAR is more expensive than a camera and it can only process spatial information. LiDAR sensors comes with a range up to 100m with 2cm accuracy for the Velodyne sensors, whereas it is only limited to 5m for a Kinect. They may also provide a 360° field of view which is convenient to locate a system in a structured environment.

1.1.1 Description

A LiDAR is a sensor allowing to compute distances. Its functioning is very similar to the one of a radar, but instead of radio waves, light is used as can be seen in Fig. 1.1. A laser beam is emitted and its reflection on the target comes back to the transmitter. The reflected beam is analyzed and, thanks to light properties, it gives the target location.

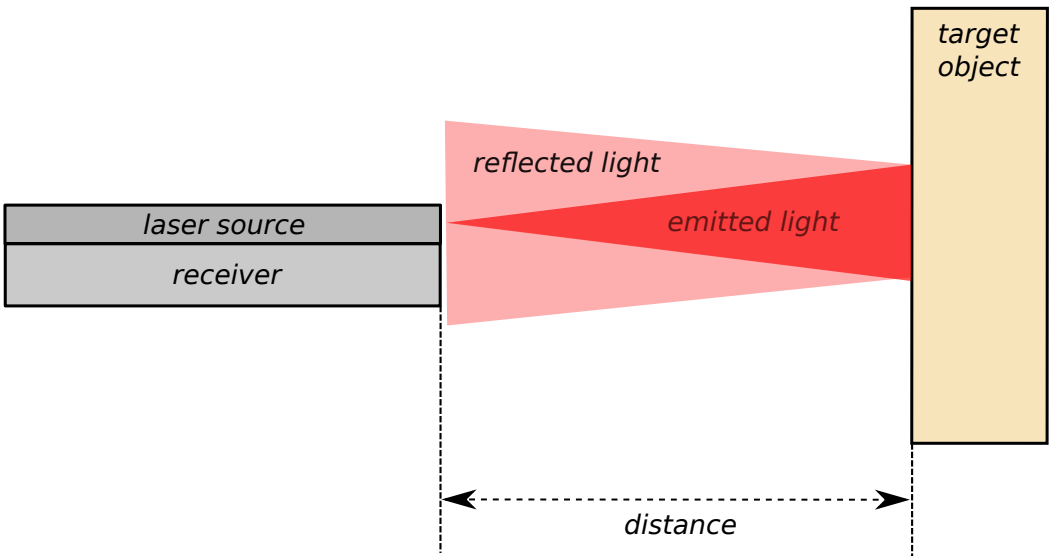


Figure 1.1 – Principle of a LiDAR sensor.

For georeferencing purpose, a LiDAR can be combined with other types of sensors such as Global Positionning System (GPS) or Inertial Measurement Unit (IMU). It can also be coupled with a camera to provide a synchronous acquisition of both sensors.

Three types of LiDARs can be found, regarding the number of dimensions they exploit:

— in 1D: only one fixed beam is necessary, it measures the distance to the object in front of the LiDAR;

— in 2D: only one beam is required, it performs a rotation which delivers information on two axes;

— in 3D: the principle of rotation is the same as in 2D, but the sensor has multiple beams. Each beam has its own angle. By rotating, the sensor will output a set of 3D points as depicted in Figure 1.2.

The generated data is generally a set of 3D points, also called a 3D point cloud. It may also come with an intensity information for each point: it is measured thanks to the reflectivity of the object hit by the laser. Its strength varies with the composition of the surface of the object reflecting the signal.

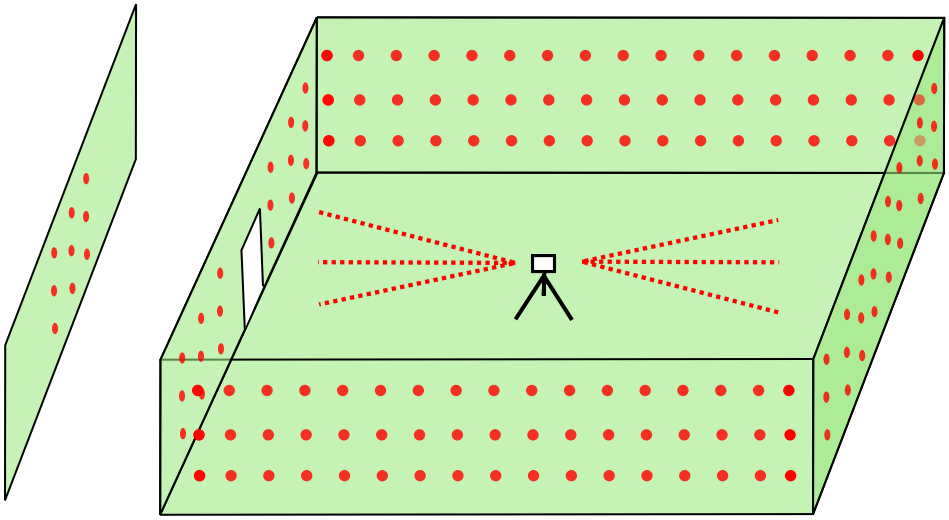
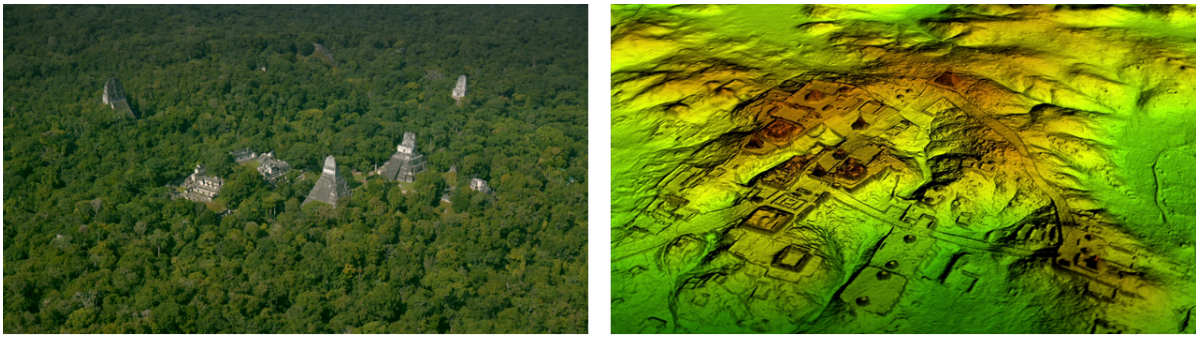


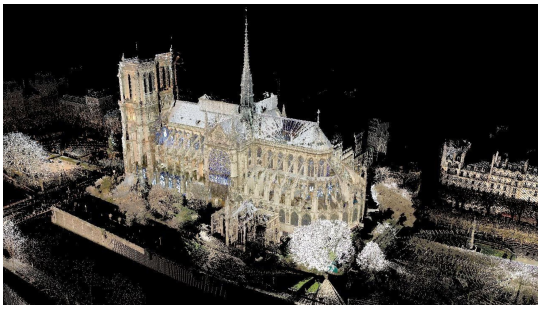
Figure 1.2 – Depiction of a 3D LiDAR capturing a point cloud in a room. In this case the LiDAR has three beams and rotates on itself. A 3D point cloud is generated (red points). A door is open and points are captured behind it.

1.1.2 Applications

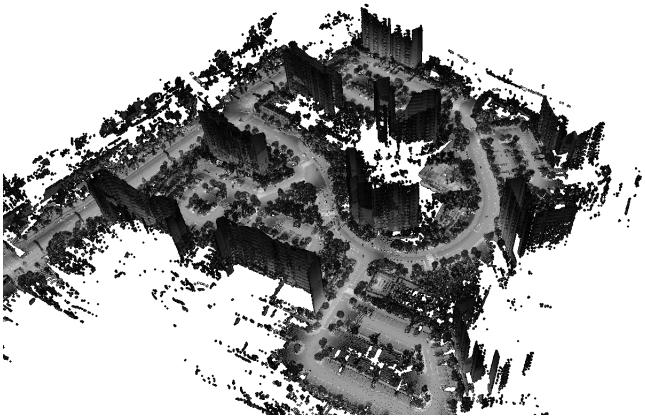
The first LiDAR sensors were used in the early 1960's in aerospace applications. In the beginning, the applications were airborne, for topographic mapping of forests, oceans, etc. The development of efficient GPS and satellite communications allowed to democratize the LiDAR sensor usage in meteorology, atmospheric research or shoreline monitoring, to finally come to city mapping (Fig. 1.3 (c)). In order to complete those acquisitions, LiDAR sensors for terrestrial use were created. They are used in civil engineering to model buildings in 3D. A noticeable application is the model of the Cathedral of Notre-Dame which was scanned before its burning in 2019, and which is helping for its reconstruction (Fig. 1.3 (b)). Airborne acquisitions are, nowadays, also used in archaeology. In 2020, the vestige of a Mayan city were found below a dense jungle in Guatemala (Fig. 1.3 (a)), thanks to LiDAR technology. Besides from 3D model reconstruction, LiDAR sensors are used in the autonomous vehicle field (Fig. 1.3 (d)(e)(f)) in order to localize the system in its environment. Nowadays LiDAR sensors are much more affordable, which makes the possibility to be used even more in research and industrial applications.



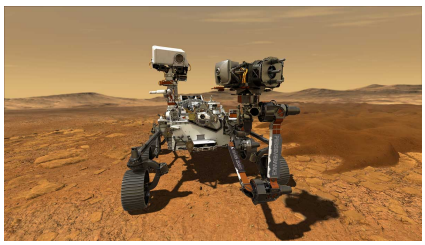
(a) Mayan megalopolis below Guatemalan jungle. Credits: National geographic



(b) LiDAR capture of Notre-Dame cathedral. Credits: l'Express



(c) Urban reconstruction. Credits: KAIST University



(d) Perseverance rover. Credits: NASA

(e) Autonomous vehicle. Credits: Keolis Rennes

(f) Autonomous vacuum cleaner. Credits: Roborock

Figure 1.3 – LiDAR applications in 3D modeling and autonomous driving.

LiDAR sensors come with a few drawbacks:

— the design of LiDAR sensors makes the density of points highly heterogeneous. The density of points is much higher close to the sensor, while it is scattered far from it as in Fig. 1.4);

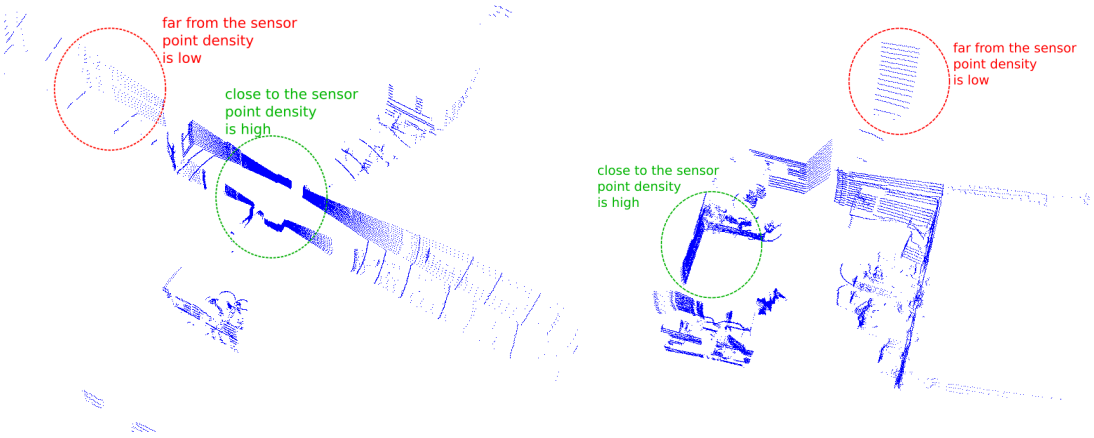


Figure 1.4 – Depiction of the point density issue.

— a precise 3D model implies a large number of input data which increases processing time;

— due to the functioning of a LiDAR sensor, since the laser is reflected by the first opaque surface it runs into, some part of the environment can be occluded. And, if its runs into a reflective surface a ghost effect may appear as shown in Figure 1.5.

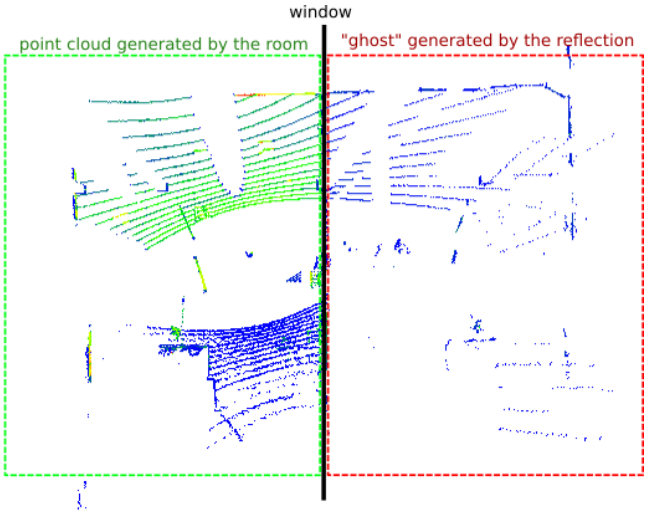


Figure 1.5 – Top view of a point cloud depicting the ghost effect. The reflection of the room in the window creates "ghost" points that do not physically exist, a "ghost" room appears on the scan.

1.2 Data representation

As stated in the introduction of this section, the data generated by a LiDAR sensor comes in the form of a point cloud. We need to further describe how such data can be represented and ordered.

1.2.1 Point cloud

A point cloud is a set of points in space. It can represent whether an object or an environment. In the case of 3D point clouds, each point is represented by its coordinates X, Y, Z. A point cloud can contain additional information such as light intensity, color or normal, at each point.

It is important to note that points stored in the point cloud have no spatial relationship with each other, in contrast with a mesh, where connectedness relationships are given by the edges. However, the contributions presented in this manuscript sometimes need to use a nearest neighbor search to match points from one point cloud to another. The nearest neighbor search problem consists in finding the point that is closest to a given point in a set of data. In our case, closeness is represented by the Euclidean distance between the points. Let us consider two point clouds ${}^1\mathbf{P}$ and ${}^2\mathbf{P}$ composed of, respectively, n and m points. If this search is made linearly, in an unordered array, it has a complexity of $O(nm)$. To tackle this problem, and to handle point clouds more wisely, data structures were created such as kD-trees and octrees to reduce search complexity.

1.2.2 kD-tree structure

A kD-tree is a space-partitioning data structure. It is used in order to accelerate the nearest neighbor search. The building phase of a kD-tree is of complexity $O(m \log m)$ with m the number of points in point cloud ${}^2\mathbf{P}$, and the search complexity is $O(n \log m)$, with n the number of points in ${}^1\mathbf{P}$.

Description

A kD-tree is a special case of a Binary Space Partitionning (BSP) tree. A BSP tree splits k-dimensional space volume in two sub-volumes by a hyperplane of the space, and then iterates on the two resulting volumes. Thus, the two sub-volumes are the children of the node that

represents the whole volume. The hyperplanes are chosen such as their normal is one of the axis of the coordinate system.

Construction

Many ways to build and handle a kD-tree can be found in the literature ([Arya et al. 1993] [Nüchter et al. 2005] [Nuchter et al. 2007]). In the classical method, with a 3D example, the normal of the plane splitting the root is aligned with the x-axis. Its children have their plane normal aligned with the y-axis, and its grand-children axes are aligned with the z-axis. The following splitting plane normal is aligned with the x-axis and so on.

To have a balanced tree, the chosen point to perform the split is located at the median coordinate of the considered direction. An example of a kD-tree is given in figure 1.6.

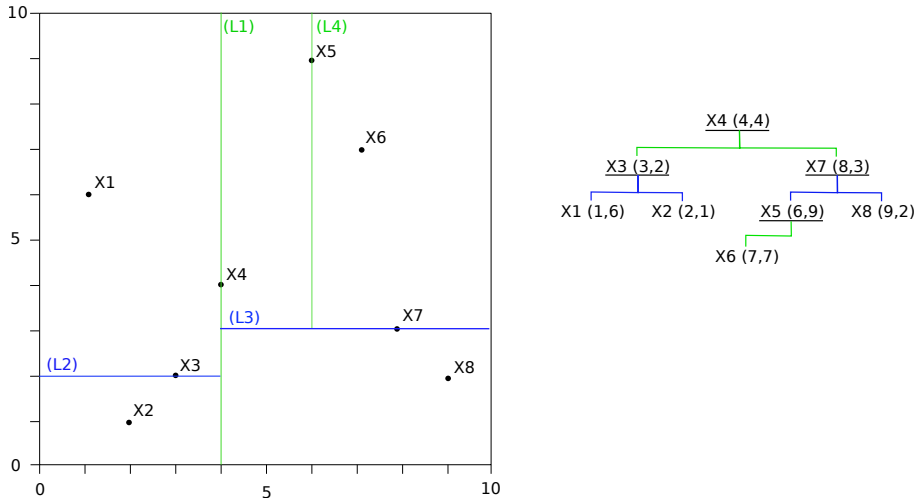


Figure 1.6 – *Left:* a 2D point set with the kD-tree splits. *Right:* The corresponding tree. *Note: a 2D example is given to ease reading.*

1.2.3 Octree structure

An octree is a tree-based data structure designed to partition 3D space. Similarly to the kD-tree, it allows to reduce search complexity. The nearest neighbor search of $O(nm)$ becomes $O(n \log m)$ after a building phase of complexity $O(m \log m)$ (with m the number of points in ${}^2\mathbf{P}$ and n in ${}^1\mathbf{P}$). As its name indicates, each node of the octree can count up to eight children, which will divide the volume in eight sub-volumes. In other words, with an octree, the space is represented by a cube which is divided in eight smaller cubes until the smallest voxel reaches the smallest resolution chosen. For each subdivision, the stored node is, most of the

time, a 3D point which is the center of this volume. An example of an octree representation is given in figure 1.7. In [Hornung et al. 2013] an open source framework is provided, designed for 3D mapping in autonomous navigation applications.

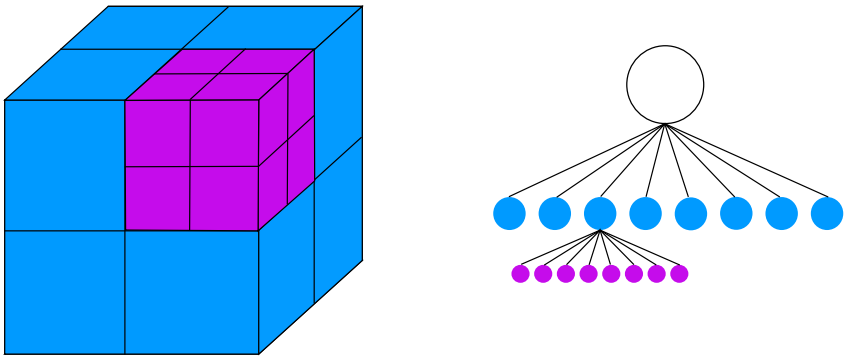


Figure 1.7 – *Left:* an octree with a few subdivisions. *Right:* the corresponding tree.

1.3 3D primitives

In order to describe the problematics presented in this thesis, the 3D primitives and their related notations are outlined in this section. Regarding the data provided by a LiDAR sensor, we have to describe 3D points notations. Nevertheless, in order to reduce the complexity of the problem and to exploit the structure of the scenes, it will be necessary to handle other types of geometrical entities. Thus in this section, we first provide the notation for 3D points. Then, the estimation of the local 3D normals used in our work is presented. Finally, the representation of the 3D planes is described.

1.3.1 3D point

In computer vision, homogeneous coordinates (or projective coordinates), are a system of coordinates used in projective geometry, similarly as Cartesian coordinates are used in Euclidean geometry. Homogeneous coordinates give the possibility to represent affine and projective transformations by matrices, which are used all along this manuscript and described in more detail in section 1.4.1. Given a 3D point $\overline{\mathbf{p}} = (X, Y, Z)^\top$ in Euclidean space, its homogeneous coordinates are defined as:

$$\mathbf{p} = (W\overline{\mathbf{p}}, W)^\top = (WX, WY, WZ, W)^\top \tag{1.1}$$

with W a scale parameter as the extra-dimension.

1.3.2 3D normal estimation

As we will explain, the normal at a point contains valuable information in order to register two point clouds. Indeed, the computation of distances and plane parameters rely on normal estimation. The normal vector at a point $\overline{\mathbf{p}}$ is denoted $\mathbf{n} = (N_x, N_y, N_z)^\top$ (Figure 1.8). In order to estimate the normal vector related to the 3D points of the point clouds, we chose to use the so-called Principal Component Analysis (PCA)[Hoppe et al. 1992]. It is a method

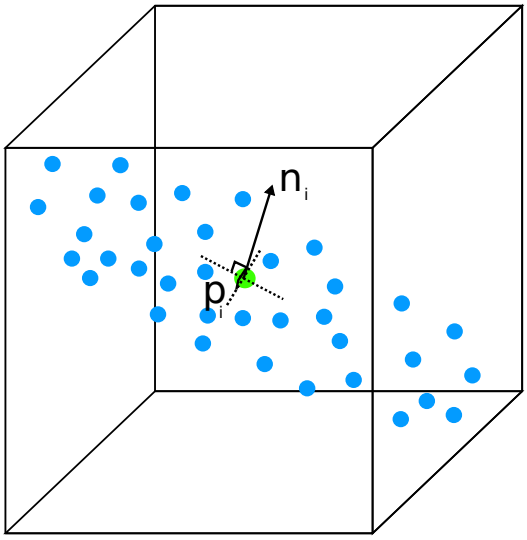


Figure 1.8 – Normal vector $\mathbf{n}_i$ at point $\mathbf{p}_i$

allowing to reduce the dimensionality of large data sets while preserving most of the information it contains. The principal components are the eigenvectors of the data's covariance matrix, obtained following these steps:

— the first step is to center the dataset by subtracting the mean for each column;
— then the covariance matrix of the centered data is computed;
— an eigen decomposition is performed on the covariance matrix, the bigger the eigenvalue the more variance is contained into the corresponding eigenvector.

In the case of computing the normal of a set of points, the chosen vector is the one corresponding to the smallest eigenvalue. The method used in this work is given in the following insert **Normal estimation by PCA by [Rusu 2010]**. Its implementation is the one found in the Point Cloud Library (PCL) [1] [Rusu et al. 2011].

1. https://pointclouds.org/

Normal estimation by PCA by [Rusu 2010]

Determining the normal of a surface at a point in the point cloud comes to estimate the normal of a plane tangent to the surface. For each point $\mathbf{p}_i$ in a dataset the covariance matrix is defined as:

$$\mathbf{C}(\mathbf{p}_i) = \frac{1}{k}\sum_{l=1}^{k}(\mathbf{p}_l - \mathbf{c})(\mathbf{p}_l - \mathbf{c})^\top \tag{1.2}$$

$$\mathbf{C}(\mathbf{p}_i) \cdot \mathbf{v}_j = \lambda_j \cdot \mathbf{v}_j \ , \ j \in \{0, 1, 2\} \tag{1.3}$$

where k is the number of points in the neighborhood of $\mathbf{p}_i$ and $\mathbf{c}$ the centroid of the k nearest neighbors. λ_j is the j-th eigenvalue of the covariance matrix and $\mathbf{v}_j$ the j-th eigenvector. If $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$ the eigenvector $\mathbf{v}_0$ corresponding to the smallest eigenvalue λ_0 is the approximation of $\mathbf{n}$ or -$\mathbf{n}$ the normal vector.

A robust version of the normal estimation based on the PCA algorithm is proposed in [Sanchez et al. 2020] where M-estimators are used.

In the literature many other techniques aiming to estimate normal vectors in point clouds can be found, such as 2-jets algorithm [Cazals et al. 2005], Voronoï Covariance Measure (VCM) [Mérigot et al. 2011], Pair Consistency Voting (PCV) [Zheng et al. 2018]. More recently, in [Ben-Shabat et al. 2019] a method using neural networks called Nesti-Net has been designed to estimate normal vectors from unorganized point clouds.

1.3.3 3D Plane

Planes are elements that can be found frequently in structured environments. Therefore we will use them in Chapters 5 and 6 of this manuscript. As planes can be represented in many ways, it is important to clarify the notation further used.

We chose to use the $(\rho, \mathbf{n})$ representation which is non ambiguous. Let:

$$\mathbf{n}^\top \mathbf{X} = \rho \tag{1.4}$$

be the equation of a plane where $\mathbf{X}$ is a point belonging to the plane, ρ is the distance of the plane from the origin and $\mathbf{n}$ its normal. Figure 1.9 gives a representation of it. ρ and $\mathbf{n}$

respond to the following conditions:

$$\rho = \|\mathbf{OH}\| \tag{1.5}$$

$$\mathbf{OH} \cdot \mathbf{n} > 0 \quad \text{and} \quad \|\mathbf{n}\| = 1 \tag{1.6}$$

where H is the projection of the origin on the plane $\mathbf{\Pi}$.

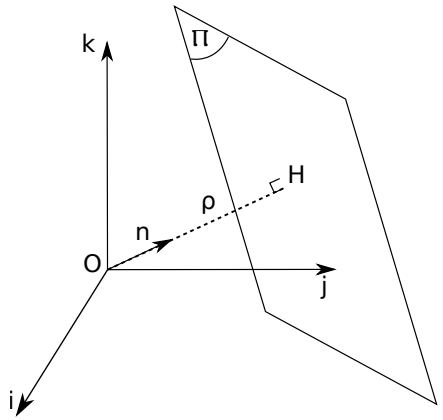


Figure 1.9 – Plane $\mathbf{\Pi}$ representation

1.4 Theoretical foundations of the registration problem

This section presents the theoretical foundations needed to express the registration problem and the parameters optimization techniques used in this thesis.

1.4.1 3D geometry

The considered point clouds are studied in 3 dimensions. Therefore, 3D frames and transformations will be presented thereafter.

Change of frame and 3D rigid transformation

Let $\mathcal{F}_a$ and $\mathcal{F}_b$ be two frames in 3D Euclidean space. Let ${}^a\overline{\mathbf{p}} = ({}^aX, {}^aY, {}^aZ)^\top$ be the cartesian coordinates of a point in $\mathcal{F}_a$ and ${}^b\overline{\mathbf{p}} = ({}^bX, {}^bY, {}^bZ)^\top$ the cartesian coordinates of the point in $\mathcal{F}_b$. The two point vectors are linked by a rigid transformation in $\mathbb{SE}(3)$, defined by the 3×3

rotation matrix ${}^a\mathbf{R}_b$ and the 3×1 translation vector ${}^a\mathbf{t}_b$ such that:

$$ {}^a\overline{\mathbf{p}} = {}^a\mathbf{R}_b {}^b\overline{\mathbf{p}} + {}^a\mathbf{t}_b \tag{1.7} $$

The rotation matrix $\mathbf{R}$ *is orthogonal (*$\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = \mathbf{I}$*) and has determinant* 1*.*

With ${}^a\mathbf{p}$ and ${}^b\mathbf{p}$ being, respectively, the homogeneous coordinates of ${}^a\overline{\mathbf{p}}$ and ${}^b\overline{\mathbf{p}}$, and thanks to projective geometry, the relationship between the points is homogenized such that:

$$ {}^a\mathbf{p} = {}^a\mathbf{T}_b {}^b\mathbf{p} \tag{1.8} $$

with:

$$ {}^a\mathbf{T}_b = \begin{bmatrix} {}^a\mathbf{R}_b & {}^a\mathbf{t}_b \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{1.9} $$

Transformation composition

Let us consider a third frame $\mathcal{F}_c$, in addition to the frames introduced in the previous section. The change of frame, from $\mathcal{F}_a$ to $\mathcal{F}_c$, can be expressed as the composition of frame changes such that:

$$ {}^a\mathbf{T}_c = {}^a\mathbf{T}_b {}^b\mathbf{T}_c \tag{1.10} $$

Minimal representation of the 3D rigid transformation

The homogeneous transformation matrix $\mathbf{T}$ allows to easily change Euclidean frame, however this representation is not minimal. A rotation matrix can be defined with only three independent variables [Hartley et al. 2004]. The transformation $\mathbf{T} \in \mathbb{SE}(3)$ can be described with a 6 dimension vector $\mathbf{q} = (\mathbf{t},\ \theta\mathbf{u}) \in \mathfrak{se}(3)$ where $\theta\mathbf{u}$ is the axis-angle representation for the parametrization of the rotation matrix $\mathbf{R}$, with $\mathbf{u}$ being a unit vector indicating the direction of the axis of rotation, and θ the magnitude of the angle around this axis. The Rodrigues formula allows to switch from $\theta\mathbf{u}$ to $\mathbf{R}$:

$$ \mathbf{R} = \cos\theta\mathbf{I}_3 + (1 - \cos\theta)\mathbf{u}\mathbf{u}^\top + \sin\theta\,[\mathbf{u}]_\times \tag{1.11} $$

The rotation matrix $\mathbf{R} = \exp([\theta\mathbf{u}]_\times)$ is the exponential matrix of $[\theta\mathbf{u}]_\times$. Note that for a vector $\mathbf{v} = (a, b, c)$, $[\mathbf{v}]_\times$ denotes the skew matrix given by:

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

The switch from vector $\mathbf{q} = (\mathbf{t}, \theta\mathbf{u})$ to $\mathbf{T}$ is given by:

$$\mathbf{T} = \exp([\mathbf{q}]_\times) = \begin{cases} \begin{bmatrix} \exp([\theta\mathbf{u}]_\times) & \frac{(\mathbf{I}_3 - \exp([\theta\mathbf{u}]_\times))[\theta\mathbf{u}]_\times \mathbf{t} + \theta\mathbf{u}\theta\mathbf{u}^\top \mathbf{t}}{\|\theta\mathbf{u}\|} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} & \text{if} \quad \theta \neq 0 \\ \begin{bmatrix} \mathbf{I}_3 & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} & \text{otherwise.} \end{cases} \tag{1.12}$$

The other way around, from $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$, θ is computed thanks to:

$$\cos\theta = \frac{1}{2}(trace(\mathbf{R}) - 1) \tag{1.13}$$

and by setting $\theta > 0$, $\mathbf{u}$ can be uniquely determined if $\sin\theta \neq 0$ by:

$$\sin\theta\,[\mathbf{u}]_\times = \frac{1}{2}(\mathbf{R} - \mathbf{R}^\top) \tag{1.14}$$

If $\sin\theta = 0$, the rotation axis $\mathbf{u}$ is the eigenvector of the rotation matrix $\mathbf{R}$ associated to the eigenvalue $\lambda = 1$.

1.4.2 Least squares optimization methods

In this thesis, one main goal is to estimate transformation parameters. To do so, a criterion (most of the time, a residual error) is minimized in order to find the optimal parameters of a vector. In this section are described the method used to estimate such parameters in general cases, first with linear systems, then with non-linear systems of equations.

1.4.2.1 Linear least squares optimization

When the observations represent linear combinations of the parameter vector $\mathbf{x} = (x_1, \ldots, x_n)$ of size n to estimate, such as:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \tag{1.15}$$

the problem can be rewritten:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.16}$$

with $\mathbf{A}$ an $n \times m$ matrix, $\mathbf{b}$ a vector of size m. If $n > m$, there are more equations than unknowns. In general, there is no solution to this system. If a solution does not exist, it is still possible to find a vector $\mathbf{x}$ that is close to provide a solution to the system, minimizing:

$$E(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{1.17}$$

In other words, a vector $\mathbf{x}$ that is the least squares solution to the system can be computed. The minimization of equation (1.17) is written:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{1.18}$$

with $\hat{\mathbf{x}}$ the optimal parameters vector. The closed-form solution of this convex problem can be obtained using:

$$\hat{\mathbf{x}} = (\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{A}\mathbf{b} = \mathbf{A}^+\mathbf{b} \tag{1.19}$$

where $\mathbf{A}^+$ represents the Moore-Penrose pseudo-inverse of $\mathbf{A}$.

The pseudo-inverse can then be computed using Cholesky, QR or Singular Value Decomposition (SVD) decompositions.

1.4.2.2 Non-Linear least squares optimization

In the estimation problem exhibited in this thesis, most of the time, the problem is non-linear. Indeed, the parameters to estimate are composed of rotations, and, according to its parametrization, the problem can be either approximated with a linear form or solved in its non-linear form. Non-linear optimization is quite difficult as there is generally no direct solution

nor it is not possible to perform an exhaustive search of the set of parameters. It is necessary to perform iterations by doing local minimization.

We are looking for the solution of a system of equations such as:

$$\mathbf{e}(\mathbf{x}) = \mathbf{0} \Leftrightarrow \begin{cases} e_1(x_1, x_2, \ldots, x_n) = 0 \\ \ldots \\ e_m(x_1, x_2, \ldots, x_n) = 0 \end{cases} \tag{1.20}$$

with:

— n unknowns: $x_1, \ldots, x_n$;

— $\mathbf{e} = (e_1, \ldots, e_m)^\top$ and $e_i(.)$ a function (potentially non-linear).

If $m > n$ there is no solution to this problem, and we chose to minimize the cost function $E(\mathbf{x}) = \|\mathbf{e}(\mathbf{x})\|^2 = \sum_{i=1}^{m} e_i(\mathbf{x})$.

Two very close approaches have proved their efficiency to solve non-linear least-squares problems: Gauss-Newton and Levenberg-Marquardt.

Gauss-Newton optimization

The Gauss-Newton minimization method is an iterative method, known to be efficient when the initialization is close to the minimum of the objective function. Minimizing the problem formulated in equation (1.20) consists in minimizing the cost function:

$$E(\mathbf{x}) = \|\mathbf{e}(\mathbf{x})\|^2 \tag{1.21}$$

The solution consists in linearizing $\mathbf{e}(\mathbf{x})$. The cost function can be locally approximated using the first order Taylor expansion, giving:

$$\mathbf{e}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{e}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x} \tag{1.22}$$

where $\mathbf{J}(\mathbf{x})$ is the Jacobian of $\mathbf{e}(\mathbf{x})$ in $\mathbf{x}$. With the Gauss-Newton method, the solution consists in minimizing $E(\mathbf{x} + \delta\mathbf{x})$ with:

$$E(\mathbf{x} + \delta\mathbf{x}) = \|\mathbf{e}(\mathbf{x} + \delta\mathbf{x})\|^2 \approx \|\mathbf{e}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\delta\mathbf{x}\|^2 \tag{1.23}$$

The minimization problem can be solved by an Iterative Least Squares (ILS) approach which

gives:

$$\delta\mathbf{x} = -\lambda\mathbf{J}(\mathbf{x})^{+}\mathbf{e}(\mathbf{x}) \tag{1.24}$$

where λ is a coefficient in $]0, 1]$ and $\mathbf{J}(\mathbf{x})^{+}$ is the pseudo-inverse of the Jacobian $\mathbf{J}(\mathbf{x})$.

Levenberg-Marquardt optimization

The Levenberg-Marquardt approach is a variation of Gauss-Newton method. Its formulation allows to change the behavior of the algorithm whether the estimation is far or near the optimum. The normal equation (1.24) is replaced by the augmented normal equation:

$$\delta\mathbf{x} = -\lambda(\mathbf{J}(\mathbf{x})^{\top}\mathbf{J}(\mathbf{x}) + \mu\mathbf{I})^{-1}\mathbf{J}(\mathbf{x})^{\top}\mathbf{e}(\mathbf{x}) \tag{1.25}$$

The damping factor μ is adjusted at each iteration. If the reduction of the residual is large, a smaller value can be used, whereas if the reduction is too small, a larger μ is used.

1.4.2.3 Robust estimation

With the previously presented methods, the assumption is made that the measurements are correct. Meaning that all considered data are inliers, which is mostly unlikely in real conditions. In practice, some erroneous input data may appear and they must not be taken into account in the optimization process as their presence may alter the quality of the estimation. Several techniques were proposed to identify and remove those erroneous data, also called outliers.

RANSAC

The RANdom SAmple Consensus (RANSAC) algorithm [Fischler et al. 1981] was designed to solve pose estimation problems of 3D point sets. More generally, it can be used to estimate parameters of models from a set of data containing outliers. The principle is to find, among all observations, a minimal subset containing only inliers that allow to identify the outliers generated by the parametrization of this subset. Each step is detailed in the insert below called **RANSAC algorithm by [Fischler et al. 1981]**.

RANSAC algorithm by [Fischler et al. 1981]

1. a random draw of minimal subsets (samples) is performed;
2. for each sample, the considered equation is solved using a least squares optimization method, to find an estimation of $\hat{\mathbf{x}}$;
3. the quality of a sample is assessed by considering the inliers it generates (observations with a residual error smaller than a chosen threshold);
4. the best sample is retrieved. It is the sample which issues the most inliers.
5. The estimation of $\hat{\mathbf{x}}$ is computed again, integrating all (and only) inliers.

If we have some knowledge about the reliability of the data, the number of required iterations can be computed theoretically. Let p be the probability for the RANSAC algorithm to select only inliers among the input data when choosing the minimal sample of size n. Let w be the probability to select an inlier each time an input is selected. Thus, the number of iterations k to obtain a sample made of only inliers can be computed thanks to:

$$k = \frac{log(1-p)}{log(1-w^n)} \tag{1.26}$$

RANSAC algorithm is at the root of many robust optimization techniques such as PROgressive SAmple Consensus (PROSAC) [Chum et al. 2005], Maximum Likelihood Estimation SAmple and Consensus (MLESAC) [Torr et al. 2000], M-estimator SAmple and Consensus (MSAC) [Torr et al. 2000].

M-estimators

An other method to reduce the impact of outliers on the estimation of the parameters is called the M-estimators. Unlike RANSAC, M-estimators will not partition the data for the estimation process, they will take into account all of them, and weight them. M-estimators belong to the so-called Iteratively Re-weighted Least Squares (IRLS) minimization methods. It consists in weighting the observations regarding the confidence in the data. The minimization problem can be rewritten such that:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \ \rho(\mathbf{e}(\mathbf{x})) \tag{1.27}$$

with $\rho(.)$ a weighting function. The non-linear problem introduced in equation (1.20) can be replaced with:

$$\mathbf{e}_\rho(\mathbf{x}) = \mathbf{D}(\mathbf{e}(\mathbf{x})) \tag{1.28}$$

and then the Gauss-Newton solution is:

$$\delta\mathbf{x} = -\lambda(\mathbf{DJ}(\mathbf{x}))^+\mathbf{De}(\mathbf{x}) \tag{1.29}$$

where $\mathbf{D}$ is a $N \times N$ diagonal matrix containing the weights that reflect the confidence in the data.

$$\mathbf{D} = \begin{pmatrix} w_1 & \dots & 0 \\ & \ddots & \\ 0 & \dots & w_n \end{pmatrix} \tag{1.30}$$

The weights w_i are values in $[0, 1]$. A value near zero means that the data is an outlier. With $\mathbf{e} = (e_1, \dots, e_n)$ the weights w_i are computed such that:

$$w_i = \frac{\psi(e_i)}{e_i} \tag{1.31}$$

where $\psi(x)$ is an influence function computed by:

$$\psi(e_i) = \frac{\partial\rho(e_i)}{\partial e_i} \tag{1.32}$$

The robust functions have to be symmetric, non-negative and monotonically-increasing. Several M-estimators can be found in the literature such as Huber's [Huber et al. 1981], Tukey's [Beaton et al. 1974], or Cauchy's.

In this thesis, both RANSAC and M-estimators are used to increase the robustness of the pose estimation problem. M-estimators are preferred when input data is numerous as they rely on a statistical aspect. The design of RANSAC algorithm, more suited to smaller input dataset, is used when fewer noisy data is available.

1.5 Conclusion

This chapter aims to provide a theoretical background to the reader, introducing notions used all along this manuscript. The sensor, the representation of its data and the 3D primitives are presented. The transformations linking the elements of a scene are introduced, as well as

optimization techniques to estimate parameters, and the associated robust methods.

The following chapter is dedicated to the presentation of the state of the art of point cloud registration.

Chapter 2

State of the art

2.1 Introduction

This chapter aims to present a global state of the art for this thesis, related to the registration of point clouds. A first section focuses on the Iterative Closest Point (ICP) algorithm and its main variations. Then, other algorithms dedicated to point cloud registration are presented. In the last section, we wanted to outline the state of the art of a point cloud registration application, which is LiDAR-based Simultaneous Localization And Mapping (SLAM).

2.2 Point cloud registration

In this thesis, we address the problem of 3D point clouds registration. It comes to looking for the rigid transformation that best aligns two point clouds. The point clouds will be denoted as ${}^{s}\mathbf{P}$ for the source point cloud and ${}^{t}\mathbf{P}$ for its target point cloud.

2.2.1 Iterative Closest Point

The most well-known method to register point clouds is the so-called ICP algorithm. It was brought by [Besl et al. 1992] in order to register 3D shapes. The aim is to minimize the sum of the point-to-point distance d_i with:

$$d_i = \| {}^{t}\mathbf{T}_s \, {}^{s}\mathbf{p}_i - {}^{t}\mathbf{p}_i \| \tag{2.1}$$

where ${}^{s}\mathbf{p}_i$ and ${}^{t}\mathbf{p}_i$ are respectively the source and the target points, and ${}^{t}\mathbf{T}_s$ the rigid transformation that links ${}^{s}\mathbf{P}$ to ${}^{t}\mathbf{P}$. Denoting $\mathbf{q} = ({}^{t}\mathbf{t}_s, \theta\mathbf{u})^T$, the minimal representation of $\mathbf{T}$, where θ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^{t}\mathbf{R}_s$, the point-to-point error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{n} d_i \tag{2.2}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \mathbf{p}(\mathbf{q}) - \mathbf{p} \tag{2.3}$$

where $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s\, {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$ and $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$.

The point-to-point distance is depicted in Figure 2.1.

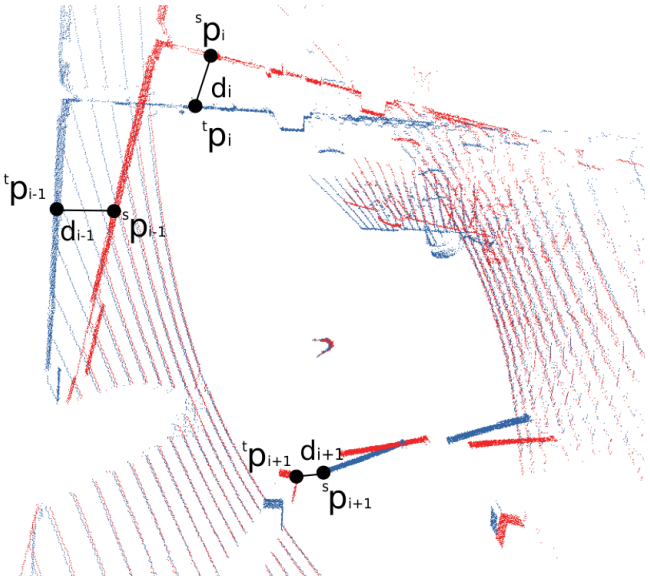


Figure 2.1 – Point-to-plane distance d_i between two point clouds. In blue, ${}^t\mathbf{P}$ the target point cloud. In red, ${}^s\mathbf{P}$ the source point cloud to register.

The method of the ICP algorithm is described in the following insert **Iterative Closest Point by [Besl et al. 1992]**.

Iterative Closest Point by [Besl et al. 1992]

1. For each point in the source, we look for its closest point in the target according to the Euclidean distance;
2. then we look for the transformation ${}^t\mathbf{T}_s$ that minimizes the sum of the point-to-point distances d_i;
3. the transformation is applied to the source;
4. steps 1 to 3 are iterated.

This method is efficient, and easy to implement, but comes with some drawbacks. One of

these problems is generated by the specificity of the Light Detection And Ranging (LiDAR) functioning, it is the sparsity of the point clouds as presented in section 1.1.2. This sparsity implies that it is highly unlikely that a point from the source will have its exact correspondent in the target point cloud. Moreover, the authors stress the fact that the nearest neighbor search is a time consuming task. Let us remind that point clouds comes as a list of unordered points, making point matching even more time consuming. The authors propose to use k-D trees in order to make this step faster. Not knowing point matches and relying on closest pairs of points also implies that the initialization has to be close to the solution. This method is also very sensitive to local minima.

In [Rusinkiewicz et al. 2001] and [Pomerleau et al. 2015] the variations and applications of the ICP are studied. In the following sections the principal variations aiming to improve the ICP algorithm are presented.

Cost function variations

As stated in the previous section, corresponding 3D shapes might be sampled differently, thus the source points will not exactly correspond to the ones in the target point cloud. To tackle this problem, variations of the ICP consists in changing the considered cost function. Thus, in [Chen et al. 1992], they propose to minimize the sum of the point-to-plane distances, $d_i^{\perp}$, which allows to avoid this problem if the local structure is planar (Fig. 2.2). It was proved to converge faster and to be more robust than the point-to-point distance [Pottmann et al. 2004]. $d_i^{\perp}$ can be written as follows:

$$d_i^{\perp} = \| {}^t\mathbf{n}_i^{\top} \cdot ({}^t\mathbf{R}_s \, {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s - {}^t\overline{\mathbf{p}}_i) \| \tag{2.4}$$

where ${}^s\overline{\mathbf{p}}_i$ and ${}^t\overline{\mathbf{p}}_i$ are respectively the source and the target points, ${}^t\mathbf{n}_i$ the normal estimated at point ${}^t\overline{\mathbf{p}}_i$ and ${}^t\mathbf{R}_s$ and ${}^t\mathbf{t}_s$ respectively the 3×3 rotation matrix and 3×1 translation vector. Considering $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^T$, the minimal representation of ${}^t\mathbf{T}_s$, the point-to-plane error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{n} d_i^{\perp} \tag{2.5}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \mathbf{n}^{\top} \cdot (\mathbf{p}(\mathbf{q}) - \mathbf{p}) \tag{2.6}$$

where $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s \, {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$, $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$ and $\mathbf{n} = (\dots, {}^t\mathbf{n}_i, \dots)$.

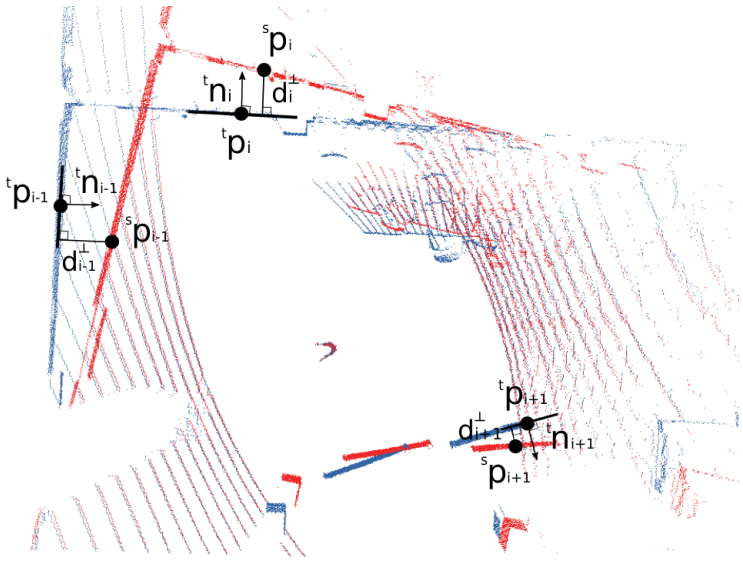


Figure 2.2 – Point-to-plane distance $d_i^{\perp}$ as described in [Low 2004]. In blue, ${}^t\mathbf{P}$ the target point cloud and the surface normals related to its points ${}^t\mathbf{p}_j$. In red, ${}^s\mathbf{P}$ the source point cloud to register.

To generalize equations (2.1) and (2.4) [Segal et al. 2009] proposes to use covariance matrices defining the degrees of freedom attributed to the source and the target during registration. By defining the covariance matrices according to the orientation of the local plane at each source and target point, a plane-to-plane registration is defined. This algorithm is called Generalized ICP (GICP). This strategy is efficient, but was proved to be slow to process for real-time applications.

As the number of planes is significantly lower than the number of points in a 3D structured environment, [Grant et al. 2019] proposes, with the algorithm Iterative Closest Point Plus Plane Optimization (IC3PO), to detect and segment planes (in opposition to locally approximate planes) in order to minimize a plane-to-plane distance $\mathbf{d}_i^{\Pi}$. The distance $\mathbf{d}_i^{\Pi}$ is given as follows:

$$\mathbf{d}_i^{\Pi} = \begin{pmatrix} {}^t\mathbf{R}_s\, {}^s\mathbf{n}_i - {}^t\mathbf{n}_i \\ \left[{}^t\mathbf{R}_s\, {}^s\mathbf{n}_i\right]^{\top} {}^t\mathbf{t}_s + {}^s\rho_i - {}^t\rho_i \end{pmatrix} \tag{2.7}$$

with ${}^s\mathbf{n}_i$ and ${}^t\mathbf{n}_i$ respectively the normals of source and target planes, ${}^s\rho_i$ and ${}^t\rho_i$ the distance of, respectively, source and target planes to the origin. The plane-to-plane error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbf{d}_i^{\Pi} \tag{2.8}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \begin{pmatrix} \mathbf{n}(\mathbf{q}) - \mathbf{n} \\ \rho(\mathbf{q}) - \rho \end{pmatrix} \tag{2.9}$$

with $\mathbf{n}(\mathbf{q}) = (..., {}^t\mathbf{R}_s {}^s\mathbf{n}_i, ...)$, $\mathbf{n} = (..., {}^t\mathbf{n}_i, ...)$, $\rho(\mathbf{q}) = (..., [{}^t\mathbf{R}_s {}^s\mathbf{n}_i]^\top {}^t\mathbf{t}_s + {}^s\rho_i, ...)$ and $\rho = {}^t\rho_i$. With IC3PO, when the problem is not fully constrained by the planes, the algorithm switches to a method that includes points in its minimization process.

Feature matching variations

In order to rely on features more robust than points, ICP-like methods can differ by the feature aimed to be matched. [Zhang et al. 2014] proposes to consider points of interests. Points, called feature points, present on planar patches or sharp edges are extracted. Then the correspondences with the target feature points are established, and the distance related to the geometrical object identified thanks to a Principal Component Analysis (PCA) is minimized. However, this method implies small relative motion between scans.

Another type of features that are found in the literature are local descriptors based on the neighborhood of points. [Rusu et al. 2009] proposes a descriptors called Fast Point Feature Histogram (FPFH), a multi-dimensional feature which describes the local geometry around a 3D point. The computation of this feature relies on 3D coordinates and surface normals of a point and its neighborhood. While it is rather fast to compute, FPFH are proved to be highly sensitive to gaussian noise [Guo et al. 2016].

To propose an alternative to structural methods, [Burel et al. 1995] introduces moment invariants as well as spherical harmonics invariants to describe 3D shapes.

Other local descriptors can be found such as spin images in [Johnson et al. 1999], or Signature of Histograms of OrienTations (SHOT) [Tombari et al. 2010]. In [Han et al. 2013] and [Wang et al. 2016], the intensity at the points is also taken into account.

In the case of plane-based registration, in [Chen et al. 2019a] a plane-line descriptor is proposed, combining distances and angles between quadruplets of planes.

Optimization method variations

ICP algorithms may vary in terms of optimization method. It is the method that will estimate the rigid transformation parameters. Two types of minimization can be identified: closed-form solutions which are in general fast, and iterative ones which are slower but more accurate.

In [Besl et al. 1992] a method based on the quaternions [Horn 1987] is used, giving a

closed-form solution to the point-to-point minimization problem. [Rusinkiewicz et al. 2001] proposes to use a solution based on the Singular Value Decomposition (SVD) [Arun et al. 1987].

In the point-to-plane case, no closed-form solution is available, [Rusinkiewicz et al. 2001] suggest to use a generic non-linear approach such as Levenberg-Marquardt or Gauss-Newton. Using these iterative methods allows to add robust function in the process [Fitzgibbon 2003], that will aim to reduce the influence of the outliers on the optimization. Another solution is to linearize the problem by using the small angle approximation. This allows to solve the minimization by using standard linear least squares methods [Low 2004] [Vlaminck et al. 2017].

For the plane-to-plane case, a closed-form solution is given in [Taguchi et al. 2013]. Rotation and translation are decoupled, allowing to estimate the rotation thanks to a method based on the SVD [Arun et al. 1987] and the translation by solving a linear system.

2.2.2 Other point cloud registration methods

In this section, we present other registration methods that aim to find the rigid transformation that links two point clouds.

Gaussian image

In [Brou 1984], the gaussian image is used in order to find the orientation of an object by using the projection on a gaussian sphere. The normals at the points are represented as points on a unit sphere centered at the origin. Thus, planes will show up as cluster of points on the sphere. [Horn 1984] adds the area information to the planar patches in order to refine the matching. Another extension is proposed in [Kang et al. 1991] where distance between planes is taken into account, giving the possibility to estimate the translation that registers two point clouds. These methods imply a total overlap between point clouds and lead to coarse alignment.

More recently, in [Sanchez et al. 2017], an algorithm called Structured Scene Features-based Registration (SSFR) is presented. With SSFR the rotation is first solved using the gaussian image, and the translation is estimated thanks to a correlation of histograms performed on the rotated point clouds. The point clouds are projected successively on each translation axis, and histograms are built from the projections. The maximum of the correlation between source and target histograms corresponds to the expected translation. The bin width in the histogram is first set to a large number in order to perform a coarse registration. It is then reduced to refine the estimation. SSFR is an accurate registration algorithm. However, computation time increase rapidly when a large number of points are considered. Nevertheless, it is robust to

subsampling, thus, the author suggests to use it for coarse registration if computation time is a limiting criterion.

Density distribution

Another method to register point clouds is based on the probability density function of the point clouds. It was first developed for 2D applications [Biber et al. 2003], but was extended for 3D in [Magnusson et al. 2007] for autonomous mining vehicles. This method is called the Normal Distribution Transform (NDT). Instead of representing the data by individual points, it is represented thanks to a set of gaussian probability distributions describing the probability of finding a point at a certain position. The registration is then solved using this representation with standard numerical optimization techniques. NDT implies using a grid to represent the data and the resolution parameter can be hard to set properly, moreover, like the ICP algorithm, NDT is sensitive to local minima.

RANSAC-based alignment

To register point clouds, we can also find methods based on a sample consensus. In [Rusu et al. 2009], SAmple Consensus Initial Alignment (SAC-IA) is presented. A large number of correspondence candidates are sampled and ranked. It uses the FPFH, so for each sample in the source, the list of points in the target with an histogram similar to the query one is built. From this list, a point is randomly selected. Then the rigid transformations defined by triplets of the sample points and their correspondences are computed and an error metric, the Huber penalty [Huber 1992], is computed to evaluate the quality of the estimations. The estimation with the smallest error is chosen. SAC-IA allows fast initial registration, but the method by itself cannot issue a fine estimation, thus an additional optimization such as a Levenberg-Marquardt method must be performed if more precision is expected.

Global registration

To perform global registration, [Pathak et al. 2010] presents the algorithm Minimally Uncertain Maximum Consensus (MUMC). MUMC is based on plane registration. Planes are extracted from the point clouds and the algorithm works directly on the "plane clouds". By finding the set of correspondences in the two scans that give the most geometrically consistent rigid

transformation, the estimation is made. An exhaustive search on rotation and translation is affordable thanks to the reduced input number. This approach proves to be more robust than point or grid based approaches in environments composed of planes, and is also faster, and requires less memory.

Neural Networks

LiDAR point clouds registration is a challenging problem for Neural Network applications. As we said in section 1.1.1, LiDAR data is sparse and unorganized. This makes it very different from image processing. For instance, it will be highly unlikely to match a source point to its exact correspondent in the target. Nevertheless, the trend for Neural Networks issued some networks designed for 3D point clouds registration.

In order to perform outliers rejection by classification on the point correspondences, 3DregNet is proposed in [Pais et al. 2020]. This Deep Neural Network (DNN) allows to compute the transformation linking two point clouds by giving the correspondences as input. First, the correspondences are classified as inliers or outliers. Then, a network estimates a coarse registration. To refine the estimation, the same network can be used one more time. 3DregNet proves to be as accurate as other state of the art algorithms while being faster.

In [Lu et al. 2019], DeepICP, an algorithm designed to perform end-to-end learning-based 3D point cloud registration is presented. The Convolutional Neural Network (CNN) of DeepICP is trained in order to directly "guess" the feature point correspondences without having to use a rejection algorithm (such as RANdom SAmple Consensus (RANSAC)). The registration is based on feature points extracted using PointNet++ [Qi et al. 2017] adding semantic to them and allowing to rely mostly on stable and not dynamic objects by identifying them. The feature points matching is performed according to the matching similarity probabilities estimated by a trained CNN. Even if the name includes ICP, only one iteration is performed for the inference. DeepICP turns out to be as accurate as state of the art algorithms.

2.3 Simultaneous Localization And Mapping related to point clouds

As mentioned in section 1.1.2, point cloud registration has wide applications, one of them is SLAM. Registration is performed in a pairwise manner, to estimate the relative translation and rotation that links two sets of data, whereas SLAM will estimate sensor pose to a global

scale, on a sequence of data. It is a technique aiming to simultaneously estimate the sensor motion and reconstruct the visited environment. In other words, a robot put in an unknown environment is then able to map its surroundings and locate itself in it. Moreover, SLAM includes the notion of map correction, thanks to the knowledge the robot has of its previous locations. It was first proposed in [Chatila et al. 1985]. In closed environments, Global Positionning System (GPS) signals are weak or even non-existent and cannot be used. SLAM techniques are the perfect fit to overcome this problem. SLAM is performed with a variety of robotic platforms and a variety of sensors. One of the most popular is visual-SLAM, which estimates successive camera poses, through input images. There is also LiDAR-based SLAM, which will be described in more details in this section, that takes as input point clouds and estimates the successive sensor poses. They both give the possibility to generate 3D maps. Camera and LiDAR information can also be fused [Debeunne et al. 2020]. A simplified SLAM framework is provided in Figure 2.3.

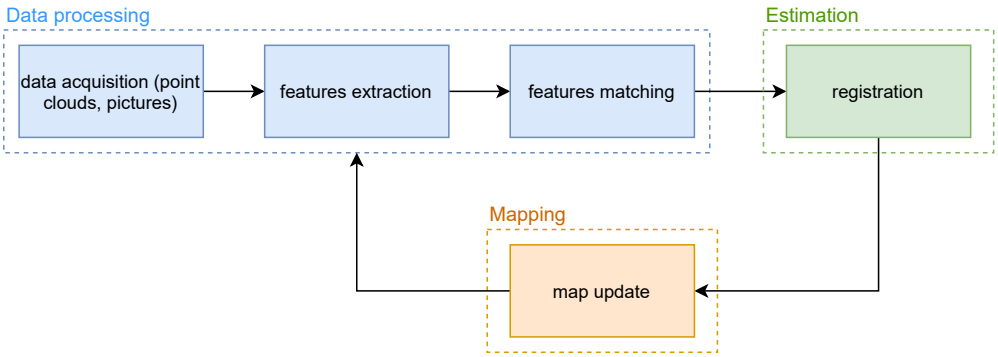
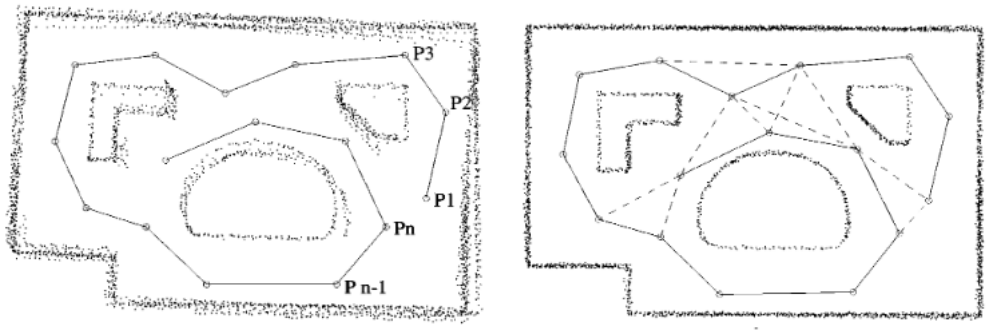


Figure 2.3 – Simplified SLAM framework.

As the literature on SLAM is rather dense, this part will focus only on LiDAR-based SLAM approaches. It includes registration methods similar to the ones presented in the previous section. In figure 2.4, an application of LiDAR graph-based SLAM is given where we can see a direct application of SLAM on global mapping. In (a) we can see that the scans are misaligned due to accumulated pose errors, whereas in (b), thanks to the relative pose constraints between scans, the alignment is corrected and generates a consistent map.



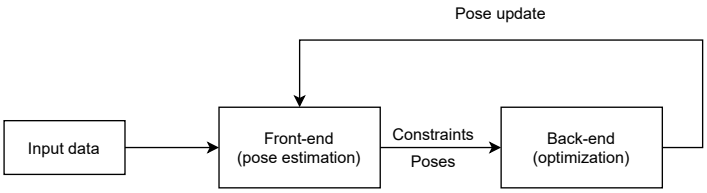
(a) (b)

Figure 2.4 – (Credits: [Lu et al. 1997] article). An example of scan mapping using SLAM. (a) Original scans alignment. (b) Scans alignment with SLAM. Solid lines: constraints due to odometry measurements and pair of scans alignments. Dotted lines: constraints due to pair of scans alignments.

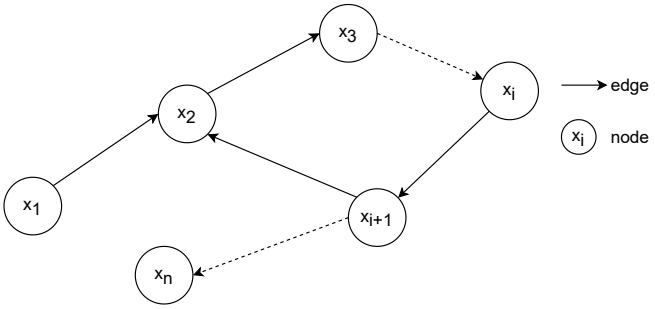
A SLAM approach using probabilistic method is proposed in [Grisetti et al. 2007], with a single-line LiDAR (in other words a 2D LiDAR). It uses a particle filter in which each particle represents a potential robot pose and carries an individual map of the environment. An improved Rao-Blackwellized particle filter method is used to reduce the number of sampled particles. Maintaining a large number of particles leads to non-negligible computation time. A proposition using occupancy grid map is given in [Grisettiyz et al. 2005] showing an order of magnitude smaller than classical approaches in the number of particles. However this method is not applicable to 3D in case of large-scale data because of memory requirements.

One way to formulate LiDAR-based SLAM is by using graph-based methods. A tutorial is provided in [Grisetti et al. 2010]. In graph-based SLAM the poses of the sensor are considered as nodes in a graph, and the spatial constraints between poses resulting from the observations are the edges between the nodes. The graph-based SLAM was first proposed in [Lu et al. 1997].

In general, a graph-based SLAM framework can be split in two parts: front-end and back-end (Figure 2.5). The front-end part aims to estimate the set of successive poses of the robot thanks to the input data. This input data comes with noise, and the previous estimation might come with a small error, and, with time, the error generated will accumulate in the successive pose estimations. The role of the back-end part is to suppress this cumulative error and to improve the estimation and the accuracy of the created map.



(a) A general graph-based LiDAR SLAM framework



(b) A pose-graph representation.

Figure 2.5 – A general graph-based framework and a pose-graph. Each node $\mathbf{x}_i$ represents a robot pose. The connecting edges represent the constraints between poses.

In [Behley et al. 2018] SuMa, a surfel-based mapping method is presented, with projection data association and point-to-plane metric. It was improved and made more robust to dynamic environment in [Chen et al. 2019b] by adding semantic information. Recently, in [Chen et al. 2021] SuMa is used in a LiDAR-based SLAM using a deep neural network, OverlapNet.

Loop Closure

In order to further constrain the SLAM problem, a loop closure step is usually added. Loop closure allows SLAM applications to correct the accumulated drift produced by the noise in successive transformation estimations, when a robot is able to identify that it is located in a previously visited area. A simple example is given in figure 2.6. A robot trajectory is estimated, but due to the accumulation of errors, a drift appears in the estimation. Fortunately, a loop is performed in this trajectory, and the robot is able to tell that it comes back to a place it has already visited. Thanks to the implied constraints, the trajectory is corrected.

Loop detection can be achieved thanks to feature-based approaches, such as geometric primitives like lines or planes, in a matching process. In [Grant et al. 2019], the loop detection is performed thanks to the detection of already met planes. Interest points are used in [Steder et al. 2010] and [Steder et al. 2011]. In [Dubé et al. 2017], a segment-based approach is presented. Using a RANSAC algorithm in a geometric test, the loops can be identified in

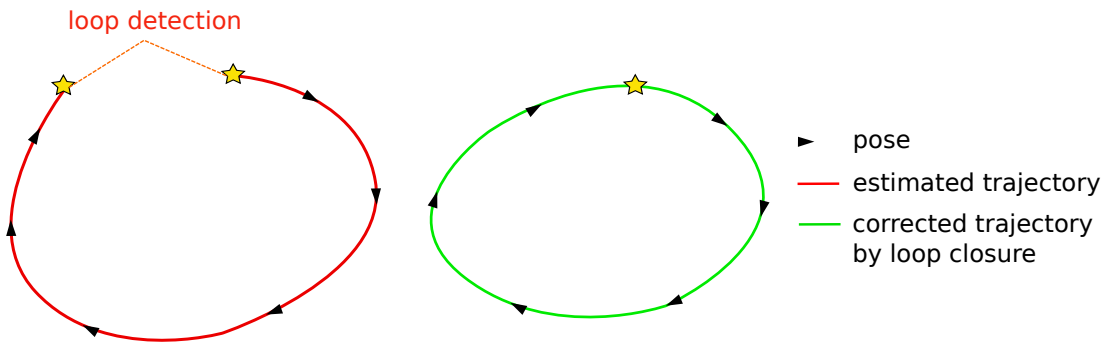


Figure 2.6 – Loop closure detection example. The first estimated trajectory accumulates error and drifts, but a loop is detected. It allows to correct the estimation and generate a more consistent trajectory.

the matching procedure. In [Magnusson et al. 2009], they propose to use the NDT algorithm to detect loops. Locations are described with feature histograms. Those features are based on surface orientation and smoothness. Finally, to detect loops, the feature histograms are matched.

2.4 Conclusion

In this chapter we presented an overview of the algorithm designed for point clouds registration. A particular focus was made on the ICP algorithm and its variations. Also, a presentation of LiDAR-based SLAM was given.

In this thesis, we want to address the 3D point clouds registration problem by proposing robust algorithms that will allow to have a good trade off between accuracy and computational efficiency. The state of the art demonstrates that using only points is limiting and that preferring integrating planar primitives is a better perspective. For instance, minimizing the point-to-plane distance was proved to converge faster and be more robust than minimizing the point-to-point one. Using only planes can efficiently register point clouds, but the resulting estimation is coarse. However, it can also tremendously reduce the dimensionality of the registration problem. The state of the art also shows that using iterative methods for optimization leads to more accurate results (and allows to use robust outliers rejection functions) than closed-form solutions, however, it leads to slower computation time. Thus we make the choice to investigate how we can use iterative methods with robust functions for optimization to ensure accuracy while keeping an acceptable computation time.

The following chapter presents the datasets used with point clouds registration algorithms.

Chapter 3

Datasets for point cloud registration

3.1 Introduction

To guaranty the efficiency of a registration algorithm, or to compare it to others, it needs to be tested on benchmark datasets. In this chapter, we will first present state of the art datasets that were designed to evaluate point cloud registration algorithms. A fuller description of the Autonomous Systems Lab (ASL) dataset [Pomerleau et al. 2012], as well as an explanation on how to read the graph we generated for our experiments will be provided in a dedicated section. Finally, we will present our first contribution, LOOP'IN, a dataset created during this thesis designed to evaluate the ability of registration algorithms to close loops in long sequences.

3.2 State of the art

This section will focus on the existing open source datasets used to assess point clouds registration algorithms properties.

In the early years of point cloud registration, the Stanford 3D Scanning Repository [Turk et al. 1994] gave the possibility to evaluate registration algorithm on several objects captured in 3D. Later, the lack of open source dataset with reliable ground truth and the rise of algorithms designed for autonomous driving applications motivated the creation of Malaga [1] [Blanco et al. 2009]. It is composed of six urban outdoors scenarios captured thanks to cameras and laser scans. Soon after, the Ford Campus dataset [2] [Pandey et al. 2011] was proposed for similar applications. It is composed of two urban datasets, captured with an autonomous car equipped

1. https://www.mrpt.org/robotics_datasets
2. http://robots.engin.umich.edu/SoftwareData/Ford

with several sensors among which cameras and LiDARs. The authors of KITTI [3] [Geiger et al. 2012] made the most of the autonomous driving platform they had for other works, to provide an exhaustive dataset coming with stereo, optical flow, visual odometry, SLAM and 3D object detection.

Until then, the point cloud registration datasets were mainly dedicated to urban navigation, and in order to propose a dataset with a larger spectrum of environmental structures, the ASL dataset [4] [Pomerleau et al. 2012] is created. It provides high-quality data with millimeter-precision ground truth of the sensor poses. It is a challenging dataset, coming with a variety of scenarios. These scenarios are used in our experiments and are detailed in the next section.

More and more datasets of LiDAR data were created, such as PAVIN [5] [Sanchez et al. 2017] composed of an outdoor sequence, or FR-IOSB and KA-Urban datasets [6] [Li et al. 2021], respectively including three and five outdoor sequences captured with a LiDAR and an Inertial Measurement Unit (IMU) on a mobile platform for the first one and mounted on a backpack for the latter. They are designed more specifically for autonomous driving applications. The difficulty is to find datasets with their ground truth.

A way to obtain the ground truth for the pose of the LiDAR sensor is to generate artificial data from a 3D model. It is now possible to do so using an open-source plugin in Blender called Blensor [Gschwandtner et al. 2011] that allows to generate synthetic 3D point clouds from a simulated trajectory. This is used in SynthCity [Griffiths et al. 2019] to provide a dataset for classification of urban furniture, but one could easily imagine using this data for registration purpose, as the transformation between scans is known. Synthetic data can also be created with CARLA [Dosovitskiy et al. 2017], an open-source urban simulator.

The mentioned datasets and a few characteristics are summarized in Table 3.1.

3. http://www.cvlibs.net/datasets/kitti/
4. https://projects.asl.ethz.ch/datasets/
5. https://projet.liris.cnrs.fr/pcr/
6. https://github.com/KIT-ISAS/lili-om

Table 3.1 – Available open Datasets designed for localization applications.

Dataset name	ground truth	real/synthetic	indoor/outdoor
Malaga [Blanco et al. 2009]	yes	real	outdoor
Ford Campus [Pandey et al. 2011]	yes	real	outdoor
KITTI [Geiger et al. 2012]	yes	real	outdoor
ASL dataset [Pomerleau et al. 2012]	yes	real	mixed
PAVIN [Sanchez et al. 2017]	no	real	outdoor
SynthCity [Griffiths et al. 2019]	yes	synthetic	outdoor
FR-IOSB [Li et al. 2021]	no	real	outdoor
KA-Urban [Li et al. 2021]	no	real	outdoor

3.3 Autonomous System Labs dataset

Description

In [Pomerleau et al. 2012] a challenging dataset (Fig 3.1, composed of eight sequences (of which two corresponds to the same area but in different seasons), to evaluate registration algorithm's robustness to different scenarios, is presented. One of the main advantages of this dataset is that each sequence comes with the ground truth of the sensor poses measured for each scan, with millimeter-precision thanks to the use of a theodolite. It is a dataset, coming with a variety of scenarios, with indoor and outdoor environments and dynamic scenes. All sequences were recorded using a Hokuyo UTM-30LX.

Each sequence comes with its specificity:

— *Apartment:* This sequence is captured indoor. It is designed to evaluate algorithm robustness to outliers coming from dynamic elements (e.g. moved furniture). The sequence was captured moving the sensor on a 2D plane in an apartment. It is a very structured scene (walls, ceiling, floor). The sequence is composed of 44 scans of about 365,000 points.

— *ETH:* This sequence is captured mostly indoor besides at the end. It aims to evaluate robustness of registration to repetitive elements. This scene was captured in a long hallway, following a straight path. It is composed of a wall, a curved ceiling and numerous pillars and arches which are repetitive elements. The sequence is composed of 35 scans of about 191,000 points.

— *Stairs:* This sequence is captured indoor. It aims to evaluate robustness to rapid varia-

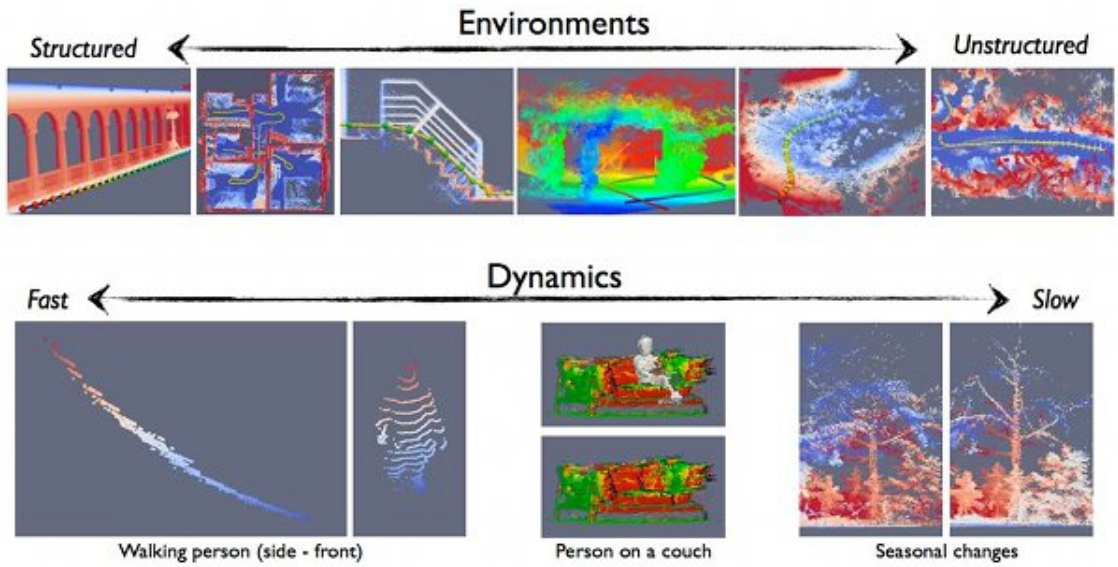


Figure 3.1 – ASL dataset (credits: ETH Zürich)

tions in scanned volumes. It starts in a long corridor, then a small staircase is crossed and finally the last scan is captured outside of the building. The path in the staircase shows that considering only 2D paths is not valid. The sequence is composed of 30 scans of about 191,000 points.

— *Mountain plain:* This sequence is captured outdoor in a concave basin. It aims to evaluate robustness to low-constrained environment. It was recorded while the sensor was moving down a small slope. The sequence is composed of 31 scans of about 102,000 points.

— *Gazebo:* This sequence captured outdoor. It aims to evaluate robustness to semi-structured area as well as moving people. The scene was captured in a park along a paved path. The summer sequence is composed of 32 scans of about 170,000 points. The winter sequence is composed of 32 scans of about 153,000 points.

— *Wood:* This sequence captured outdoor. It aims to evaluate robustness to unstructured environments. The scene is mainly composed of vegetation besides from a small paved road crossing the wood. The summer sequence is composed of 37 scans of about 182,000 points. The winter sequence is composed of 32 scans of about 178,000 points.

Besides from the diversity and quality provided by the dataset, some relative motion between two scans may be unrealistic considering navigation applications. Also, the dataset was recorded with a LiDAR providing a high density of points, it is important to check if the tested algorithms are also robust with lower quality point clouds.

Evaluation metrics linked to ASL dataset

As stated previously, the ASL dataset comes with the ground truth of the sensor pose in the global frame. Knowing the poses of the sensor allows to evaluate the accuracy of the estimation provided by the tested algorithm. To do so, as in [Pomerleau et al. 2013], the accuracy is evaluated with the Euclidean distance Δ_t between the estimated transformation and the ground truth for translation and the geodesic distance Δ_r for rotation:

$$\Delta_t = \|\hat{\mathbf{t}} - \mathbf{t}^*\| \tag{3.1}$$

$$\Delta_r = arccos\left(\frac{trace(\mathbf{R}^{*-1}\hat{\mathbf{R}}) - 1}{2}\right) \tag{3.2}$$

with $\hat{\mathbf{t}}$ and $\hat{\mathbf{R}}$ the estimated translation and rotation, $\mathbf{t}^*$ and $\mathbf{R}^*$ the ground truth translation and rotation respectively.

In the upcoming experiments, in order to define a registration as successful, the estimations have to follow two conditions:

$$\Delta_t < t_{tr} \qquad \text{and} \qquad \Delta_r < t_{rot} \tag{3.3}$$

with t_{tr} the translation error threshold and t_{rot} the rotation error threshold, respectively set to 0.1m for translation and 2.5° for rotation, as suggested in [Magnusson et al. 2015]. Note that rotation and translation errors are sometimes presented separately but a result is valid only if both rotation and translation errors are smaller than their respective threshold. In order to represent these errors more visually than percentages, graphs representing cumulative probabilities of the errors are provided to the reader. An example of such curves is given in Figure 3.2. The horizontal axis represents the rotation error from the ground truth and the vertical axis the cumulative probabilities of the rotation error. The vertical bar represents the error threshold, in this case t_{rot}. The more top-left the curve the better the algorithm performs. The expected behavior is to attain 1 (meaning all point clouds of the sequence are registered) before the error threshold is reached. If so, it means that 100% of the scans of the sequence are successfully registered (according to the chosen thresholds). If 1 is not reached before the threshold, it means that some registration were unsuccessful or not accurate enough. For instance in this example: the error threshold t_{rot} is set to 2.5°.

— The blue curve reaches t_{rot} at around 0.95, meaning that 95% of the estimation have an error smaller than the chosen threshold. The curve eventually reaches 1 but the

estimation are too far from the ground truth. The conclusion is that the algorithm providing the blue curve successfully registered 95% of the scans.

— The red curve reaches t_{rot} at around 0.80, meaning that 80% of the estimation have an error smaller than the chosen threshold. After that the curve is plateauing around 0.80. The most likely explanation is that the algorithm diverged or is stuck in a local minima for the unsuccessful registrations. The conclusion is that the algorithm providing the red curve successfully registered 80% of the scans.

— The green curve reaches 1 before t_{rot} is met, meaning that 100% of the estimation have an error smaller than the chosen threshold. The conclusion is that the algorithm providing the green curve successfully registered all scans of the sequence.

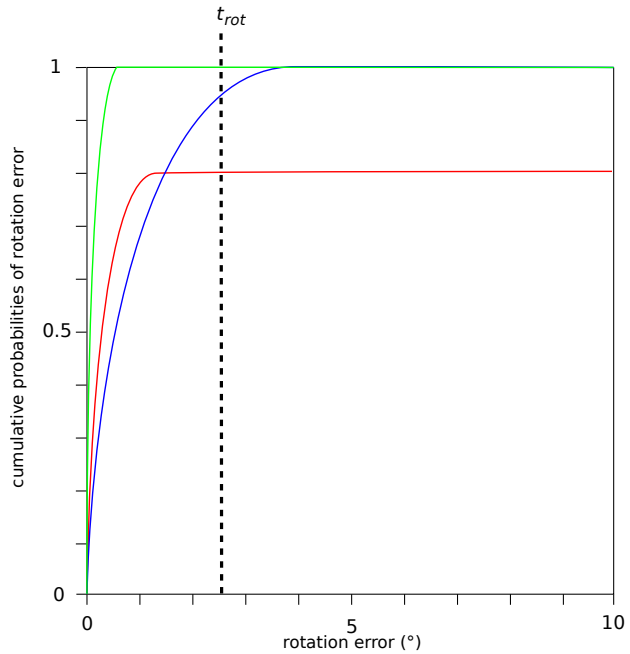


Figure 3.2 – Example of cumulative probabilities for rotation error. The horizontal axis represents the rotation error from the ground truth and the vertical axis the cumulative probabilities of the rotation error.

3.4 LOOP'IN dataset

3.4.1 Description

LOOP'IN is a challenging dataset designed, during this thesis, to evaluate registration algorithms on long sequences and loop closure on real-world indoor data[7]. Our aim was to provide a new challenging dataset to the point cloud community. As we did not have the possibility to measure accurately the ground truth for the sensor pose, we made sure to add loops in the trajectory, thus the user would be able to check if his algorithm is prone to drift or is able to close the loops. LOOP'IN was captured with a VLP-16 Puck Hi-Res, which is a LiDAR providing a lower and sparser density of points than the one provided in ASL dataset, making it even more challenging. It is composed of 16 lasers in a 20° range. The LiDAR is mounted on a moving platform pushed by an operator. As the captures are sometimes performed in narrow areas, the LiDAR is tilted towards the floor, to make sure to capture floor points.

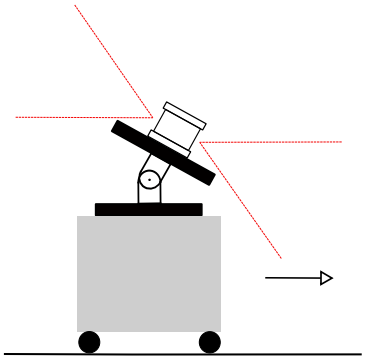


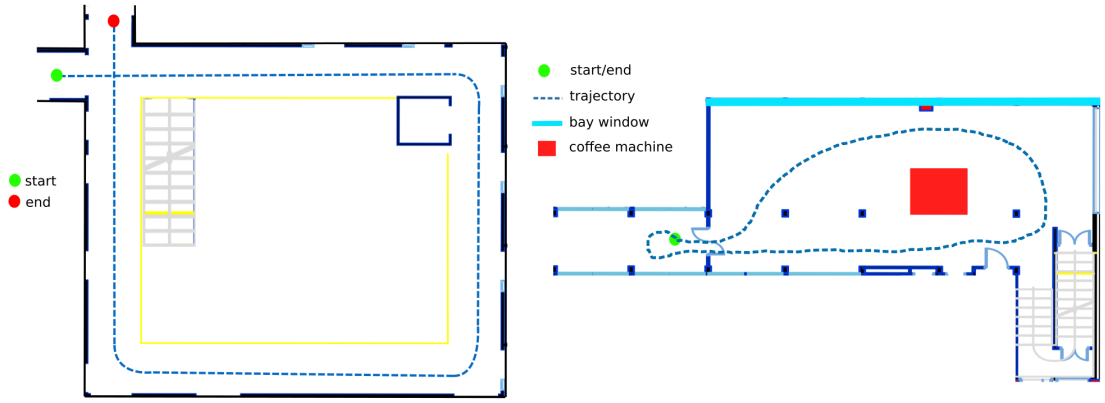
Figure 3.3 – Configuration of the moving plateform.

The dataset is composed of two indoor sequences:

— *Balcony Loop*(Fig 3.4 (a)): The sequence is composed of a loop with a crossing of the trajectory in the end. The loop is made around a balcony surrounding an almost square room. The sequence is composed of 142 scans of about 28,000 points. Some point clouds of the sequence are displayed in Figure 3.5 (a).

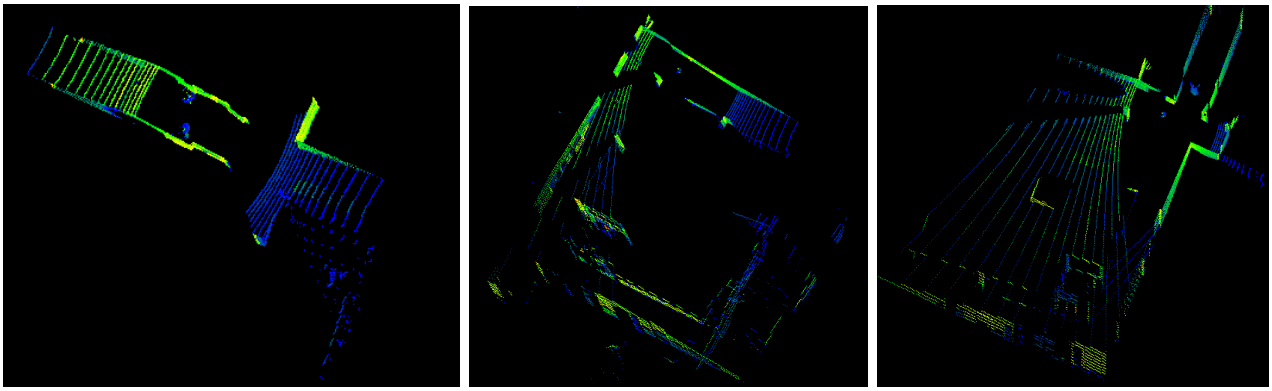
— *Coffee Hall Loop*(Fig 3.4 (b)): The sequence is composed of a loop starting and ending at the same location, in a large coffee hall. Furniture is present as well as some moving people. One side of the hall is a large bay window (making reflections appear on scans). The sequence is composed of 171 scans of about 28,000 points. Some point clouds of the sequence are displayed in Figure 3.5 (b).

7. https://github.com/kfavre/LOOP-IN_dataset.

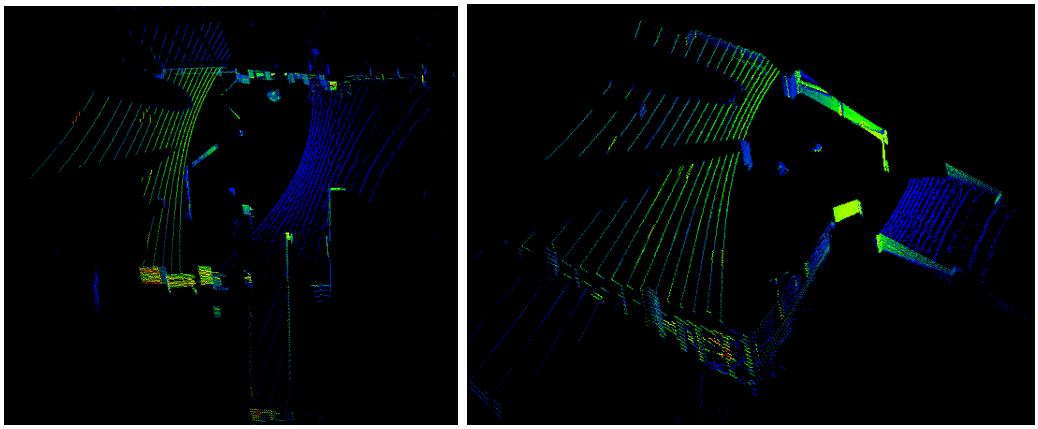


(a) *Balcony Loop* sequence (b) *Coffee Hall Loop* sequence

Figure 3.4 – Trajectories of the sensor in LOOP'IN dataset.



(a) *Balcony Loop*



(b) *Coffee Hall Loop*

Figure 3.5 – Extracts of point clouds from LOOP'IN dataset.

3.4.2 Error case

During the realization of this dataset we encountered a few problems related to the fact that the acquisition were made indoors. Those problems are common when capturing data and can be easily fixed. We list them bellow.

Sliding on yaw-axis

LOOP'IN dataset was captured in areas which were sometimes tight. If the LiDAR is placed horizontally, it is more likely than no floor (or ceiling) points will be captured. In case of 6-DoF registration, not taking these points removes a constraint on yaw-axis, thus, during transformation estimation, we end up with the source point cloud that will unwantedly slide along this axis (Figure 3.6).

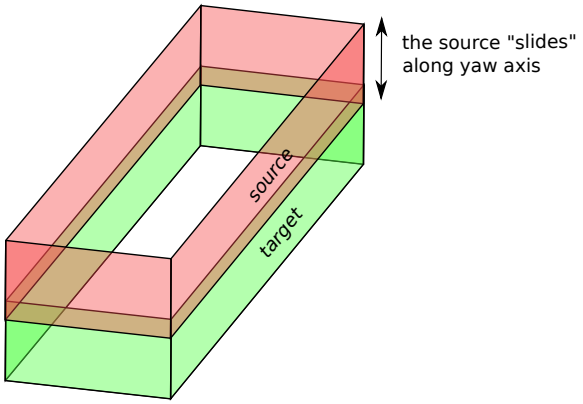


Figure 3.6 – Description of sliding problem on yaw-axis.

To cope with this issue, we just had to add a tilting angle to our device, and to make sure that floor points were captured.

Corridor effect

In the same idea of missing constraints, it can happen that the capture trajectory goes through a corridor. This time, it is the roll-axis of the mobile device which is not constrained. The result is similar to the previous one. As we can see on figure 3.7, the source slides along the corridor axis of the target. This phenomenon is called the "corridor effect".

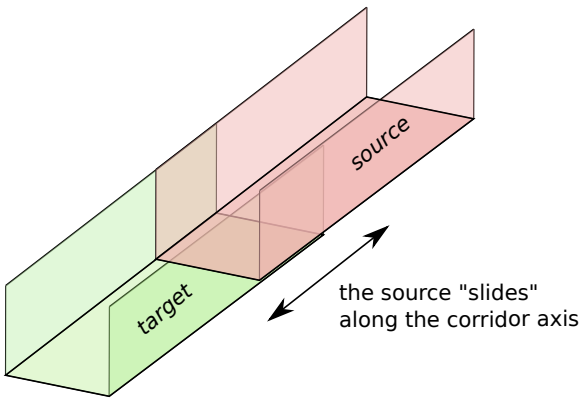


Figure 3.7 – Description of the corridor effect.

This time the problem is harder to solve. When the capture is performed within a corridor with rooms along it, all doors should be open, in order to capture points on the walls of these rooms, and to avoid getting this effect. If this is not possible, in case of LiDAR-only registration, a motion constraint must be added to the evaluated algorithm, for instance a Kalman filter with the constraints of constant speed.

3.5 Conclusion

In this chapter we presented a state of the art of the existing datasets designed to evaluate point cloud registration algorithms. Most of them are designed for urban navigation. We further presented the ASL dataset, a dataset composed of various types of scenarios. This dataset is provided with the ground truth of the sensor poses, which allows us to evaluate the accuracy of our algorithms, that is why it is used in this manuscript to evaluate the proposed algorithms. We also presented LOOP'IN, a challenging dataset created during this thesis, captured in indoor environments. As it was no possible to measure precisely the ground truth of the sensor pose, loops were added in the trajectories so that we would be able to check if the algorithms successfully close the loop or fail in these long sequences.

The next chapter outlines our multi-resolution registration algorithm for 3D point clouds.

Chapter 4

Multi-resolution registration of 3D point clouds

4.1 Introduction

The problematic we are willing to solve is to register 3D point clouds as accurately as possible in an acceptable time frame. As stated in chapter 2, the ICP algorithm [Besl et al. 1992] is well fitted for 3D point clouds registration. We saw that this method is rather fast, efficient when favorable conditions are met and easy to implement but comes with a few drawbacks.

This chapter presents a registration algorithm based on a multi-resolution scheme, Gauss-Newton based Multi-Resolution ICP (GNMR-ICP). A first section introduces a state of the art related to point-to-plane ICP algorithm as well as multi-resolution approaches. Then, the proposed multi-resolution registration algorithm is described in more details. Finally, the accuracy and speed of the proposed method is assessed in some experiments.

4.2 State of the art

This section provides a reminder for the reader of the notations used for the ICP formulation as well as a small state of the art dedicated to this chapter. The aim of the registration problem is to align two point clouds denoted: ${}^{s}\mathbf{P}$ the source point cloud and ${}^{t}\mathbf{P}$ its target point cloud.

4.2.1 ICP formulation

Point-to-point ICP

The point-to-point distance d_i used in the classical ICP algorithm [Besl et al. 1992] can be formulated as follows:

$$d_i = \| {}^{t}\mathbf{T}_s \, {}^{s}\mathbf{p}_i - {}^{t}\mathbf{p}_i \| \tag{4.1}$$

where ${}^s\mathbf{p}_i$ and ${}^t\mathbf{p}_i$ are respectively the source and the target points, and ${}^t\mathbf{T}_s$ the 4×4 rigid transformation that links ${}^s\mathbf{P}$ to ${}^t\mathbf{P}$. Denoting $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^T$, the minimal representation of $\mathbf{T}$, where θ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^t\mathbf{R}_s$, the point-to-point error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{n} d_i \tag{4.2}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \mathbf{p}(\mathbf{q}) - \mathbf{p} \tag{4.3}$$

where $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$ and $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$.

Point-to-plane ICP

The ICP based on the point-to-plane distance was presented in section 2.2.1. It is defined by:

$$d_i^{\perp} = \|{}^t\mathbf{n}_i^{\top} \cdot ({}^t\mathbf{R}_s {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s - {}^t\overline{\mathbf{p}}_i)\| \tag{4.4}$$

where ${}^s\overline{\mathbf{p}}_i$ and ${}^t\overline{\mathbf{p}}_i$ are respectively the source and the target points, ${}^t\mathbf{n}_i$ the normal estimated at point ${}^t\overline{\mathbf{p}}_i$ and ${}^t\mathbf{R}_s$ and ${}^t\mathbf{t}_s$ respectively the 3×3 rotation matrix and 3×1 translation vector. Considering $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^T$, the point-to-plane error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{n} d_i^{\perp} \tag{4.5}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \mathbf{n}^{\top} \cdot (\mathbf{p}(\mathbf{q}) - \mathbf{p}) \tag{4.6}$$

where $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$, $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$ and $\mathbf{n} = (\dots, {}^t\mathbf{n}_i, \dots)$.

4.2.2 Minimization

Minimizing the point-to-plane distance is a classic non-linear least squares problem. In section 2.2.1 we saw that [Rusinkiewicz et al. 2001], suggests to solve this optimization problem by linearizing it. It is a widely used method for 3D point cloud registration, as it is easier to solve than a nonlinear problem and runs faster. Indeed, if sensor motion is small between two acquisitions, the small angle approximation can be made.

This formulation of the ICP is used for comparisons with our proposed method, thus we describe it in more details in the insert below, entitled **Small angle approximation for point-to-plane ICP by [Low 2004]**.

Small angle approximation for point-to-plane ICP by [Low 2004]

The rotation in ${}^t\mathbf{R}_s$ is redefined as follows: ${}^t\mathbf{R}_s = \mathbf{R}_z(\gamma) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha)$ where α, β, γ are rotation around respectively x, y and z axes. The small angle approximation states that for a small angle θ its value can be approximated to 0 which yields $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. Therefore when α, β, $\gamma \approx 0$ an approximation of the ${}^t\mathbf{R}_s$ can be written:

$$ {}^t\mathbf{R}_s^* = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \tag{4.7} $$

Equation (4.4) becomes a linear expression of the six parameters of $\mathbf{q}$ of the type $\mathbf{Aq} - \mathbf{b}$ where $\mathbf{q} = [\alpha \ \ \beta \ \ \gamma \ \ t_x \ \ t_y \ \ t_z]^\top$. Minimizing this problem is equivalent to minimizing:

$$ \hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \ \|\mathbf{Aq} - \mathbf{b}\|^2 \tag{4.8} $$

where

$$ \mathbf{A} = \begin{bmatrix} \vdots & \vdots \\ {}^t\mathbf{n}_i{}^\top \left[{}^s\overline{\mathbf{p}}_i\right]_\times & {}^t\mathbf{n}_i{}^\top \\ \vdots & \vdots \end{bmatrix} \tag{4.9} $$

and

$$ \mathbf{b} = \begin{bmatrix} \vdots \\ {}^t\mathbf{n}_i{}^t\overline{\mathbf{p}}_i - {}^t\mathbf{n}_i{}^s\overline{\mathbf{p}}_i \\ \vdots \end{bmatrix} \tag{4.10} $$

As stated in section 1.4.2.1, such an expression can be solved by:

$$ \hat{\mathbf{q}} = \mathbf{A}^+\mathbf{b} \tag{4.11} $$

where $\mathbf{A}^+$ represents the Moore-Penrose pseudo-inverse of $\mathbf{A}$. Note that $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ must be injected in the original rotation matrix ${}^t\mathbf{R}_s$ and not in the approximated one ${}^t\mathbf{R}_s^*$ (which does not have rotation matrix properties (see section 1.4.1 for more details)).

This approximation is used in [Vlaminck et al. 2017]. This method tends to be highly sensitive to noise as it leads to several approximations in the rotation matrix of the rigid transformation and is not suitable for large rotational motions.

Solving the optimization problem using non-linear iterative approaches (such as Gauss-Newton or Levenberg-Marquardt methods seen in section 1.4.2.2) allows to estimate the transformation while easily adding robust M-estimators functions to it (an example is given in [Fitzgibbon 2003] using the point-to-point distance and a Levenberg-Marquardt minimization).

4.2.3 Outliers rejection

The processed data is most likely containing outliers (e.g. wrong point matches). Those outliers can have an unwanted, non-negligible impact, on the estimation of the solution. So, it is important to get rid of them. To make the minimization robust to outliers, several techniques have been designed. They can be divided into two kinds:

— hard rejection (relying on a threshold);

— soft rejection (relying on continuous functions).

An analysis of fourteen robust functions (e.g. Cauchy's, Tukey's, Huber's M-estimators, trimmed filters [Phillips et al. 2007], ...) applied to ICP is made in [Babin et al. 2019], designed for mobile robotic applications. Their conclusion is that, when they are correctly tuned, most outliers filters have similar performances. This study also shows that Cauchy's M-estimators are more stable against different environments. In [Segal et al. 2009], a filter based on a maximum distance between point matches is proposed. A threshold corresponding to this distance is set manually. If the distance between matched points is bigger than this threshold, they are not taken into account in the minimization. While making the function more robust to outliers, [Fitzgibbon 2003] also states that using M-estimators in the minimization makes it more time consuming.

4.2.4 Multi-resolution ICP

To reduce computation time of the matching process, [Jost et al. 2003] introduce a multi-resolution ICP algorithm using the point-to-point distance. The idea is to register the point clouds from coarse to fine resolution. The data being significantly reduced in the levels with the lowest resolution, the algorithm converges faster and the transformation estimated at the lowest level is used as an initialization for the level below. The experiment is performed on a

duck toy captured with a structured light range finder. Even if the application is different from ours, the root problem is similar. The article demonstrates that the use of a multi-resolution scheme tremendously improves computation time. Such a multi-resolution scheme is used in [Vlaminck et al. 2017] thanks to an octree structure. This time, the point-to-plane distance is minimized, thanks to the small angle approximation. Once more, this scheme speeds up the whole process.

4.2.5 Positioning

Our first focus is in registering point clouds as accurately as possible, so it is natural to chose the point-to-plane distance for the optimization of the problem. Following this idea, the minimization is done using the Gauss-Newton approach, allowing to solve the optimization problem while keeping a minimal representation of the parameters to estimate. Moreover, M-estimators are used in order to make the problem more robust to wrong point matches as they are known to be a good fit in this kind of applications. As stated previously, the use of M-estimators tends to increase computation time. To tackle this problem, a multi-resolution scheme is added in the process.

4.3 Proposed multi-resolution point-to-plane ICP: GNMR-ICP

This section describes our 3D point cloud registration method, GNMR-ICP, based on a multi-resolution scheme, that estimates transformation by using a robust non-linear method. An example of registration performed by GNMR-ICP is given in Figure 4.1.

4.3.1 Overall description of GNMR-ICP

The proposed method is based on the ICP algorithm. Thus, the main steps are:

1. closest point matching;
2. transformation estimation (by minimizing the point-to-plane distance);
3. iteration on 1. and 2. until the convergence is reached.

However a multi-resolution scheme is introduced. To do so, the points are stored in octrees (the data structure is described in section 1.2.3). An illustration of the hierarchical levels generated

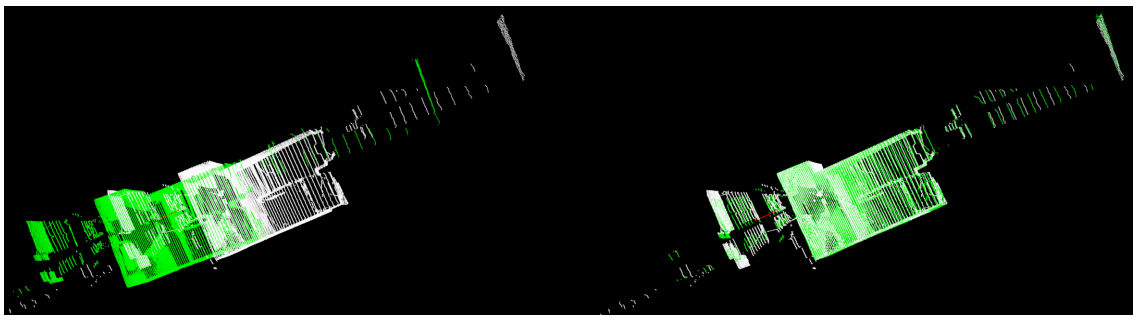


Figure 4.1 – Registration of scans 6 and 7 from *Stairs* sequence in ASL dataset using GNMR-ICP. In green ${}^{s}\mathbf{P}$ the source point cloud, in white ${}^{t}\mathbf{P}$ the target point cloud. - *Left*: before registration. - *Right*: after registration.

by the octree is given in Figure 4.2: a point cloud captured in a square room is stored in an octree of resolution 0.01m. For each subdivision, the stored node is the centroid of the cube. This can be exploited by using these centroids in a new point cloud, which represent the same environments but with fewer data. We end up with a hierarchy of point clouds, from coarse to fine resolution. The point cloud on the left is the one with the coarsest resolution, and the one on the right, the one with the finest resolution.

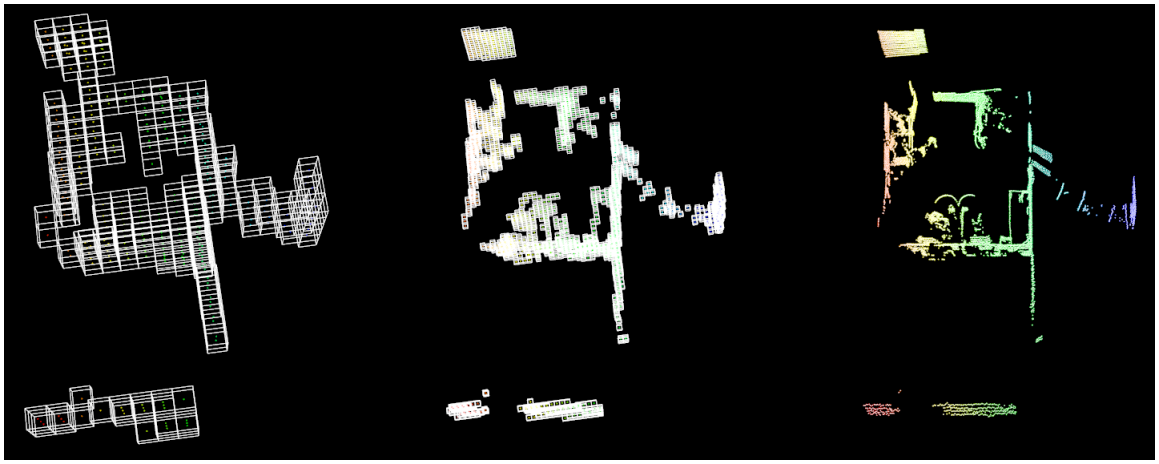


Figure 4.2 – Representation of the hierarchical levels generated by the octree of a point cloud. *Left:* resolution 0.64m - *Center:* resolution 0.16m - *Right:* resolution 0.01m

With GNMR-ICP, after the octrees are constructed from source and target, and the new point clouds extracted, the ones with the coarsest resolution (containing a few points) are registered using the point-to-plane distance minimization described further in section 4.3.4 with the Gauss-Newton method. The fact that these two point clouds are composed of a small number of points makes the minimization faster at this level, and gives a good initialization

for the next level of resolution. Thus, the finest resolution level, which should be the most time consuming if a lot of iteration is needed, ends up to be initialized close to the optimal solution, decreasing the number of iterations needed to converge. The global framework of the proposed method is described in Figure 4.3.

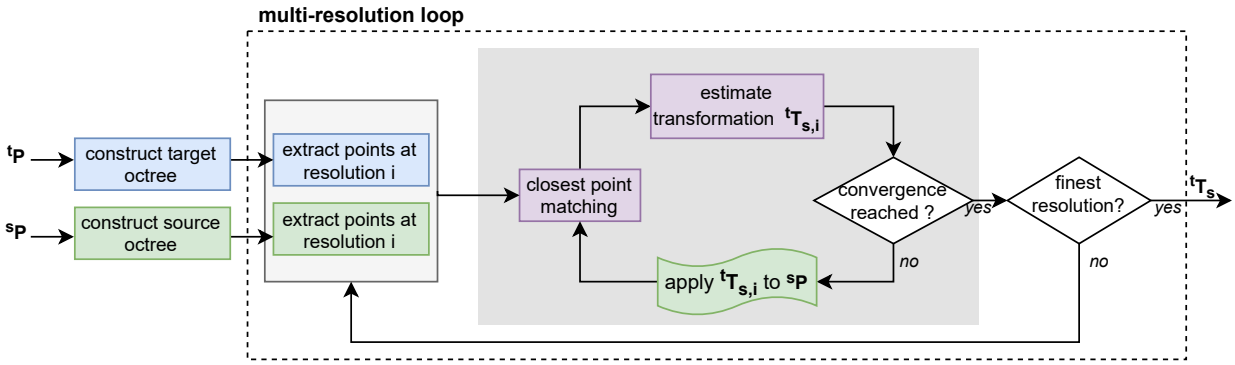


Figure 4.3 – GNMR-ICP framework

Each step of the framework is described further in this section:

— the normal estimation is outlined in section 4.3.2;

— the point matching is described in section 4.3.3;

— the robust point-to-plane distance minimization is detailed in section 4.3.4.

4.3.2 Normal estimation

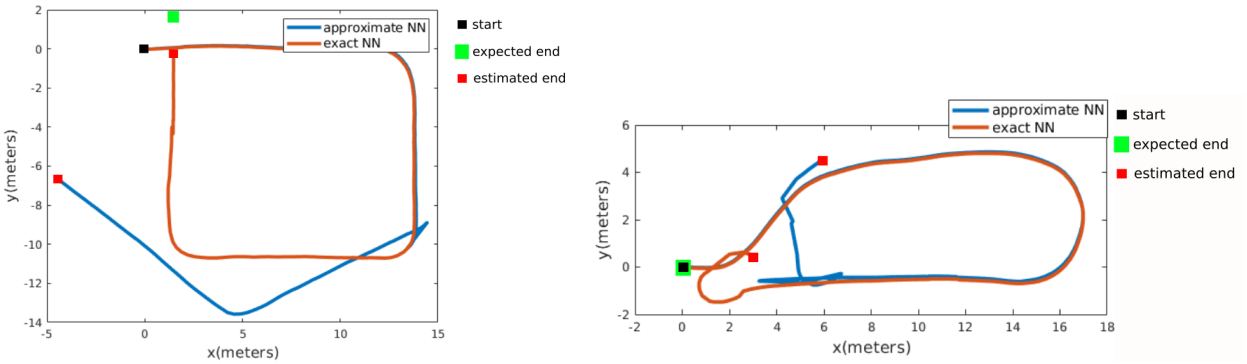
GNMR-ICP aims to estimate the transformation between two point clouds by minimizing the point-to-plane distance $d^{\perp}$. This implies to estimate the normal at the considered points in the target point cloud. To do so, a PCA is performed with the points in the neighborhood of the one we are interested in. The explanation of the functioning of the PCA was provided in section 1.3.2. First, the normals at each target point are estimated with the original target point cloud, considering the k nearest neighbors (with $k = 50$ in our case) of the target points. Then, to compute the normals relative to an octree level, the normals that are the closest to the centroids in the original point clouds are taken.

4.3.3 Point Matching

To estimate the point-to-plane distance, the points of the source point clouds need to be matched with their correspondent in the target point cloud. However, the true correspondences

are not known. Thus, for each source point, the nearest target point is chosen as the correspondent. There are several techniques to find the nearest neighbor of a point in a point cloud. One can look for the exact nearest neighbor of a point, which is the closest point according to Euclidean point-to-point distance. An other technique, which uses the advantages of an octree, is the approximate search: instead of looking for the exact closest point, the point which is the closest in the voxel of the query point will be chosen. This point might not be the closest point, but it is less time consuming than an exact nearest neighbor search. It is a trade off between time and accuracy.

In Figure 4.4 one can see the impact of the choice of the point matching method on the long sequences of LOOP'IN dataset presented in section 3.4. With LOOP'IN, the expected goal for the registration algorithm is to close the loops. In both cases the trajectories begin smoothly and follow the same track. But, once an error is made, using approximate nearest neighbor, the influence of the wrongly registered points cannot be compensated and, in this cases, we can see that GNMR-ICP cannot recover from these mistakes. Either way, the expected goal of LOOP'IN is not reached.



(a) *Balcony Loop* (b) *Coffee Hall Loop*

Figure 4.4 – Influence of nearest neighbor search method on LOOP'IN dataset on the estimation of the trajectories with GNMR-ICP. Axes are in meters.

4.3.4 Point-to-plane distance minimization

As stated previously, the point-to-plane distance $d^{\perp}$ defined in equation (4.4) is minimized thanks to a Gauss-Newton approach. The estimation is made robust by the use of M-estimators.

4.3.4.1 Gauss-Newton approach

We defined in section 1.4.1 a minimal representation of the rigid transformation ${}^t\mathbf{T}_s$, $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^T$ where θ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^t\mathbf{R}_s$. We have to minimize the point-to-plane distance $d^\perp$ (eq. 4.4) by minimizing the related error:

$$\mathbf{e}(\mathbf{q}) = \mathbf{n}^\top \cdot (\mathbf{p}(\mathbf{q}) - \mathbf{p}) \tag{4.12}$$

with $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s\, {}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$, $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$ and $\mathbf{n} = (\dots, {}^t\mathbf{n}_i, \dots)$. This is a nonlinear least-squares minimization problem and can be solved using the Gauss-Newton algorithm presented in section 1.4.2.2. Solving it consists in minimizing the cost function $E(\mathbf{x}) = \|\mathbf{e}(\mathbf{q})\|$ The derivation of this minimization problem is given in section 1.4.2.2, with the Jacobian corresponding to $d^\perp$:

$$\mathbf{J} = \begin{pmatrix} \vdots & \vdots \\ -{}^t\mathbf{n}_i^\top & {}^t\mathbf{n}_i^\top \left[{}^s\overline{\mathbf{p}}_i\right]_\times \\ \vdots & \vdots \end{pmatrix} \tag{4.13}$$

The solution is given by:

$$\delta\mathbf{q} = -\lambda \mathbf{J}^+ \mathbf{e}(\mathbf{q}) \tag{4.14}$$

where λ is a coefficient in $]0, 1]$ and $\mathbf{J}^+$ is the pseudo-inverse of the $N \times 6$ Jacobian $\mathbf{J}$. The pose is then updated at each iteration:

$$\mathbf{q}_{k+1} = \mathbf{q}_k \oplus \delta\mathbf{q} = \exp^{\delta\mathbf{q}} \mathbf{q} \tag{4.15}$$

where $\oplus$ denotes the composition over $\mathfrak{se}(3)$ obtained via the exponential map.

4.3.4.2 M-estimators

To make the minimization robust and reduce the influence of outliers on the estimation, M-estimators functions are used. An explanation of how to use M-estimators in optimization problems is given in section 1.4.2.3. As a reminder, the error vector in equation (4.12) becomes:

$$\mathbf{e}_\rho(\mathbf{q}) = \mathbf{D}(\mathbf{e}(\mathbf{q})) \tag{4.16}$$

with $\mathbf{D}$ the $N \times N$ diagonal matrix containing the weights that reflect the confidence in the data. The solution becomes:

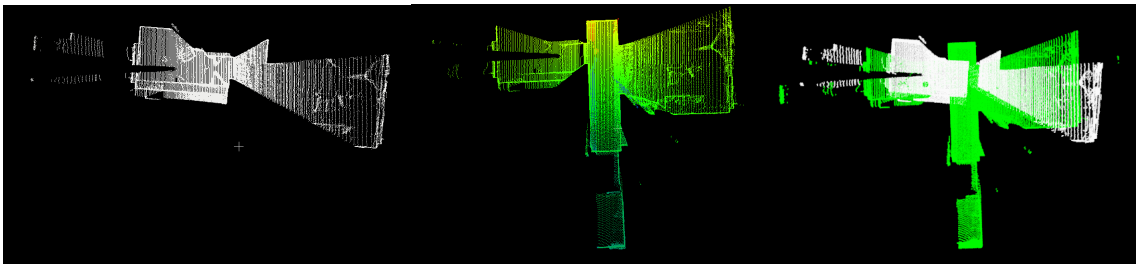
$$\delta\mathbf{q} = -\lambda(\mathbf{DJ})^{+}\mathbf{De}(\mathbf{q}) \tag{4.17}$$

Of the various influence functions that exist in the literature we consider Huber's, Tukey's and Cauchy's M-estimators defined in Table 4.1. We refer back to the work of [Malis et al. 2006] for the parametrization choices.

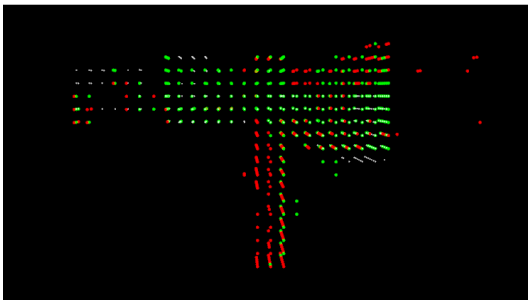
Table 4.1 – Description of M-estimators robust influence functions and their associated weights

Type	Conditions	$\rho(e_i)$	$w(e_i)$
Huber	$\begin{cases} \text{if } \lvert e_i\rvert \leq k \\ \text{otherwise} \end{cases}$	$\begin{cases} \frac{e_i^2}{2} \\ k\left(\lvert e_i\rvert - \frac{k}{2}\right) \end{cases}$	$\begin{cases} 1 \\ \frac{k}{\lvert e_i\rvert} \end{cases}$
Tukey	$\begin{cases} \text{if } \lvert e_i\rvert \leq k \\ \text{otherwise} \end{cases}$	$\begin{cases} \frac{k^2\left(1-\left(1-\left(\frac{e_i}{k}\right)^2\right)^3\right)}{6} \\ \frac{k^2}{6} \end{cases}$	$\begin{cases} \left(1-\left(\frac{e_i}{k}\right)^2\right)^2 \\ 0 \end{cases}$
Cauchy		$\frac{k^2}{2}\log\left(1+\left(\frac{e}{k}\right)^2\right)$	$\frac{1}{1+\left(\frac{e_i}{k}\right)^2}$

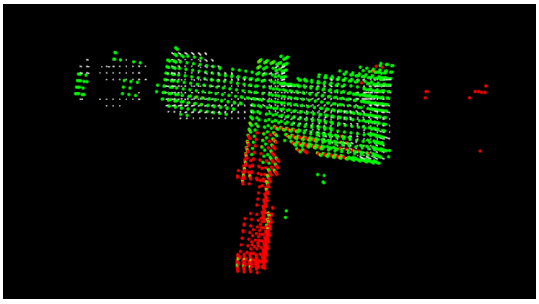
In Figure 4.5 we present two point clouds to register from the *Apartment* sequence of the ASL dataset. In this example, the two point clouds to register have a small overlap: in the source (Fig 4.5 (b)), a corridor can be seen, which is not present in the target (Fig 4.5 (a)). The influence of the points that materialize this corridor would make the estimation fail if no robust function was used. In Figure 4.5 (d)(e)(f)(g)(h) we show how points are rejected by the M-estimators during the estimation at each resolution level of GNMR-ICP. On the figure, one can clearly see that those points, among others, are identified by the M-estimators (in red). Thanks to this, their influence on the estimation problem will be canceled leading to a successful registration.



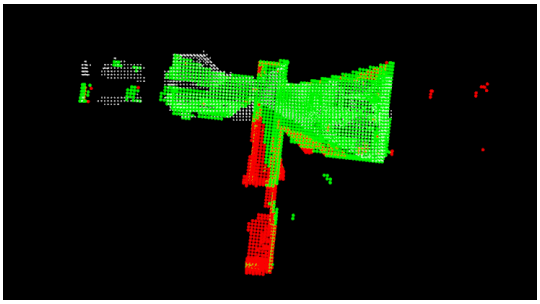
(a) Target point cloud ${}^{t}\mathbf{P}$. (b) Source point cloud ${}^{s}\mathbf{P}$. (c) ${}^{t}\mathbf{P}$ and ${}^{s}\mathbf{P}$ before registration.



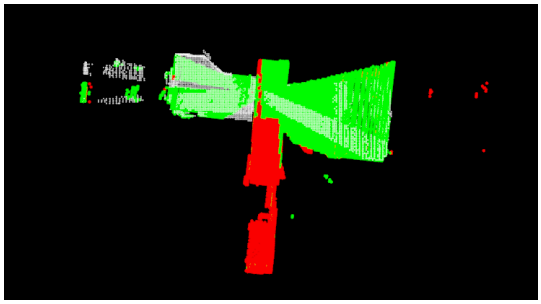
(d) Registration at first resolution.



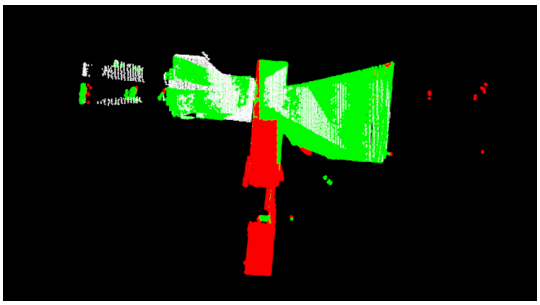
(e) Registration at second resolution.



(f) Registration at third resolution.



(g) Registration at fourth resolution.



(h) Registration at fifth resolution.

Figure 4.5 – Point rejection with M-estimators in GNMR-ICP. *In green*: ${}^{s}\mathbf{P}$ the source point cloud - *In white*: ${}^{t}\mathbf{P}$ the target point cloud - *In red*: point rejected by M-estimators.

4.4 Experiments and discussions

In the following section, two experiments are presented. The first experiment illustrates the influence of the number of levels used in the multi-resolution scheme on accuracy and computation time. The second experiment compares GNMR-ICP to the multi-resolution version of the small angle ICP as well as state-of-the-art algorithms regarding their accuracy on indoor sequences.

To assess accuracy, the sequences of ASL datasets are used in the scheme depicted in section 3.3. The quality of the registration is evaluated by computing the Euclidean distance Δ_t for translation and geodesic distance Δ_r for rotation of the estimated transformation to the ground truth provided in the dataset thanks to eq. (3.1) and eq. (3.2). The computed distances need to be smaller than the chosen error thresholds t_{tr} and t_{rot} to be considered successful [Magnusson et al. 2015], with:

— $t_{tr} = 0.1m$

— $t_{rot} = 2.5°$

This can be performed because the ASL dataset comes with the ground truth of the poses of the sensor. For more details about this dataset, please refer to section 3.3.

4.4.1 Influence of the number of levels on accuracy and processing time

This experiment aims to show the impact of the number of levels in a multi-resolution scheme on processing time and accuracy.

The studied algorithm is GNMR-ICP, using Huber's M-estimator as it demonstrated to be slightly more accurate among others. The octree resolution is set to 3cm. The algorithm is run on each sequence, and the number of multi-resolution levels is changed, from 1 (in other words, the Gauss-Newton point-to-plane ICP algorithm without multi-resolution, but with a subampling grid generated by the octree with a resolution of 3cm) to 6 levels.

A summary presenting the percentage of successful registration for each sequence and each number of levels used is given in Table 4.2. The best results are highlighted in green. Globally, the more levels are used, the better the algorithm performs (e.g. 7% success with 1 level on Mountain sequence to 93% with 5 levels). It is interesting to notice that, excepting the *Stairs* sequence, the 1-level GNMR-ICP performs poorly in comparison with the others. With this dataset, the 5-level GNMR-ICP is the one providing the best results: it succeeds in registering the entirety of *ETH*, *Stairs* and *Wood* sequences (according to t_{tr} and t_{rot}).

With *Gazebo* and *Mountain* sequences, it also gives the best results with respectively 90% and 93% of success. For *Apartement* sequence, the 3-level GNMR-ICP is the one providing the best results (with 93% of successful registration). It can also be noticed that using the 6-level GNMR-ICP deteriorates the performances of the algorithm as it gives less accurate result than the 5-level one. But the number of the levels to use seems to have a limit as in this example we observed a deterioration of the results by using the 6-level GNMR-ICP.

Table 4.2 – Percentage (%) of successful registration for different number of levels with GNMR-ICP.

Sequence	1 level	2 levels	3 levels	4 levels	5 levels	6 levels
Apartment	68	88	93	88	82	86
ETH	58	100	100	100	100	100
Gazebo	67	73	87	87	90	87
Mountain	7	41	62	93	93	87
Stairs	90	93	90	93	100	96
Wood	43	83	93	96	100	93

Regarding processing time, only the estimation step of the algorithm is taken into account. The experiment is processed on a desktop computer with an Intel Xeon W-2133, 3.6GHz CPU and 32GB RAM. A graph summarizing mean processing time, on each sequence for each number of levels used in the multi-resolution scheme, is given in Figure 4.6. On *Apartment*, *ETH*, *Gazebo*, *Stairs* and *Mountain* sequences, the trend is that the more levels are used, the less time is needed for the estimation. Here, the 4-level GNMR-ICP performs the fastest. It shows one of the main interest of using multi-resolution: the coarsest level is composed of a few points, which makes the estimation at this level fast. Then, the estimations at finer levels, by being close to the expected solution require only a few iteration, which takes less time. Whereas, the classical ICP needs to make its estimation considering all the points at once, leading to more iterations to converge. At 5 and 6 levels processing time is slightly increasing. On *Apartment* sequence, processing time is reduced by a factor of 2 by using 4 levels of resolution instead of 1. With *Stairs* sequence, the factor is about 1.8. It is interesting to notice that the sequences giving the best decreasing factor are the more structured ones. Both are composed of large planar structures. On the other hand, *Wood* sequence, which is the less structured one, is the only sequence which does not follow the rule of time decreasing. It is most likely that the multi-resolution scheme shows more interest in structured environment as it allows to reduce redundant information (such as points on a plane) without degrading it

too much (in terms of geometrical shapes).

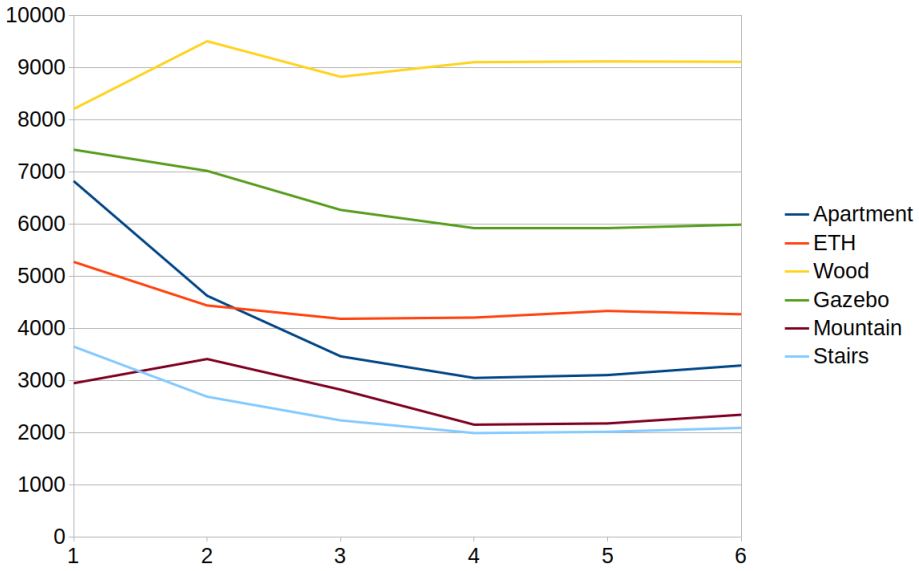


Figure 4.6 – Influence of the number of multi-resolution levels on processing time. - *Horizontal axis*: the number of levels used in the multi-resolution scheme. - *Vertical axis*: mean processing time in milliseconds.

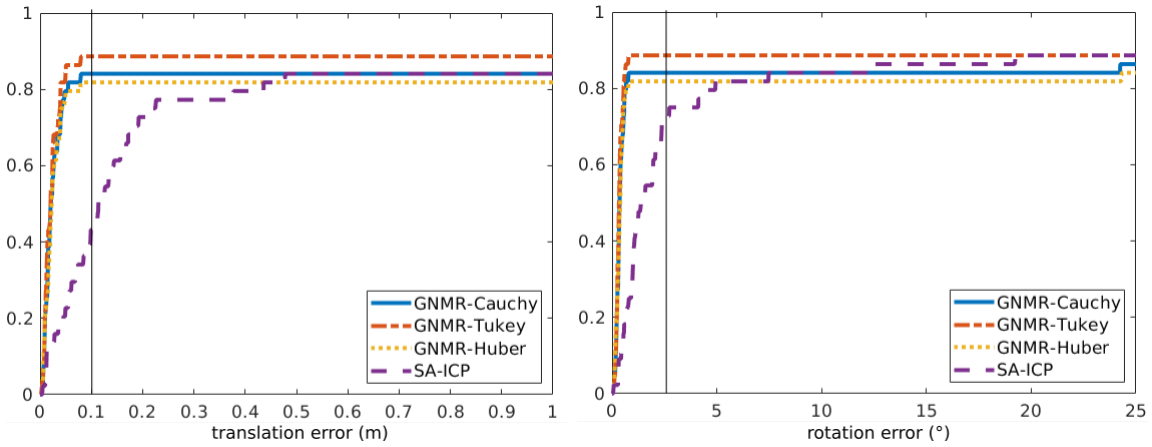
These experiments showed that using multi-resolution generates more accurate pose estimation. Using only 1 level gave poor accuracy results, while using 5 of them led to significantly more successful registrations. More over, we saw on *Apartment* and *Stairs* sequences a significant improvement in computation time. Those sequences are the most structured of all the provided ones. Whereas, the most unstructured sequence, *Wood* was not affected by the time reduction.

4.4.2 Accuracy analysis

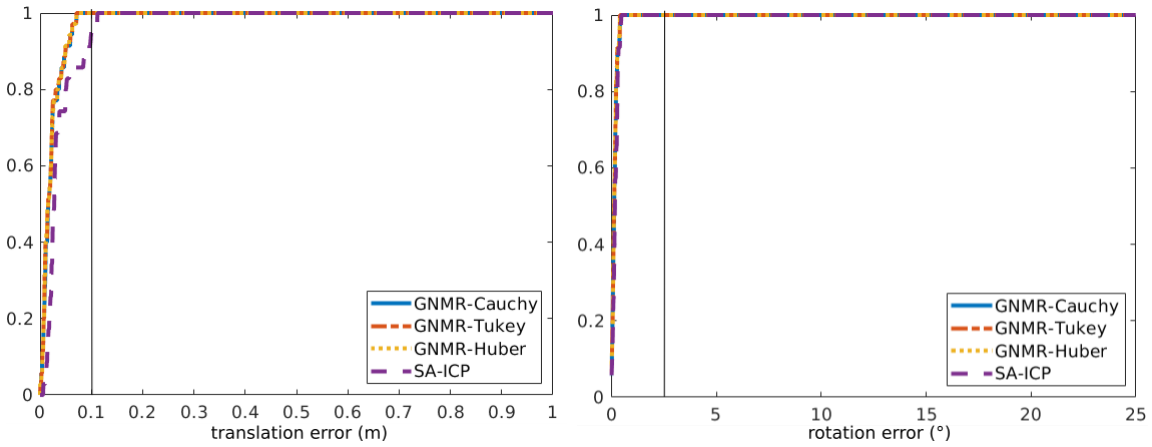
Accuracy analysis of multi-resolution methods

In this experiment the accuracy of both GNMR-ICP and small angle point-to-plane ICP (further denoted SA-ICP) in its multi-resolution form are compared. To do so, the point clouds of the sequences are registered pairwise, thanks to the chosen algorithms. The number of levels to consider,for both algorithms, is set to 5, and the octree resolution is set to $3cm$. SA-ICP is granted a maximum distance filter to remove point correspondences that are too far from each other. The maximum distance is set to $1m$.

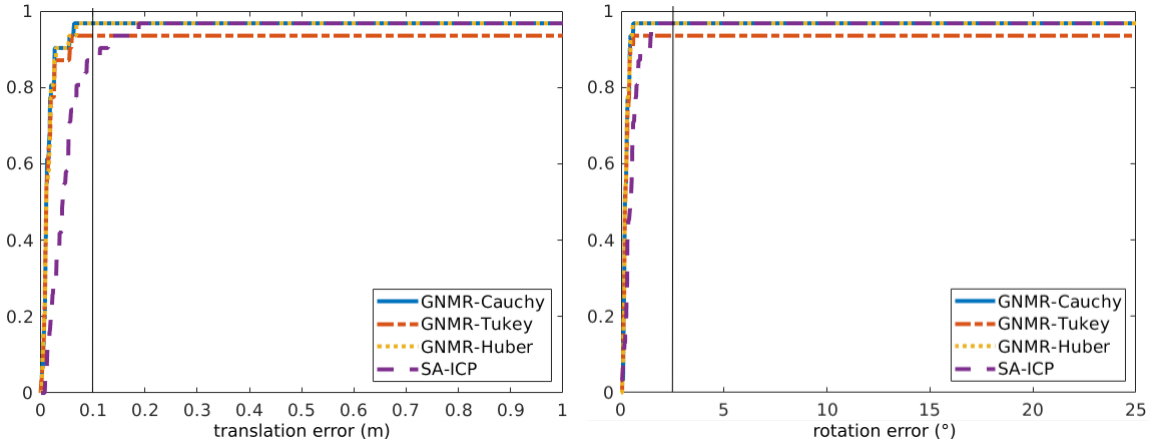
Translation and rotation errors for GNMR-ICP (considering the three types of M-estimators) and SA-ICP are depicted in Fig. 4.7 and Fig. 4.8. *Note that rotation and translation errors are presented separately but a result is valid only if both rotation and translation errors are smaller than respectively* t_{tr} *and* t_{rot}.



(a) *Apartment*

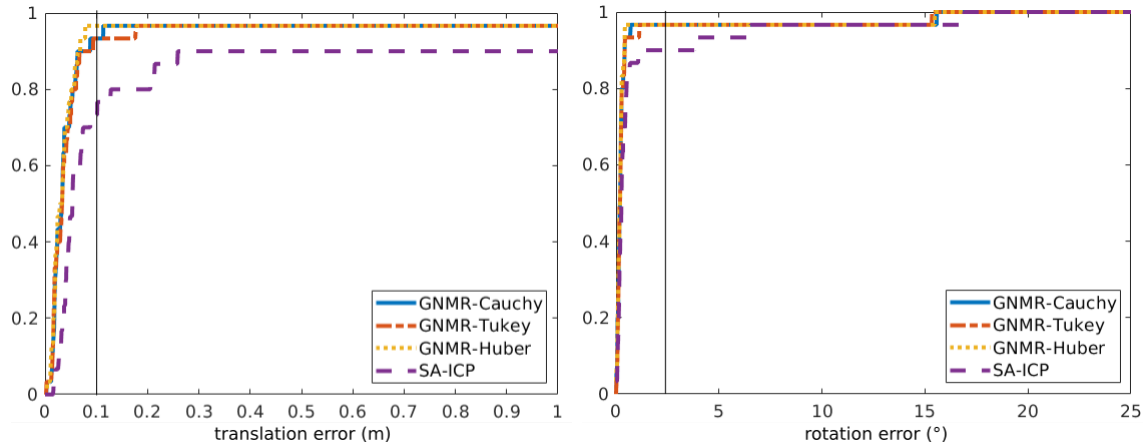


(b) *ETH*

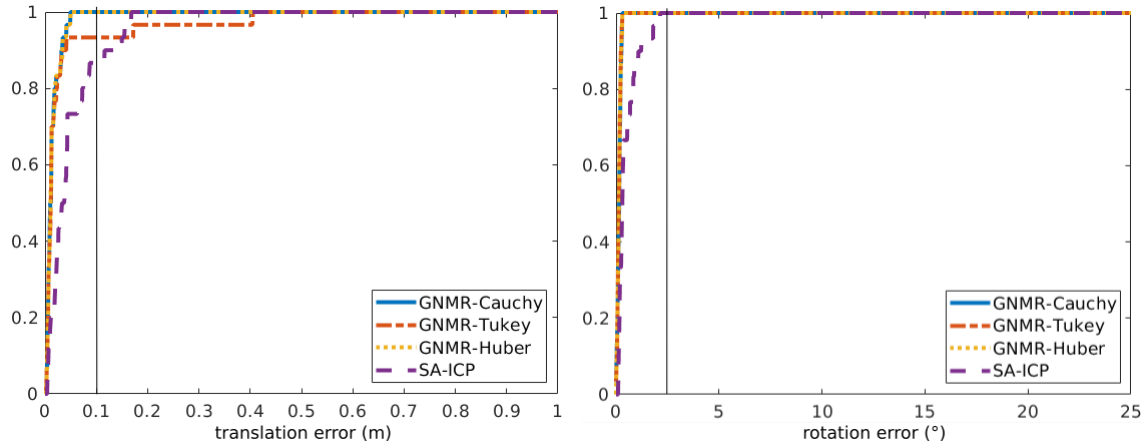


(c) *Gazebo*

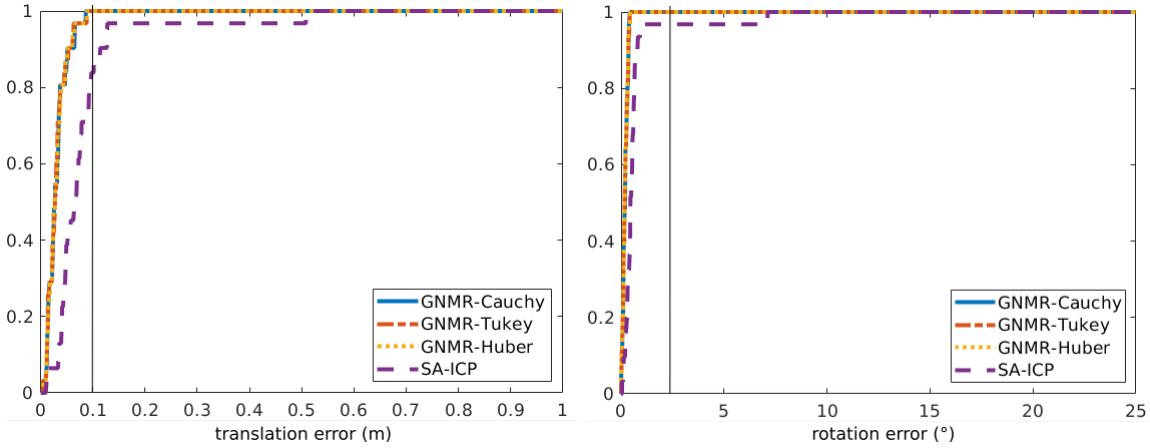
Figure 4.7 – Cumulative probabilities on translation and rotation error on *Apartment*, *ETH*, and *Gazebo* sequences of the ASL dataset. *Left*: translation error (in meters). - *Right*: rotation error (in degrees). - *Vertical axis*: cumulative probability. - *Horizontal axis*: the error. The thresholds t_{tr} and t_{rot} are materialized with bars at, respectively, 0.1m and 2.5°.



(a) *Mountain plain*



(b) *Stairs*



(c) *Wood*

Figure 4.8 – Cumulative probabilities on translation and rotation error on *Mountain plain*, *Stairs*, and *Wood* sequences of the ASL dataset. *Left*: translation error (in meters). - *Right*: rotation error (in degrees). - *Vertical axis*: cumulative probability. - *Horizontal axis*: the error. The thresholds t_{tr} and t_{rot} are materialized with bars at, respectively, 0.1m and 2.5°.

The curves represent cumulative probabilities of the errors. As explained in section 3.3, the more top-left the curve the better the algorithm performs. The expected behavior is to attain 1, meaning all scans of the sequence are registered before the error threshold is reached. If so, it means that 100% of the scans of the sequence are successfully registered (according to the chosen t_{tr} and t_{rot}). If 1 is not reached before the threshold, it means that the registration error is too large to be considered successful. On the plots, one can see that globally, SA-ICP performs less accurately than the Gauss-Newton based ICP algorithms for translation and rotation. For instance, considering the *Apartment* sequence, only 43% of the translation are estimated accurately for SA-ICP, whereas for the GNMR-ICP algorithms, the one giving the worst result (Huber) reaches 81% of successful estimation in translation. *Apartement* is the sequence including the more challenging rotations in all the dataset, explaining why the difference is so significant in this case. Nevertheless, the trend is the same with the other sequences: SA-ICP gives less accurate results. About the Gauss-Newton based algorithms, besides some exceptions, their results considering accuracy are very similar. The GNMR-ICP using Tukey's M-estimators is the one giving results slightly less accurate (except for the *Apartment* sequence where it performs the best with 88% successful registration in translation whereas Cauchy's and Huber's give respectively 84% and 81%).

A summary presenting the percentage of successful registration (for both translation and rotation) for each sequence and each algorithm is given in Table 4.3. The best results are highlighted in green. As stated previously, Gauss-Newton based ICP performs better than the one using the small angle approximation. The GNMR-ICP using Huber's M-estimators performs slightly better than the others.

Table 4.3 – Percentage (%) of successful registration on ASL dataset (regarding t_{tr} and t_{rot}).

Sequence	SA-ICP	GNMR-ICP$_{Huber}$	GNMR-ICP$_{Tukey}$	GNMR-ICP$_{Cauchy}$
Apartment	43	82	88	84
ETH	94	100	100	100
Gazebo	87	90	94	97
Mountain	73	93	93	93
Stairs	87	100	93	100
Wood	84	100	93	100

Now, considering only successful registrations, it is possible to evaluate the accuracy of the registration. Once more, the Euclidean and geodesic distance from the ground truth are observed. The results are detailed in Table 4.4, giving the mean errors in translation and rotation for each tested algorithm, considering only the successful registrations. The worst results are highlighted in red. For each sequence, SA-ICP turns out to be the most distant to the ground truth, both for translation and rotation error. Regarding the Gauss-Newton based ICP algorithms, their accuracy values are very similar, with a millimeter difference.

Table 4.4 – Mean translation and rotation error for successful registration for each M-estimator and SA-ICP on ASL dataset

	Translation error (m)				Rotation error (°)			
	SA-ICP	GNMR-ICP			SA-ICP	GNMR-ICP		
		Huber	Tukey	Cauchy		Huber	Tukey	Cauchy
Apartment	0.049	0.023	0.022	0.023	0.75	0.35	0.34	0.36
ETH	0.033	0.022	0.022	0.022	0.17	0.14	0.14	0.14
Gazebo	0.040	0.016	0.015	0.015	0.44	0.24	0.24	0.24
Mountain	0.047	0.033	0.034	0.033	0.21	0.19	0.19	0.19
Stairs	0.033	0.015	0.013	0.015	0.4	0.15	0.14	0.15
Wood	0.08	0.031	0.03	0.031	0.68	0.2	0.2	0.2

As expected, the small angle multi-resolution ICP algorithm performs less accurately than the Gauss-Newton based algorithms. Considering GNMR-ICP algorithms, globally, Huber's M-estimators gives a better rate of successful registration and Tukey's gives more accurate results, but the differences is not significant enough to tell whether Cauchy's, Tukey's or Huber's M-estimators performs the best in these scenarios.

Accuracy comparison with state of the art algorithms

In the following experiment, GNMR-ICP is compared to three state-of-the-art registration algorithms in terms of accuracy on the three indoor sequences of the ASL dataset. The chosen algorithms are GICP [Segal et al. 2009], NDT [Magnusson et al. 2015], and the point-to-plane ICP with the Point Cloud Library (PCL) implementation.

— GICP has three major parameters. Maximum iterations is set to 10, Euclidean fitness epsilon is set to 10^{-6} and maximum correspondence distance is set to 0.8m as set in [Zong et al. 2019].

— NDT, with the steps recommended in [Magnusson et al. 2015]. Transformation epsilon is set to 10^{-3}, step size 0.1, maximum iteration 5, first step resolution 1.0m, second step resolution 2.0m, third step resolution 1.0m and last step 0.5m.

— Point-to-plane ICP (denoted ICP-PCL), similarly to GICP has three major parameters. Maximum iterations is set to 100, Euclidean fitness epsilon is set to 10^{-6} and maximum correspondence distance is set to 0.8m.

The success rates are summarized in Table 4.5.

Table 4.5 – Percentage of successful registration (translation and rotation combined) for the evaluated algorithms on each sequence.

Sequence	GICP	NDT	ICP-PCL	GNMR-ICP
Apartment	75	77	43	82
ETH	100	100	100	100
Stairs	97	97	90	100

The proposed algorithm GNMR-ICP gives a higher success rate than the evaluated state of the art algorithms. It is able to register 100% of both *ETH* and *Stairs* sequences while giving a success rate of 82% for the *Apartement* sequence which is composed of large rotation motion.

4.5 Conclusion

In this chapter, a method performing 3D point cloud registration by using a variation of the classical ICP algorithm based on point-to-plane distance minimization is presented. This method is using a multi-resolution scheme based on octrees. The point-to-plane minimization problem is solved using a Gauss-Newton approach. It is made robust by using M-estimators.

In registration problematics, closed-form minimization methods are sometimes preferred to iterative methods as they are less time consuming. A closed-form solution is used in [Low 2004] and [Vlaminck et al. 2017] for point-to-plane distance optimization based on the small angle approximation. The aim of the algorithm proposed in this chapter is to perform accurate registration by avoiding such an approximation by using an iterative method. The multi-resolution scheme not only aims to handle the data more wisely, it also allows to reduce processing time.

The first experiment aimed to show the influence of the number of levels used in the multi-resolution scheme on accuracy and processing time, with the GNMR-ICP. The algorithm is used, first, without multi-resolution, then, with 2 to 6 levels of resolution. The accuracy is, once more, assessed using the ASL dataset. It turns out that using more than 1 level of resolution leads to more accurate results. The experiment also demonstrates that, in this scenario, there is a limit on the number of levels to use, as 6 levels gave less accurate results than 5. It also shows that in structured environments, the use of a multi-resolution scheme tends to decrease processing time. This trend appears more obviously in scenarios including planar structures (implying redundant information on points). In the case of unstructured data (more specifically the *Wood* sequence), no influence on time is observed.

The second experiment compared the accuracy of three Gauss-Newton based multi-resolution ICP using different M-estimators with the algorithm proposed in this chapter, GNMR-ICP, and the small angle multi-resolution variation used in [Vlaminck et al. 2017] (SA-ICP) as well as state of the art algorithms on the ASL [Pomerleau et al. 2012] dataset. As stated previously, this dataset comes with the ground truth, allowing to compute the translation and rotation error of the issued estimations. As expected, the GNMR-ICP algorithms perform better than the SA-ICP. The three variations generate more successful estimation than the small angle variation. More over, considering only the successful registration, the Gauss-Newton based algorithms also give more accurate estimations. About the choice of the M-estimators, the difference in terms of accuracy is not significant enough to tell whether one is performing better than the others. Huber's M-estimators gives slightly better results in this scenario. Regarding the other state of the art algorithms, GNMR-ICP gives higher success rates on the indoor sequences.

The contribution of the use of a multi-resolution scheme on a point-to-plane non-linear ICP algorithm is demonstrated on accuracy and processing time. However, it shows its limits on unstructured environments. To assess the robustness of the proposed method, it would be interesting to perform similar experiments on datasets using different types of LiDARs. Moreover, this method is a good candidate for GPU acceleration, thus it could be interesting to do such an implementation and observe the impact it has on processing time. This would allow to combine computational efficiency and accuracy.

These experiments showed that GNMR-ICP tends to decrease processing time, nevertheless, the gain is not satisfying enough. The number of points to consider to perform the minimization is still high, leading to a process that stays time consuming compared to other state-of-the-art algorithms, as can be seen in Figure 4.9 on *Apartment* sequence. Even if

processing time is still high, we can note that GNMR-ICP ensures a high accuracy.

The experiments brought to the fore the presence of redundant information on planar structures. Using planes instead of points can tremendously reduce the dimensionality of the minimization problem, leading to smaller processing time. Therefore, the following chapter will present a registration method based on plane-to-plane registration.

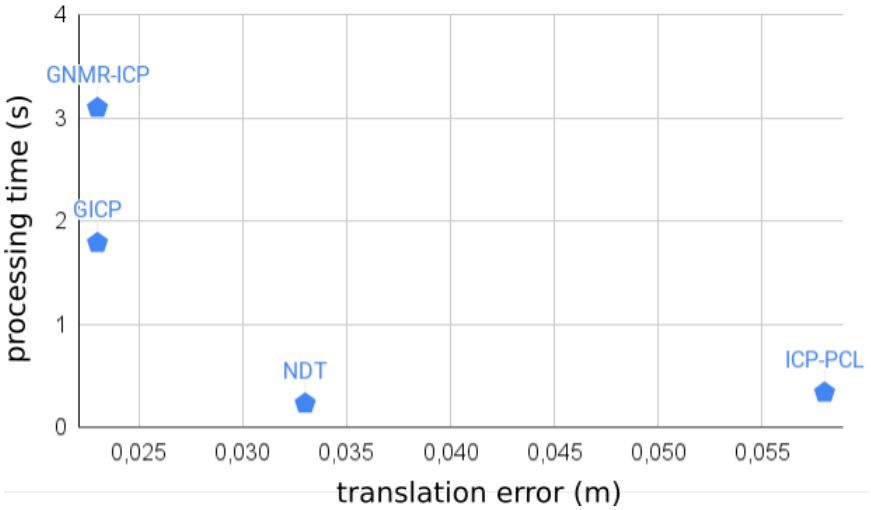


Figure 4.9 – Translation error (m) / processing time (s) for GNMR-ICP and state-of-the-art algorithms on Apartment sequence.

Chapter 5

Plane-based registration of 3D point clouds

5.1 Introduction

In the previous chapter, we showed that using large sets of points in registration problem can be time consuming. But, we also saw that points belonging to the same planar structure are providing redundant information, leading to the main idea of this chapter to use planes instead of points. Man-made environments are usually well-structured, which implies the presence of strong planar structures. Using planes instead of points can tremendously reduce the dimensionality of the registration problem.

In this chapter, we present our plane-based registration method, New Accurate Plane-based ICP (NAP-ICP). First, a state of the art related to plane-to-plane distance minimization, plane segmentation and plane matching is given. In a third section, our plane-based registration algorithm is described in details. Finally, the impact of the registration steps of the proposed algorithm as well as its accuracy and computational efficiency are assessed through experiments.

5.2 State of the art

The goal of the plane-to-plane registration is to register two point clouds, a source ${}^{s}\mathbf{P}$ to its target ${}^{t}\mathbf{P}$. Using planes implies, plane segmentation and plane matching. In this section, we will present related state-of-the-art algorithms.

5.2.1 Plane-to-plane distance formulation

We presented in section 2.2.1 the formulation of the plane-to-plane distance $\mathbf{d}_i^{\Pi}$ as proposed by [Grant et al. 2019]. We remind that $\mathbf{d}_i^{\Pi}$ is defined as follows:

$$\mathbf{d}_i^{\Pi} = \begin{pmatrix} {}^t\mathbf{R}_s {}^s\mathbf{n}_i - {}^t\mathbf{n}_i \\ \left[{}^t\mathbf{R}_s {}^s\mathbf{n}_i\right]^\top {}^t\mathbf{t}_s + {}^s\rho_i - {}^t\rho_i \end{pmatrix} \tag{5.1}$$

with ${}^s\mathbf{n}_i$ and ${}^t\mathbf{n}_i$ respectively the normals of source and target planes, ${}^s\rho_i$ and ${}^t\rho_i$ the distance of, respectively, source and target planes to the origin, ${}^t\mathbf{R}_s$ and ${}^t\mathbf{t}_s$ the rotation and translation linking source to target point clouds. Considering $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^T$ the minimal representation of ${}^t\mathbf{T}_s$, where θ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^t\mathbf{R}_s$, the plane-to-plane error has to be minimized such that:

$$\hat{\mathbf{q}} = \underset{\mathbf{q}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathbf{d}_i^{\Pi} \tag{5.2}$$

which consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$, with:

$$\mathbf{e}(\mathbf{q}) = \begin{pmatrix} \mathbf{n}(\mathbf{q}) - \mathbf{n} \\ \rho(\mathbf{q}) - \rho \end{pmatrix} \tag{5.3}$$

with $\mathbf{n}(\mathbf{q}) = (..., {}^t\mathbf{R}_s {}^s\mathbf{n}_i, ...)$, $\mathbf{n} = (..., {}^t\mathbf{n}_i, ...)$, $\rho(\mathbf{q}) = (..., \left[{}^t\mathbf{R}_s {}^s\mathbf{n}_i\right]^\top {}^t\mathbf{t}_s + {}^s\rho_i, ...)$ and $\rho = {}^t\rho_i$.

5.2.2 Plane-to-plane distance minimization

Minimizing the plane-to-plane distance is a non-linear least squares problem. In [Zong et al. 2019] the transformation is estimated using a method based on the SVD. They introduce a normalization operation in order to improve the registration estimated by the SVD. The original points are first translated so that their centroid is at the origin and then scaled by a factor so that the average distance from the origin is equal to $\sqrt{3}$.

In the insert below called **Plane-to-Plane registration by [Taguchi et al. 2013]**, the derivation used in [Taguchi et al. 2013] using the SVD, which will be used further in our minimization process, is provided.

In [Grant et al. 2019], IC3PO, when enough plane correspondences are available, a closed-form solution is used to solve the translation (which is decoupled from rotation). The rotation is solved using a quaternion-based solution derived by [Davenport 1968] called the "q-method". When plane correspondences are not sufficient, points are added to the optimization prob-

lem which is then solved using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) quasi-Newton method [Wright et al. 1999].

Plane-to-Plane registration by [Taguchi et al. 2013]

The rotation and translation are decoupled. For the rotation we need to minimize:

$$\sum_{i=1}^{n} \| {}^{t}\mathbf{R}_{s} \, {}^{s}\mathbf{n}_{i} - {}^{t}\mathbf{n}_{i} \| \tag{5.4}$$

For translation, a linear constraint is added such that we minimize:

$$\sum_{i=1}^{N} \| ({}^{t}\mathbf{R}_{s} \, {}^{s}\mathbf{n}_{i})^{\top} \, {}^{t}\mathbf{t}_{s} + {}^{s}\rho_{i} - {}^{t}\rho_{i} \| \tag{5.5}$$

First, to find ${}^{t}\mathbf{R}_{s}$, as in [Arun et al. 1987] and [Lorusso et al. 1995], a 3×3 correlation matrix $\mathbf{H}$ is built such as:

$$\mathbf{H} = \sum_{i=1}^{N} {}^{s}\mathbf{n}_{i} \, {}^{t}\mathbf{n}_{i}^{\top} \tag{5.6}$$

Its SVD decomposition ($\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}$) allows to compute the optimal rotation matrix ${}^{t}\hat{\mathbf{R}}_{s}$ by:

$${}^{t}\hat{\mathbf{R}}_{s} = \mathbf{V}\mathbf{U}^{\top} \tag{5.7}$$

The translation constraint in equation (5.5) corresponds to a linear system of type $\mathbf{A} \, {}^{t}\mathbf{t}_{s} = \mathbf{b}$. where

$$\mathbf{A} = \begin{pmatrix} ({}^{t}\mathbf{R}_{s} \, {}^{s}\mathbf{n}_{1})^{\top} \\ \vdots \\ ({}^{t}\mathbf{R}_{s} \, {}^{s}\mathbf{n}_{N})^{\top} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} {}^{t}\rho_{1} - {}^{s}\rho_{1} \\ \vdots \\ {}^{t}\rho_{N} - {}^{s}\rho_{N} \end{pmatrix}$$

The least-squares solution of this problem (refer to section 1.4.2.1 for more details) is given by:

$${}^{t}\hat{\mathbf{t}}_{s} = \mathbf{A}^{+}\mathbf{b} \tag{5.8}$$

To get a plane-to-plane distance, we need to extract planes in each point cloud and to match them. The two following subsections are dedicated to the state of the art related to those issues.

5.2.3 Plane segmentation

Approaches using planes are interesting to use in man-made environments. However, segmenting planes can be time consuming. In [Poppinga et al. 2008] and [Pathak et al. 2009], the 3D data from the sensor are used as range images. The neighborhood relation of the pixels is then used to segment the planes in a region growing scheme. It also includes a polygonalization of the planes in surface models.

In [Rabbani et al. 2006], a region growing process based on smoothness is introduced in order to segment planes. This method can struggle with the sparseness of points.

In [Fischler et al. 1981],[Pathak et al. 2010] and [Taguchi et al. 2013], RANSAC approaches are used in order to fit points to planar patches. As this method is based on a RANSAC method, when too many points are considered, it can significantly affect processing time.

In [Grant et al. 2013] a Hough-based method is used: the used rotating LiDAR forms cones in space, leading to conic section on planar surfaces. Then, each conic section found in a scanline of a sensor sweep votes for all possible planes that could have produced it. The votes are accumulated for each conic section. Once all voting has been completed, the accumulator is thresholded to find candidate planes.

5.2.4 Plane matching

Matching planes after the segmentation is another challenging task. With MUMC [Pathak et al. 2010], the goal is to find the set of patch correspondences, in source and target point clouds, that gives the most geometrically consistent 3D transformation.

In [Chen et al. 2020], a plane/line descriptor is proposed to establish structure correspondences. This descriptor is in the form of a vector, and correspondences are computed thanks to the Euclidean distance. The two closest descriptors are matched together.

In [Zong et al. 2019] they state that using the plane parameters obtained thanks to the segmentation may not be highly accurate and propose, a combination of plane characteristics to estimate plane correspondences: it includes surface area, centroid of the planes, boundary information and a so-called minimum bounding rectangle. The pairs of planes are then established through similarity metrics they propose, such as shape difference and area ratio.

In [Grant et al. 2019], the data rate acquisition is supposed very high, which leads to low relative translation between scans. Thus, planes with the projections of the origin of the sensor close to each other and almost parallel are matched. This is not applicable to large motion scenarios.

5.3 Our plane-to-plane registration NAP-ICP

After the presentation of the state of the art related to the plane-to-plane distance problematic, this section is dedicated to our contribution NAP-ICP algorithm.

5.3.1 Positioning

As our data is composed of many points, we chose to segment planes using a region growing approach to keep it fast enough. As we saw previously, finding plane correspondences is a challenging problem, so we decided to propose a solution based on plane features. To register planes, we chose to adopt a two-step scheme: first, the transformation is estimated using a closed-form minimization of the plane-to-plane distance. Using planes means having few inputs (in comparison with points), which gives us the possibility to robustify this closed-form minimization with a RANSAC algorithm. The estimation is then refined using a Gauss-Newton approach. For the sake of accuracy, the transformation estimation is completed with a point-to-plane registration additional step. As a matter of fact, using planes with parameters estimated thanks to the region growing algorithm tends to smooth the original data and approximate it. This additional step aims to correct this approximation and refine the estimation.

5.3.2 Overall description

The reader has to note that with this method, the area corresponding to the set of points composing the planes is taken into account, so we should use the term of planar patches instead of planes, but in order to ease reading, the term planes will be preferred in the following sections even if the area is used.

The framework of NAP-ICP is given in Fig. 5.1. Similarly to the classical ICP algorithm the method iteratively performs the matching step and the minimization step. However in the proposed method the first features to be matched are planes. Once matched, the rigid transformation minimizing the plane-to-plane distance is estimated. After the plane-to-plane registration is performed, an additional point-to-plane registration is done. An example of registration using NAP-ICP is given in Fig. 5.2.

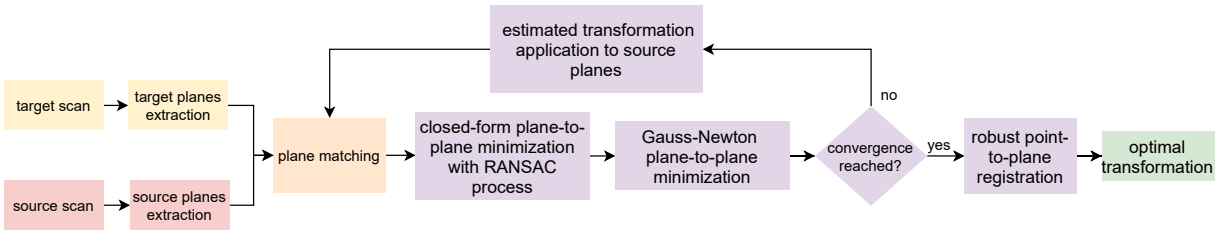


Figure 5.1 – Overview of NAP-ICP.

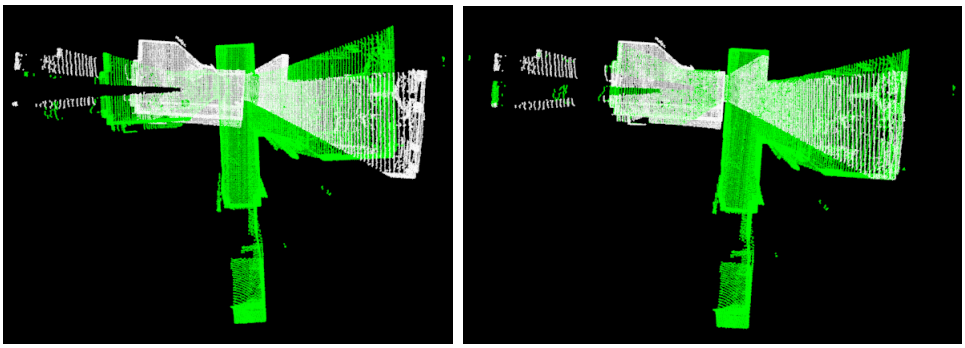


Figure 5.2 – Example of the registration between two point clouds (scans 3 and 4 from *Apartment* sequence from ASL dataset [Pomerleau et al. 2012]) using NAP-ICP. The overlap between scans is small, yet the proposed method succeeds in registering the two point clouds accurately. In white the target point cloud - In green the source point cloud. *Left:* before registration - *Right:* after registration.

Each step of the framework is described further in this section:

— planes segmentation is described in section 5.3.3;

— the score metric for finding best planes correspondences is detailed in section 5.3.4;

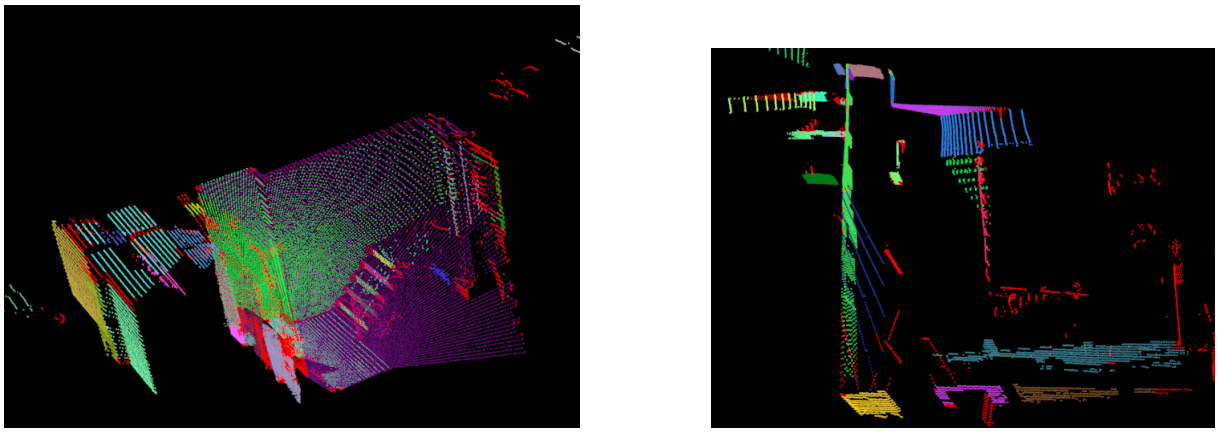
— robust plane-to-plane registration is described in section 5.3.5;

— the additional point-to-plane minimization leading to finer registration is detailed in section 5.3.5.

5.3.3 Plane segmentation in NAP-ICP

In order to segment the planes of the point cloud we chose the region growing method as in [Rabbani et al. 2006]. As stated in section 5.2.3, with this method, the points in a neighborhood with a small angle difference between normals are considered to be on the same surface. Thus, the first step is to estimate the normals at each point of the point cloud. To do so, the PCA algorithm (see section 1.3.2 for more details) is used within the neighborhood

of each point. Once it is done, the clusters are built using the region growing approach. A few examples of plane segmentation can be seen in Figure 5.3 on point clouds from ASL and LOOP'IN datasets.



(a) *Stairs* (b) *Balcony Loop*

Figure 5.3 – Plane segmentation using region growing approach. Each extracted plane is in a different color. Red points are outliers, i.e. they belong to no plane.

Now that the plane segmentation is done, for each plane, its corresponding plane parameters can be estimated thanks to a RANSAC algorithm. These parameters are ρ the distance to the origin of the plane and $\mathbf{n}$ its normal.

Moreover, the surface of the plane, denoted S, is computed. To do so, thanks to the implementation of the PCL, a convex hull (Fig. 5.4) is built for each plane and directly gives us the area of the convex hull.

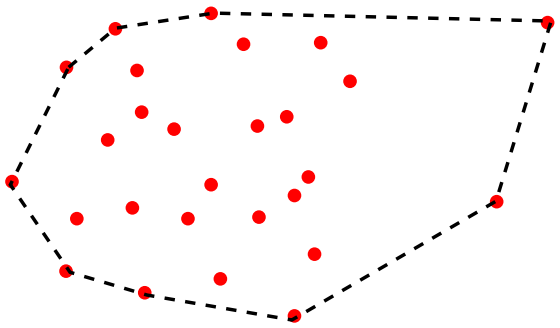


Figure 5.4 – Convex hull of a set of points

Finally, the centroid of the plane is computed by finding the centroid of the set of points it is built from.

The reason why these additional features are computed is discussed further in section 5.3.4.

5.3.4 Plane matching in NAP-ICP

Once the planes are segmented, the next step is matching each source plane to the closest one in the target point cloud. For each extracted plane ${}^{s}\mathbf{\Pi}_i$ in the source, a list of planes in the target that are potential matches is made. Each target candidate ${}^{t}\mathbf{\Pi}_j$ is given a score within the range $[0; 1]$. This score is computed thanks to several plane correspondences features. Most of them assume a small relative motion between point clouds. In [Zong et al. 2019] the centroids of the planes are used to calculate its minimum bounding rectangle. We chose to use them in order to compute the distance d_c, representing the distance between the centroids of paired planes. Corresponding planes might not have the same shape from one point cloud to the next because of occlusion, thus using the centroids might not always be the best choice. Hence, as in [Grant et al. 2013], we also chose to match pairs of planes using the distance between the projections of the origin d_o. The dot product of the normals of the planes ϕ_n is also computed. Similarly to [Zong et al. 2019], the area ratio of the paired planes is considered.

The features are computed as follows:

— the distance between the projections of the origin on source plane and target plane d_o, assuming low relative motion, is expected to be close to 0:

$$d_o = \|{}^{s}\rho_i {}^{s}\mathbf{n}_i - {}^{t}\rho_j {}^{t}\mathbf{n}_j\| \tag{5.9}$$

— the distance between the centroids of source and target planes d_c, assuming low relative motion, is expected to be close to 0:

$$d_c = \|{}^{s}\mathbf{c}_i - {}^{t}\mathbf{c}_j\| \tag{5.10}$$

with ${}^{s}\mathbf{c}$ and ${}^{t}\mathbf{c}$ the centroids of ${}^{s}\mathbf{\Pi}_i$ and ${}^{t}\mathbf{\Pi}_j$ respectively;

— the area ratio between the planes S_r, expected to be close to 1 as the planes are expected to have similar areas:

$$S_r = \frac{min({}^{s}S_i, {}^{t}S_j)}{max({}^{s}S_i, {}^{t}S_j)} \tag{5.11}$$

with ${}^{s}S_i$ and ${}^{t}S_j$ the area of source and target planes respectively;

— the dot product of the normals of the planes ϕ_n, when assuming small rotation the planes are expected to be almost parallel, so the dot product is expected to be close

to 1:

$$\phi_n = {}^s\mathbf{n}_i \cdot {}^t\mathbf{n}_j \tag{5.12}$$

Each feature is remapped between $[0; 1]$, 0 corresponding to the minimum value the feature could have and 1 the maximum, and weighted, which leads to a score defined as follows:

$$score = \alpha \cdot \hat{d}_o + \beta \cdot \hat{d}_c + \gamma \cdot (1 - \hat{S}_r) + \delta \cdot (1 - \hat{\phi}_n) \tag{5.13}$$

with $\hat{.}$ denoting the rescaled value, and the weights α, β, γ and δ subject to:

$$\alpha + \beta + \gamma + \delta = 1 \tag{5.14}$$

Parameters α, β, γ, δ were chosen empirically to fixed values for all experiments such as: $\alpha = 0.35$, $\beta = 0.4$, $\gamma = 0.1$ and $\delta = 0.15$.

For these features, we can apply the three following conditions to validate the matching candidates:

— the score of the matched planes has to be smaller than a threshold:

$$score < \epsilon_{score} \tag{5.15}$$

— matched planes with their centroid being too far from each other need to be discarded:

$$d_c \ > \ \epsilon_{centroid} \tag{5.16}$$

— matched planes with a notable difference in area are discarded:

$$S_r \ > \ \epsilon_S \tag{5.17}$$

The valid pairs of correspondent planes form the list of correspondences between source and target. Using this function, there is no guarantee that the best plane correspondences are always the one with the smallest score. So, all correspondences respecting the previous conditions are kept, including ones that might be wrong. This may also include several occurrences of the same source plane, with different target planes and vice-versa. This problem is fixed by using a robust estimation method presented further in section 5.3.5.

5.3.5 Plane-to-plane registration

Now that the set of plane correspondences is built, the rotation and translation parameters can be estimated by minimizing the plane-to-plane distance $\mathbf{d}_i^{\Pi}$ such as:

$$\mathbf{d}_i^{\Pi} = \begin{pmatrix} {}^t\mathbf{R}_s\, {}^s\mathbf{n}_i - {}^t\mathbf{n}_i \\ \left[{}^t\mathbf{R}_s\, {}^s\mathbf{n}_i\right]^\top {}^t\mathbf{t}_s + {}^s\rho_i - {}^t\rho_i \end{pmatrix} \tag{5.18}$$

Robust plane-to-plane initialization

To perform a closed-form minimization of the plane-to-plane distance we used the derivation of [Taguchi et al. 2013] presented in section 5.2.1.

As the previously created correspondence list may contain outliers, it is important to discard them as they can lead to divergence in the minimization step. To do so, a RANSAC process is applied to make the minimization more robust. Only three non-parallel planes are needed in the source and target respectively as a minimal set of data for rotation and translation estimation. Each sample of the RANSAC algorithm is selected respecting this condition. Through experiments, we consider that the correspondence list is composed of 70% valid data. Using the equation from section 1.4.2.3 to compute the number of iterations needed to obtain a sample composed of only inliers, we find that at least 7 iterations are needed in our RANSAC process to be sure to find at least 95% of inliers.

Gauss-Newton plane-to-plane minimization

The plane correspondences identified as inliers by the RANSAC process are given as input of the Gauss-Newton approach. This method requires a minimal representation of ${}^t\mathbf{T}_s$ the transformation to be estimated. Again, a vector denoted $\mathbf{q} = ({}^t\mathbf{t}_s, \theta\mathbf{u})^\top$ is used, where θ and $\mathbf{u}$ are the angle and the axis of the rotation ${}^t\mathbf{R}_s$. The plane-to-plane error is minimizing using a Gauss-Newton approach. Solving it consists in minimizing the cost function $E(\mathbf{q}) = \|\mathbf{e}(\mathbf{q})\|^2$:

$$\mathbf{e}(\mathbf{q}) = \begin{pmatrix} \mathbf{n}(\mathbf{q}) - \mathbf{n} \\ \rho(\mathbf{q}) - \rho \end{pmatrix} \tag{5.19}$$

with $\mathbf{n}(\mathbf{q}) = (..., {}^t\mathbf{R}_s\, {}^s\mathbf{n}_i, ...)$, $\mathbf{n} = (..., {}^t\mathbf{n}_i, ...)$, $\rho(\mathbf{q}) = (..., \left[{}^t\mathbf{R}_s\, {}^s\mathbf{n}_i\right]^\top {}^t\mathbf{t}_s + {}^s\rho_i, ...)$ and $\rho = {}^t\rho_i$ the error vector of the distance between the target point cloud and the source point cloud transformed with the previous estimated transformation. The $4N \times 6$ Jacobian matrix

associated with the plane-to-plane distance is $\mathbf{J}$:

$$\mathbf{J} = \begin{pmatrix} \vdots & \vdots \\ \mathbf{0}_{3\times3} & [{}^t\mathbf{R}_s\,{}^s\mathbf{n}_i]_\times \\ -[{}^t\mathbf{R}_s\,{}^s\mathbf{n}_i]^\top & \mathbf{0}_{1\times3} \\ \vdots & \vdots \end{pmatrix} \tag{5.20}$$

Point-to-plane registration

To ensure an accurate registration, a refinement step is added to find the best expected rigid transformation. To do so a point-to-plane registration is added at the end of the process. A robust non-linear minimization of the point-to-plane distance similar to the one presented in section 4.3.4 is performed. Each source point is matched to its closest target point according to the Euclidean distance. Then the rigid transformation that registers source to target point cloud is computed by minimizing the point-to-plane distance $d^\perp$ (equation (2.2.1)) with the related error vector:

$$\mathbf{e}(\mathbf{q}) = \mathbf{n}^\top \cdot (\mathbf{p}(\mathbf{q}) - \mathbf{p}) \tag{5.21}$$

with $\mathbf{p}(\mathbf{q}) = (\dots, {}^t\mathbf{R}_s\,{}^s\overline{\mathbf{p}}_i + {}^t\mathbf{t}_s, \dots)$, $\mathbf{p} = (\dots, {}^t\overline{\mathbf{p}}_i, \dots)$ and $\mathbf{n} = (\dots, {}^t\mathbf{n}_i, \dots)$. Its $N \times 6$ Jacobian $\mathbf{J}$ is defined by:

$$\mathbf{J} = \begin{pmatrix} \vdots & \vdots \\ -{}^t\mathbf{n}_i^\top & {}^t\mathbf{n}_i^\top\,[{}^s\overline{\mathbf{p}}_i]_\times \\ \vdots & \vdots \end{pmatrix} \tag{5.22}$$

Again, the influence of the outliers coming from wrongly matched points is reduced using M-estimators.

5.4 Experiments and discussions

In the following section, three experiments are presented. The first one depicts the impact of the registration steps of NAP-ICP on the accuracy of the results. The second one compares the accuracy of NAP-ICP with state of the art registration algorithms. The third one compares the same algorithms in terms of computation time. The last one evaluates the sensitivity of NAP-ICP to drift on long sequences.

Once more, to assess accuracy, the sequences of ASL dataset are used in the scheme

presented in section 3.3. To do so, the point clouds of the sequences are registered pairwise, thanks to the chosen algorithms. As the proposed algorithm is designed to register planar surfaces, the choice was made to consider only the indoor sequences of the dataset. Using NAP-ICP on unstructured data would make no sense, thus, only *Apartment*, *ETH* and *Stairs* sequences are used. As explained in section 3.3, the estimation of the performed registration is evaluated by computing the distance of the estimated transformation to the ground truth provided by the dataset using Δ_t and Δ_r. The computed distances need to be smaller than chosen thresholds [Magnusson et al. 2015] (t_{tr} for translation and t_{rot} for rotation) already used in Chapter 4, with:

— $t_{tr} = 0.1m$

— $t_{rot} = 2.5°$

5.4.1 Impact of the registration steps of NAP-ICP on accuracy

This experiment aims to show the need for two main steps in NAP-ICP:

— first: the RANSAC algorithm used in the closed-form minimization to make it robust;

— second: the additional point-to-plane registration minimization step.

Impact of the RANSAC algorithm on minimization

In section 5.3.5 we saw that NAP-ICP uses a RANSAC approach in order to remove outliers from the correspondence list generated by the score function. To show the need for this choice, in this experiment NAP-ICP is run considering only the best correspondences for each source plane (the plane getting the smallest score value). In figure 5.5 (and summarized in Table 5.1) we can see, as expected, that the version of NAP-ICP using RANSAC outperforms the one without RANSAC in terms of accuracy. Considering that when using planes, only a few data are used in the minimization process, one wrong correspondence can lead to a wrong estimation, so it is obvious that outliers need to be removed.

Impact of the two-step minimization scheme

As said previously, NAP-ICP performs the transformation estimation in two successive steps. First a plane-to-plane minimization is done followed by a point-to-plane registration. In Fig. 5.6, the impact of these steps can be observed with curves representing the cumulative probabilities on translation errors and rotation errors. The percentages of successful registration are summarized in Table 5.2. Globally, one can observe that for each sequence, the plane-to-plane registration gives a good initialization of the rigid transformation but is still far from ground

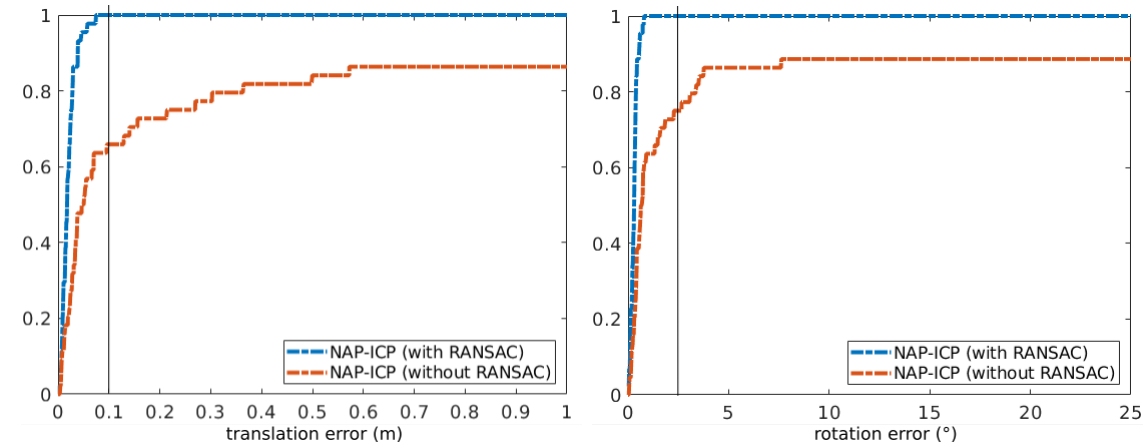
Table 5.1 – Percentage of successful registration (both for translation and rotation) for NAP-ICP **with and without RANSAC.**

Sequence	NAP-ICP without RANSAC	NAP-ICP with RANSAC
Apartment	66	100
ETH	80	100
Stairs	87	100

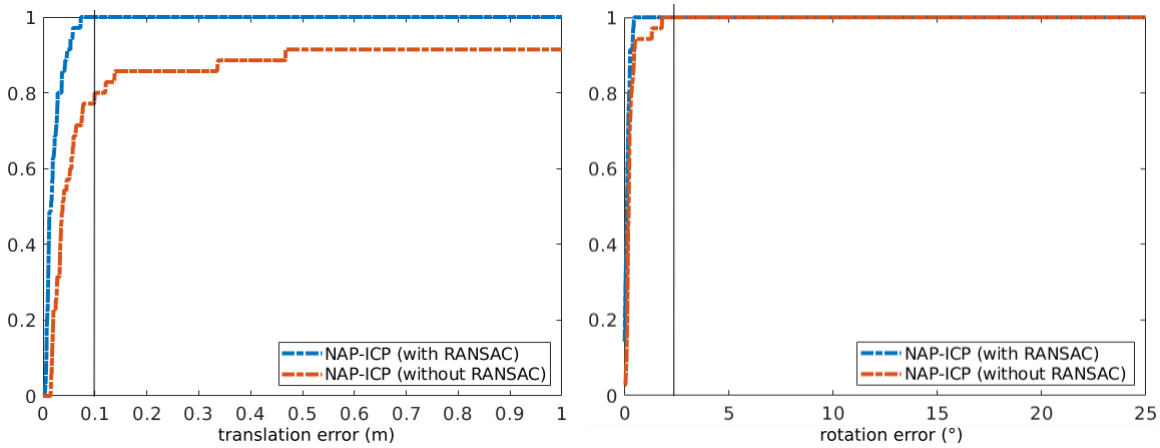
truth, whereas plane-to-plane with point-to-plane reaches 100% for both rotation and translation. For instance, considering the *Apartment* sequence, only 36% scans are well registered regarding translation error (81% success rate in rotation). On *ETH* and *Stairs* (which are sequences not subject to large rotations), regarding rotation, even if the plane-to-plane registration gives results sufficient to reach the expected threshold (100% for *ETH* and 97% for *Stairs*), the addition of the point-to-plane proves to give a more accurate estimation. By comparison, for all tested sequences, the point-to-plane registration step addition achieves a 100% success rate in both rotation and translation. This proves the ability of the plane-to-plane minimization to give a result close enough to what is expected in order to obtain an accurate registration with a robust point-to-plane registration.

Table 5.2 – Percentage of successful registration (both for translation and rotation) for NAP-ICP **with and without point-to-plane registration addition**

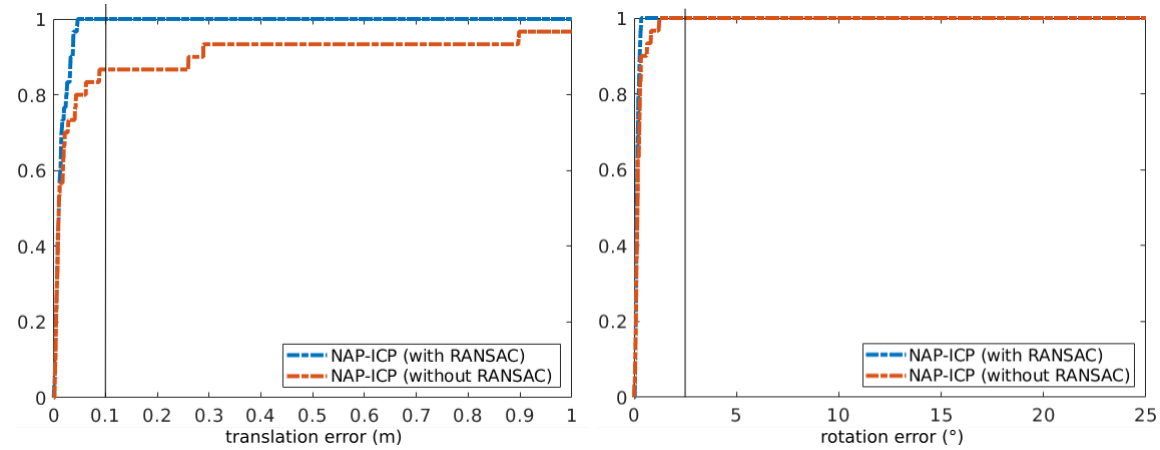
Sequence	NAP-ICP without point-to-plane	NAP-ICP with point-to-plane
Apartment	36	100
ETH	37	100
Stairs	70	100



(a) *Apartment*

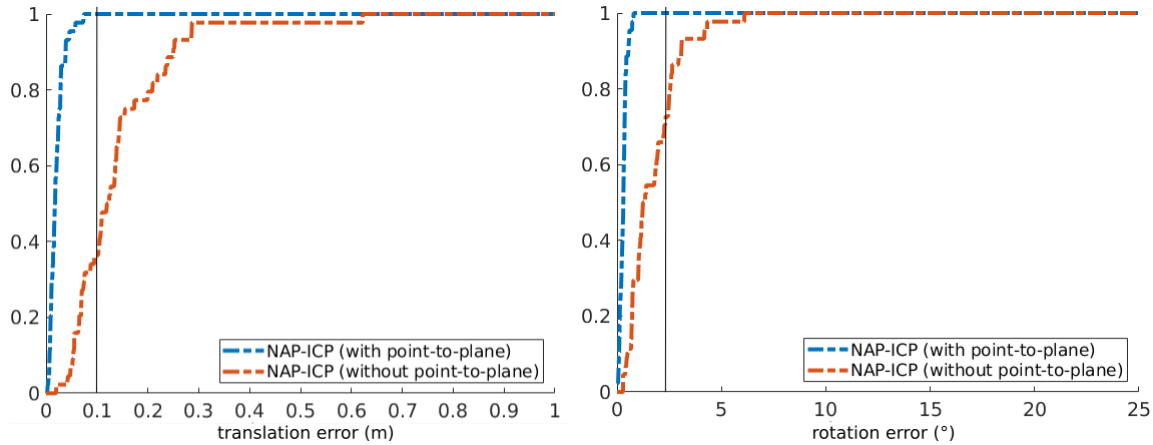


(b) *ETH*

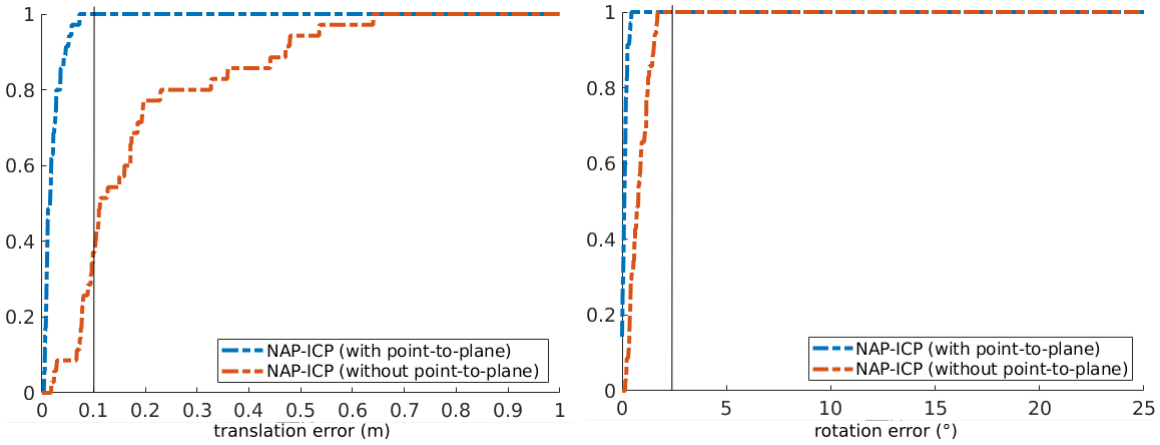


(c) *Stairs*

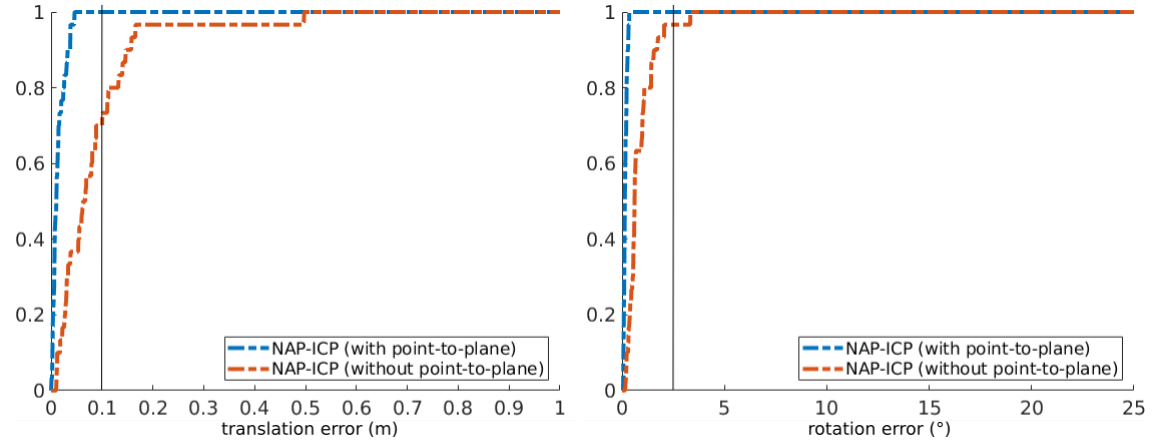
Figure 5.5 – Cumulative probabilities of translation and rotation errors of **NAP-ICP with and without RANSAC**. *Left:* translation error (in meters) on the horizontal axis. The vertical bar represents the threshold t_{tr}. *Right:* rotation error (in degrees) on the horizontal axis. The vertical bar represents the threshold t_{rot}.



(a) *Apartment*



(b) *ETH*



(c) *Stairs*

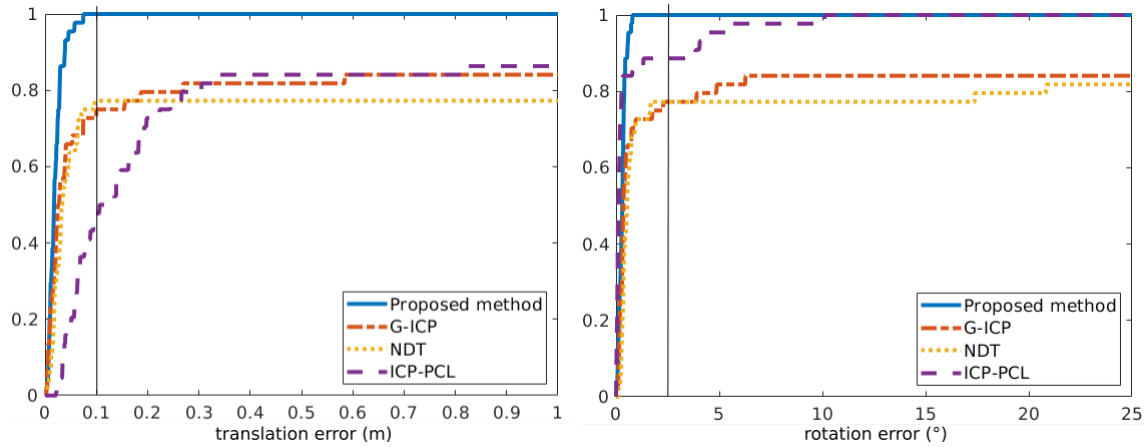
Figure 5.6 – Cumulative probabilities of translation and rotation errors of NAP-ICP **with and without point-to-plane registration addition**. *Left:* translation error (in meters) on the horizontal axis. The vertical bar represents the threshold t_{tr}. *Right:* rotation error (in degrees) on the horizontal axis. The vertical bar represents the threshold t_{rot}.

5.4.2 Accuracy comparison with state-of-the-art algorithms

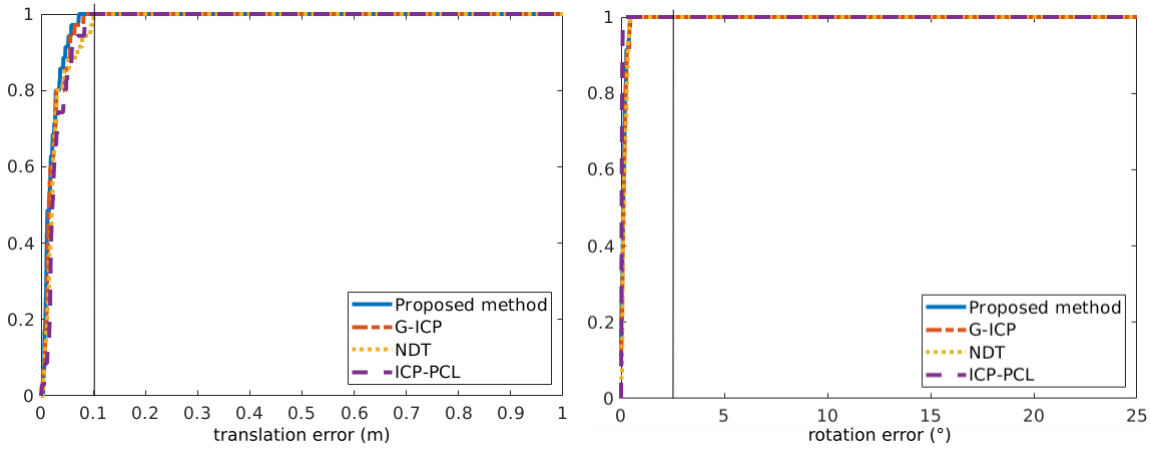
In the following experiment, NAP-ICP is compared to three state-of-the-art registration algorithms in terms of accuracy on the three indoor sequences of the ASL dataset. NAP-ICP is compared to the same algorithms as in Chapter **??**: GICP ([Segal et al. 2009]), NDT ([Magnusson et al. 2015], and the point-to-plane ICP. The parametrization used in section 5.4.2 is used. The cumulative probabilities for translation and rotation errors are given in Figure 5.7 and the success rates are summarized in Table 5.3.

Table 5.3 – Percentage of successful registration (translation and rotation combined) for the evaluated algorithms on each sequence.

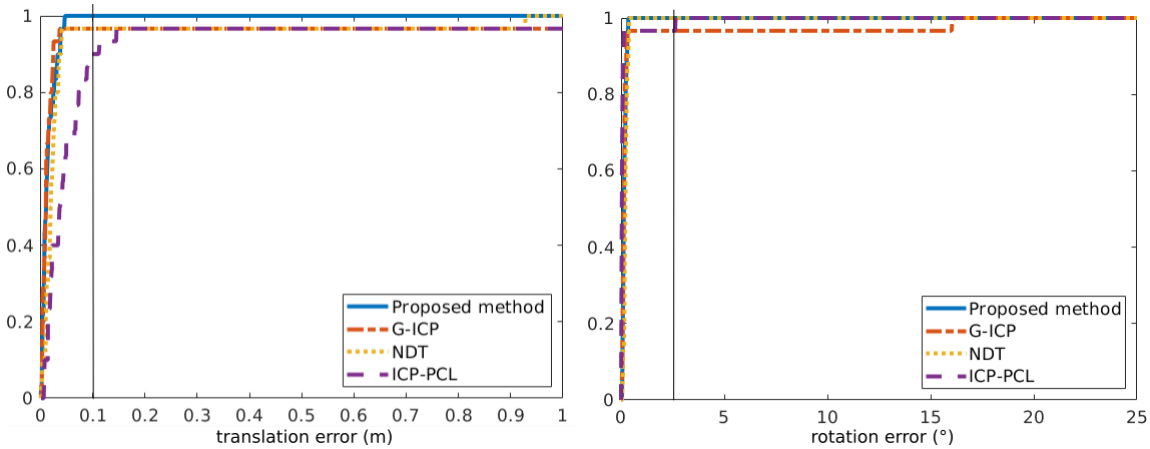
Sequence	GICP	NDT	ICP-PCL	GNMR-ICP	NAP-ICP
Apartment	75	77	43	82	100
ETH	100	100	100	100	100
Stairs	97	97	90	100	100



(a) *Apartment sequence*



(b) *ETH sequence*



(c) *Stairs sequence*

Figure 5.7 – Cumulative probabilities of translation and rotation errors for each sequence on each evaluated algorithm. *Left:* translation error (in meters) on the horizontal axis. The vertical bar represents t_{tr}. *Right:* rotation error (in degrees) on the horizontal axis. The vertical bar represents t_{rot}.

Globally, the proposed NAP-ICP algorithm generates more successful registrations than GICP, NDT and ICP-PCL. On *Apartment*, *ETH* and *Stairs* sequences NAP-ICP achieves a 100% rate of successful registration. GICP, NDT and ICP-PCL also give a 100% success rate on *ETH* sequence. However, their results on *Stairs* are not as good, even if still satisfying. On *Apartment* NAP-ICP significantly outperforms the state-of-the-art algorithms. GICP achieves 75% of successful registrations, 77% for NDT and only 43% for ICP-PCL. This sequence includes large rotations (38% of the sequence is composed of motion with more than $\pm 35^{\circ}$ rotation on yaw axis) and GICP, NDT and ICP-PCL sometimes struggle to find the right solution when the proposed method NAP-ICP succeeds even if the assumption of small relative motion was made.

Now, considering only successful registrations, it is possible to evaluate the accuracy of the registration. Again, the Euclidean and geodesic distances from the ground truth are observed. The results are detailed in Table 5.4, giving the mean errors in translation and rotation for each tested algorithms, considering only the successful registrations. The best results are highlighted in green and the worst in red. For each sequence, as expected, ICP-PCL turns out to be the more distant to the ground truth, considering whether translation or rotation error. NAP-ICP gives the most accurate results with *Apartment* and *ETH* sequences. GICP generates the best results on *Stairs*, but the proposed algorithm NAP-ICP is not far behind.

Table 5.4 – Mean translation and rotation error for successful registration for each tested algorithm on ASL dataset.

	Translation error (m)				**Rotation error (°)**			
Method	**GICP**	**NDT**	**ICP-PCL**	**NAP-ICP**	**GICP**	**NDT**	**ICP-PCL**	**NAP-ICP**
Apartment	0.023	0.033	0.058	0.020	0.35	0.53	1.05	0.28
ETH	0.024	0.028	0.03	0.021	0.15	0.16	0.17	0.13
Stairs	0.012	0.02	0.04	0.014	0.14	0.22	0.48	0.16

The most significant feature of the proposed method in this experiment is its accuracy and robustness to large motion scenarios (especially rotations) in comparison with other algorithms.

5.4.3 Computation time comparison with state of the art algorithms

In the following experiment, NAP-ICP is compared to the three state-of-the-art registration algorithms, GICP, NDT, ICP-PCL in terms of computation time.

No speed optimization is performed in NAP-ICP, however it is important to estimate the performances of the proposed method at this point. The experiments were held on a desktop computer with an Intel Xeon W-2133, 3.6GHz CPU and 32GB RAM. Processing time for the tested algorithms on each sequence is detailed in Table 5.5. For each method, it includes all steps from point clouds preprocessing to transformation estimation.

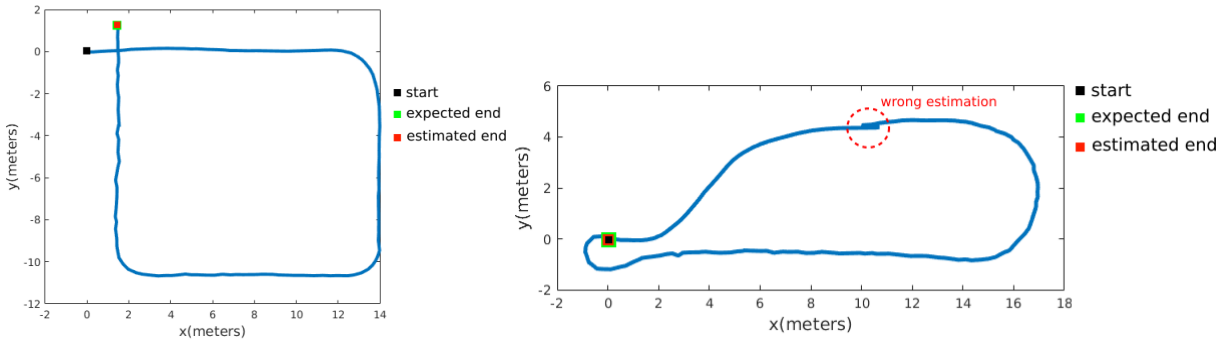
Table 5.5 – Average processing time for each sequence in milliseconds.

Sequence	GICP	NDT	ICP-PCL	NAP-ICP
Apartment	1790	233	339	500
ETH	1800	484	808	1000
Stairs	1300	211	375	360

On all sequences NDT is the fastest algorithm, followed by ICP-PCL. NAP-ICP method is slower than the aforementioned algorithms but is also more accurate. GICP is the slowest method to handle this dataset.

5.4.4 Drift sensitivity evaluation on LOOP'IN dataset

With LOOP'IN dataset, the expected goal is to close the loops of the two sequences. In Figure 5.8 we can see the trajectories estimated by NAP-ICP on both trajectories of LOOP'IN dataset. We can see that NAP-ICP estimates rather smooth trajectories for both sequences. On *Coffee Hall Loop* we can see that an error is made within the process, but NAP-ICP handles it quite well, as the rest of the trajectory is computed as expected and it succeeds in closing the loop. No particular behavior can be observed on *Balcony Loop* except from the fact that the estimated trajectory ends where it is supposed to.



(a) *Balcony Loop* (b) *Coffee Hall Loop*

Figure 5.8 – Trajectories in xy-plane computed using NAP-ICP algorithm on LOOP'IN dataset. Axes are in meters.

5.5 Conclusion

In this chapter, a method performing 3D point cloud registration based on plane-to-plane distance, NAP-ICP is presented. This method uses a score function based on plane features in order to establish the plane matches. The plane-to-plane distance minimization problem is first solved using a robust closed-form minimization approach and refined thanks to a Gauss-Newton approach. To ensure an optimal accuracy, an additional point-to-plane nonlinear optimization is performed at the end of the process.

When registering point clouds using point-based method, the number of input points can sometimes be too large. This large number has a direct impact on processing time, in particular during the minimization step. The aim of the algorithm proposed in this chapter is to perform registration based on planes, rather than points, to reduce the number of inputs.

The first experiment shows the importance of the several registration step of NAP-ICP. First, the addition of the RANSAC algorithm in the closed-form minimization step proves to generate much more accurate results. Indeed, the plane correspondences might come with a few outliers, and the RANSAC method helps removing them. Still in the first experiment, we show the importance of the addition of the point-to-plane registration at the end of the process. Planes parameters are estimated thanks to the set of points they are related to. This method is prone to approximations, which leads to a coarse registration, good enough to provide an initialization for the point-to-plane registration step that achieves a 100% success rate for all three indoor sequences of the ASL dataset.

In the second experiment, NAP-ICP is compared to state of the art algorithms GICP, NDT and ICP-PCL. On each sequence, NAP-ICP performs better than these algorithms. It

is interesting to note that NAP-ICP demonstrates to be robust to large rotation motion as it succeeds in registering the whole *Apartment* sequence (which is composed of 38% of rotation motion larger than $\pm 35°$) where GICP gave 75% success rate, NDT 77% and ICP-PCL 43%.

In the third experiment, we show that NAP-ICP algorithm is not only accurate but is a good compromise between speed and accuracy. While the tested state-of-the-art methods generate either fast or accurate results, NAP-ICP turns out to be the one giving the best trade-off.

Finally, on LOOP'IN dataset, we show that NAP-ICP is able to close loops even if unsuccessful estimations are made along the registration process.

In this chapter, we showed that using plane primitives in the estimation process allows to accurately estimate the transformation between scans, but also reduces time necessary for minimization. We also showed that NAP-ICP is the algorithm offering the best trade-off between accuracy and processing time, as can be seen on Figure 5.9 on *Apartment* sequence.

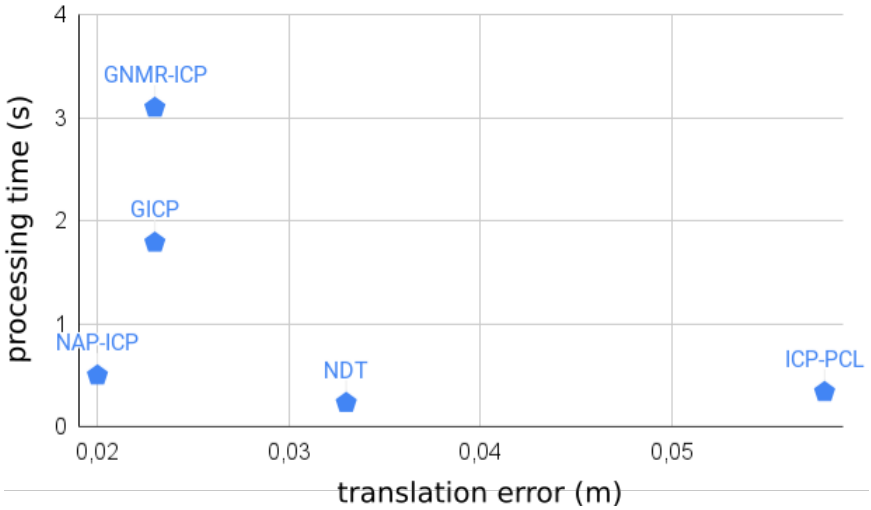


Figure 5.9 – Translation error (m) / processing time (s) for NAP-ICP and state-of-the-art algorithms on *Apartment* sequence.

We showed that finding plane matches is not a trivial problem. The use of a score metric is a good enough solution, but we argue the robustness of such a function as it is based on weights that were chosen empirically. More tests using datasets with ground truth, and more rotation motions, could give us more information about the efficiency of this method.

In the following chapter, we will show how the plane features chosen in NAP-ICP can be used within a learning process, to establish plane correspondences for a plane-based registration.

Chapter 6

Learning-based plane matching for plane-to-plane registration

6.1 Introduction

In the previous chapter, we showed that using planes instead of points was an interesting choice since it allowed to tremendously reduce the complexity of the transformation estimation problem. But, we also saw that finding plane correspondences in such a problem was not a trivial task. We established a set of plane pair features to generate a list of correspondences through a score function. This allowed us to obtain the expected behavior for NAP-ICP, highlighting the fact that those features were a good match for establishing plane correspondences.

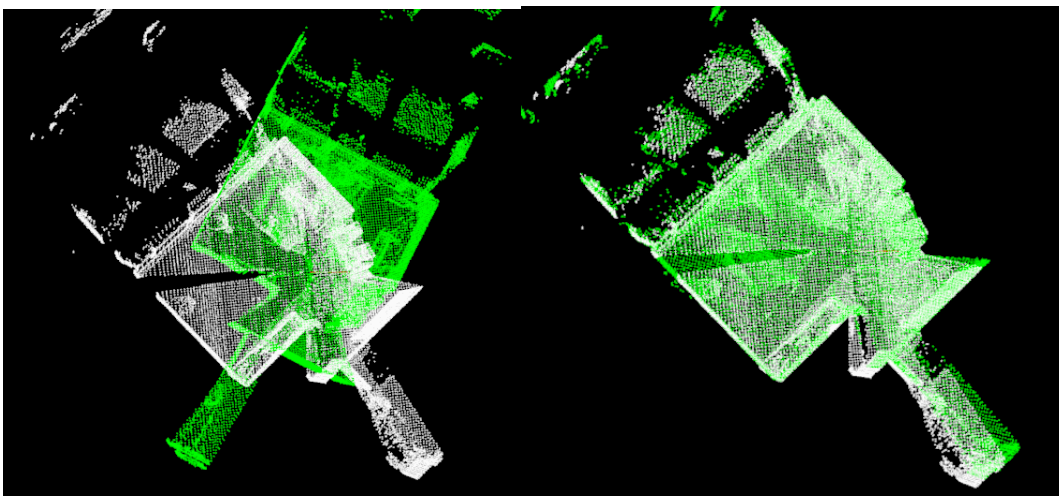
In this chapter, we present Plane-based Accurate Registration ICP (PAR-ICP), an improvement of NAP-ICP, a plane-based registration method using a learning-based method to establish plane correspondences. In the first section, a brief description of the overall function is provided. Then, the learning-based method using a Random Forest classifier is described in details. Finally, the accuracy of the registration algorithm and its ability to close loops is evaluated in a few experiments.

6.2 Overall description of PAR-ICP

As in Chapter 5, the term plane is preferred to planar patch in the following sections even if the area is used.

The method proposed in this chapter, PAR-ICP, is a variation of NAP-ICP where the plane matching is made using a learning-based method instead of a score function. An example of registration obtained with PAR-ICP is shown in Figure 6.1.

The global process of PAR-ICP is given in figure 6.2. Once more the plane-to-plane distance is minimized first, using a robust closed-form optimization and refined using a Gauss-Newton



(a) Before registration. (b) After registration.

Figure 6.1 – Example of registration between two point clouds (scans 8 and 9 from *Apartment* sequence from ASL dataset [Pomerleau et al. 2012]). The rotation between scans is large (more than 70° on yaw axis), yet the proposed method succeeds in registering the two point clouds accurately. In white the target point cloud - In green the source point cloud.

approach. An additional point-to-plane registration step is present to ensure accuracy. In section 5.2.4, we showed that finding pair correspondences is not an easy task, thus the novelty of PAR-ICP stands in its plane matching process. In NAP-ICP, plane matching was performed by extracting valid plane matches thanks to a score function built as a weighted sum of plane pairs features. The weights were chosen empirically and we argue that this solution would be robust in different scenarios. We chose to use a learning based method since the weights will not be empirically chosen but learned, and by training the method on different kind of data would be more robust. That is why in PAR-ICP, we decided to abstract from this weighting and to use a learned classification based on Random Forests.

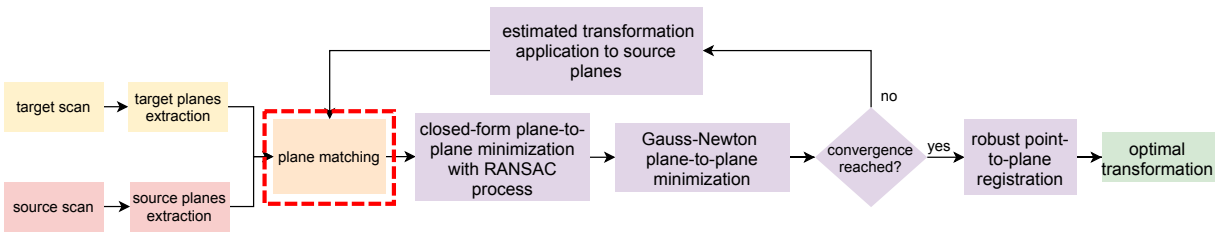


Figure 6.2 – PAR-ICP algorithm overview.

The following section will focus on how is trained the Random Forest classification to

establish plane pairs.

6.3 Learned plane matching

In section 5.3.3, we saw how we obtained a list of source planes and a list of target planes from the LiDAR data, defined by their normal, their distance from the origin, their area and their centroid. From these two lists, the aim is to establish the source/target planes correspondences. In NAP-ICP, the correspondence matching phase was performed using a score function constructed as a weighted sum of plane pairs features. The weights were chosen empirically. Moreover, the valid pairs were selected regarding a threshold also fixed empirically. Even if this function gave the expected results, we argue that this weighting would be robust to different types of data than the one we used for our experiments. In PAR-ICP, the features are not weighted, but a classifier is trained to analyze which values of the computed characteristics correspond to a valid pair of planes. This would allow to train this classifier on different types of data, coming from different sensors or types of scene to make the plane matching more robust. For each extracted plane ${}^{s}\mathbf{\Pi}_i$ in the source, a list of planes in the target, that are potential correspondences for the source plane, is built.

6.3.1 Random Forest principle

Random Forest classification [Breiman 2001] algorithm is an ensemble tree-based learning algorithm. It predicts the class of a target instance from values of several input variables, named features. A Random Forest classifier gathers the predictions of many decision trees into a single model to take a decision. An example of representation of the random forest principle is given in Fig. 6.3. The training set is composed of n instances. Each instance is represented as a vector of m features $f_1, \ldots, f_m$ and a class label. First, k samples are built from randomly chosen subsets of the original training set using the bootstrapping principle [Efron 1979]. Then, k decision trees are built based on the k bootstrap samples. Once the model is trained, it is ready to predict classes of new instances. The k trees give a prediction. The class with the highest number of votes wins. By combining several predictions of de-correlated decision trees the model is less prone to overfitting than a single decision tree. It also follows the assumption that a model made up of many mediocre trees performs better than a single good one.

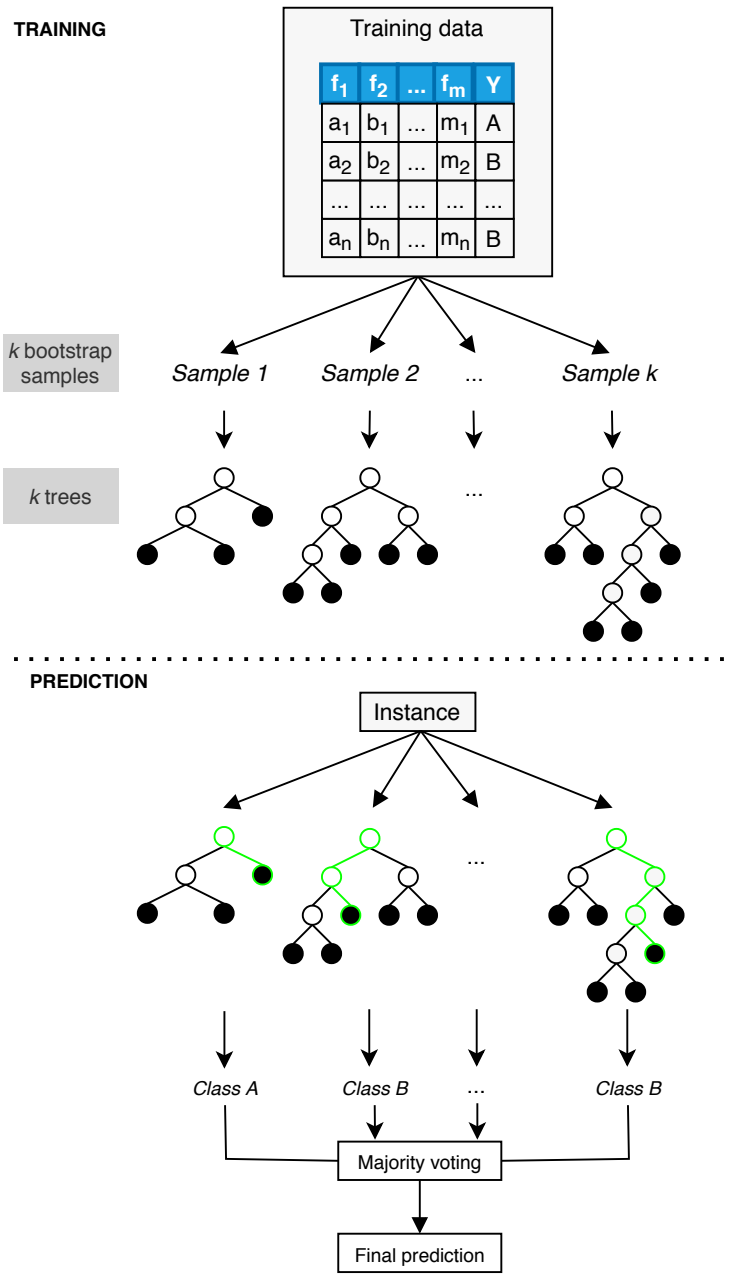


Figure 6.3 – Random Forest principle.

6.3.2 Training data and data augmentation

Labeling

To train the classifier, a set of labeled data is needed. To do so, we used the ASL dataset [Pomerleau et al. 2012]. First, planes are extracted from the point clouds as previously presented in section 5.3.3. Ground truth rotation and translation are then used to align the source and target point clouds. Using a graphic interface, for each source plane, the corresponding target plane is identified visually. The pair of identified corresponding planes is then manually labeled as a *Match* whereas all remaining source/target pairs are labeled as *Not-Match*. An illustration of plane labeling is given Figure 6.4. In Figure 6.5 an example on real data shows how the plane corresponding to the ceiling is identified in two consecutive point clouds. This operation was performed on all segmented planes.

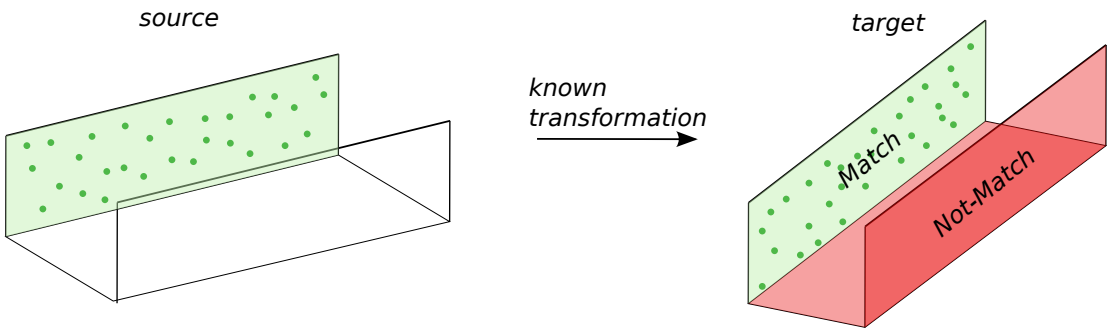


Figure 6.4 – Description of plane matching labeling. The query plane from source is identified in the target, thus the pair is labeled as *Match*. The two other planes pairs are labeled as *Not-Match*.

Moreover, we made sure that the data used for the training was balanced. As we can see in Figure 6.4, if a source plane is matched with a target plane, it means it cannot be matched with the remaining target planes. In this case we get one source/target *Match* for two *Not-Match*. Which means the dataset is composed of many more *Not-Match* labels than *Match*, if left as is. Thus, as many *Not-Match* labeled instances as *Match* were randomly picked to form the final dataset to keep the problem well-balanced.

Training

After the labeling, the source is reset to its original position, before registration and the plane pair features can be computed. As we saw in Chapter 5 that the features used for NAP-ICP allowed to obtain the expected behavior, we used them as input for the Random Forest

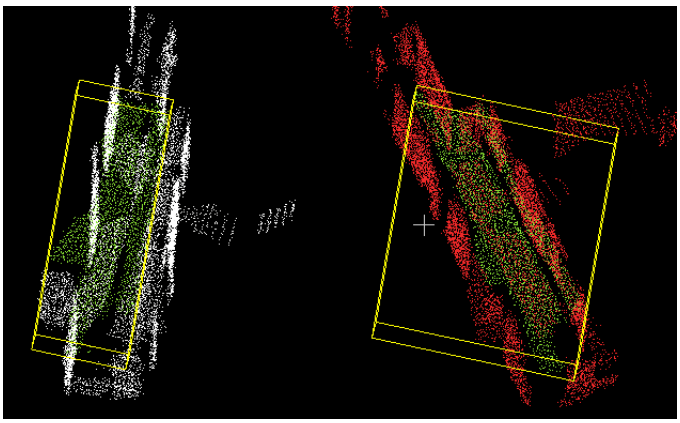


Figure 6.5 – Plane matches identification on real data using ASL dataset (*Stairs* sequence). In white the source point cloud. In red the target point cloud. Points belonging to the ceiling have been highlighted in green for both point clouds. The point clouds have been shifted for the sake of clarity.

classifier. Here is a reminder of the features for the reader:

- the distance between the projections of the origin on source planar patch and target planar patch d_o;
- the distance between the centroids of source and target planar patches d_c;
- the area ratio between the planar patches S_r;
- the dot product of the normals of the planes ϕ_n;

For more details please refer to section 5.3.4.

The input of the training step of the Random Forest is thus a set of 5×1 vectors corresponding to the features and the class label (d_o, d_c, S_r, ϕ_n, label), as we saw in section 6.3.1, with 50% of *Match* pairs and 50% of *Not-Match* pairs.

Nevertheless, the dataset is not composed of many scans. In order to train the model with a larger set, data augmentation is performed on the previously labeled data [Shorten et al. 2019]. Rotations randomly chosen in $[-70°; +70°]$ are applied on the source point clouds. The advantage is that we do not have to relabel the data, we just need to compute the features for each additional rotation.

Once trained, the Random Forest classifier is able to determine whether a pair of planes is a *Match* or a *Not-Match* given the 4×1 vectors corresponding to the features (d_o, d_c, S_r, ϕ_n) of the pair in input.

6.4 Experiments and discussions

The first experiment aims to evaluate the accuracy of the Random Forest classifier. The second experiment compares the accuracy of PAR-ICP with state of the art registration algorithms as well as NAP-ICP. The experimental conditions used in section 5.4 are used. The third experiment evaluates the ability of PAR-ICP to close loops.

6.4.1 Results on matching through classification

The classifier is evaluated through its accuracy, defined as the percentage of successful classification compared to ground truth. In order to avoid having test data in the training set, a 3-fold cross validation process was used: two sequences were used for training while the third sequence was kept for testing. The accuracy for each tested sequence is presented in Table 6.1. As can be seen, the classifier produces some false matches, but their occurrence is less than 15% on the tested data. Such a percentage of outliers is easily manageable for a robust pose estimation method, and is compatible with the RANSAC method and parametrization presented in section 5.3.5.

Table 6.1 – 3-fold cross-validation results of the pairing plane process

k	*Apartment*	*ETH*	*Stairs*	Accuracy (%)
1	*testing*	training	training	87.9
2	training	*testing*	training	89.3
3	training	training	*testing*	91.6

6.4.2 Accuracy comparison with state of the art algorithms

In this experiment, PAR-ICP is compared to three state-of-the-art registration algorithms as well as NAP-ICP in terms of accuracy on the three indoor sequences of the ASL dataset. PAR-ICP is compared to the same algorithms as in Chapter **??**: GICP ([Segal et al. 2009]), NDT ([Magnusson et al. 2015], and the point-to-plane ICP. The parametrization used in section 5.4.2 is used. *Note that in this context, for PAR-ICP, the 3-cross fold validation is used for the Random Forest classifier establishing correspondences, in order to avoid using data that were already seen in the validation. When the algorithm is tested on Apartment it*

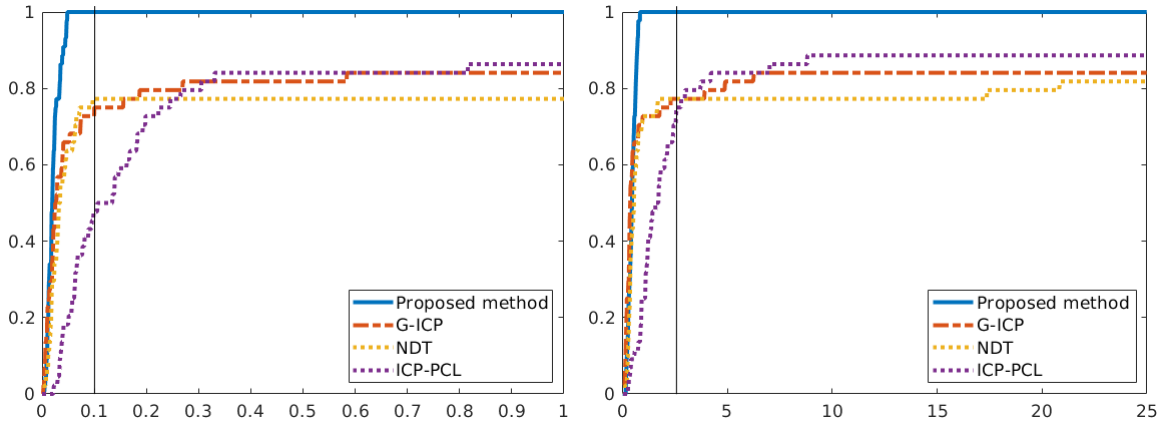
has been trained only on ETH and Stairs. When the algorithm is tested on ETH it has been trained only on Apartment and Stairs. When the algorithm is tested on Stairs it has been trained only on ETH and Apartment

The cumulative probabilities for translation and rotation errors are given in Figure 6.6 and the success rates are summarized in Table 6.2. In [Zong et al. 2019], several algorithms are evaluated following the same procedure: Fast Global Registration (FGR) [Zhou et al. 2016], GH-ICP [Yue et al. 2018] and the algorithm proposed in the article further called Planar Global Registration (PGR). Their success rate were added in the Table 6.2. For more details about the algorithms parameters, please refer to [Zong et al. 2019].

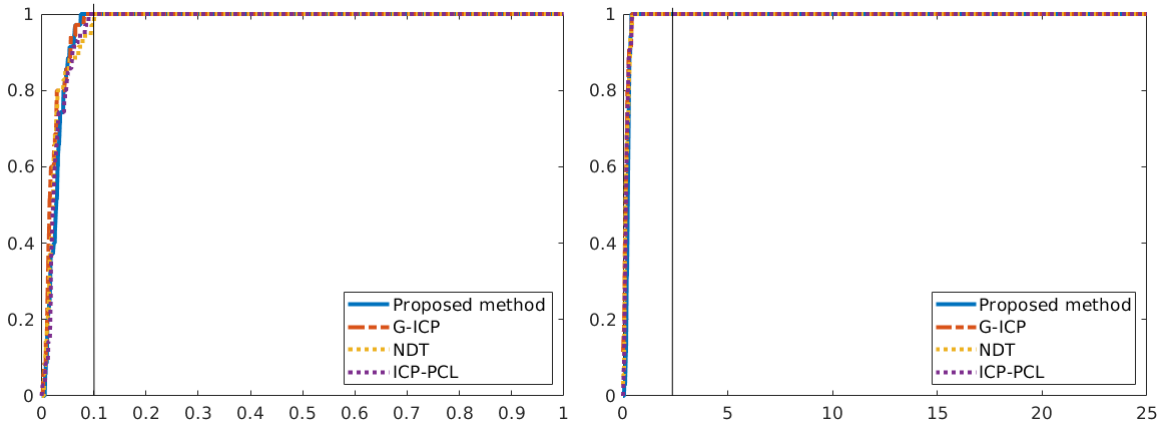
Globally, PAR-ICP and PGR perform very well in indoor environments with strong planar features. On each sequence, PAR-ICP succeeds in registering 100% of the scans according to the chosen thresholds, whereas PGR while registering 100% of successful registration on *Apartment* reaches 90% on *ETH* and 97% on *Stairs*. As expected PAR-ICP generates as much successful registration as NAP-ICP. Considering the other algorithms, the major difference is visible on *Apartment* sequence. While PAR-ICP succeeds in registering all the scans, GICP only achieves 75% of successful registrations, 77% for NDT, 43% for ICP-PCL and 52% for GH-ICP. It is probably due to the presence of large rotations in the sequence, proving PAR-ICP robustness to this type of motion. The exception if for FGR that succeeds in registering a 100% of *Apartment* but performs poorly on *ETH* and *Stairs*.

Table 6.2 – Percentage of successful registration (translation and rotation combined) for the evaluated algorithms on each sequence

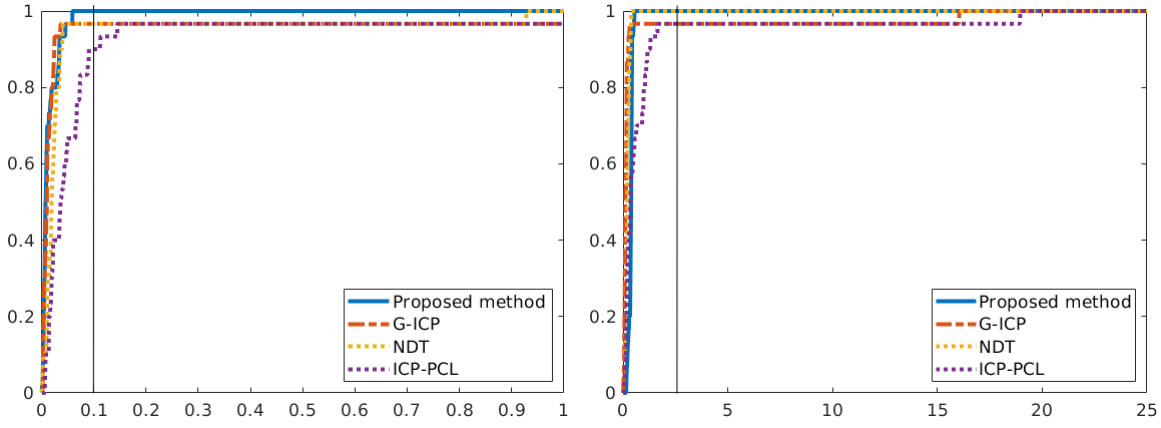
Sequence	GICP	NDT	ICP-PCL	PGR	FGR	GH-ICP	NAP-ICP	PAR-ICP
Apartment	75	77	43	100	100	52	100	100
ETH	100	100	100	90	63	93	100	100
Stairs	97	97	90	97	3	94	100	100



(a) *Apartment sequence*



(b) *ETH sequence*



(c) *Stairs sequence*

Figure 6.6 – Cumulative probabilities of translation and rotation errors for each sequence on each evaluated algorithm. *Left:* translation error (in meters) on the horizontal axis. The vertical bar represents the threshold t_{tr}. *Right:* rotation error (in degrees) on the horizontal axis. The vertical bar represents the threshold t_{rot}.

Now considering only successful registrations, it is possible to evaluate the accuracy of the registration. The Euclidean and geodesic distance from the ground truth are observed. The results are detailed in Table 6.3, giving the mean errors in translation and rotation for each tested algorithm, considering only the successful registrations. The best results are highlighted in green and the worst in red. As we saw previously, ICP-PCL turns out to be the more distant to the ground truth, considering whether translation or rotation error. GICP gives the most accurate results on all sequences, except for the translation in *Apartment*. Even if GICP issues more accurate results on the successful registrations, let us remind that PAR-ICP provided a larger success rate of estimations.

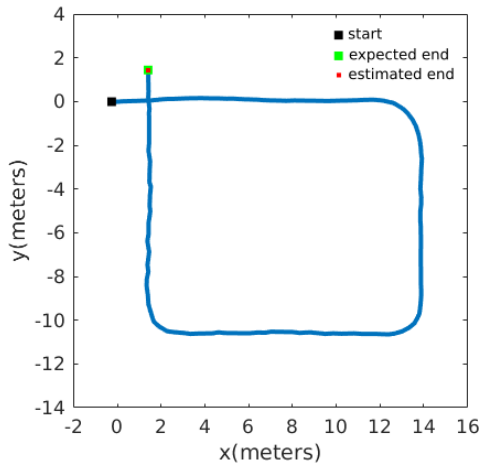
Table 6.3 – Mean translation and rotation error for successful registration for each tested algorithm on ASL dataset.

	Translation error (m)				Rotation error (°)			
Method	**GICP**	**NDT**	**ICP-PCL**	**PAR-ICP**	**GICP**	**NDT**	**ICP-PCL**	**PAR-ICP**
Apartment	0.023	0.033	0.058	0.022	0.35	0.53	1.05	0.45
ETH	0.024	0.028	0.03	0.029	0.15	0.16	0.17	0.23
Stairs	0.012	0.02	0.04	0.013	0.14	0.22	0.48	0.36

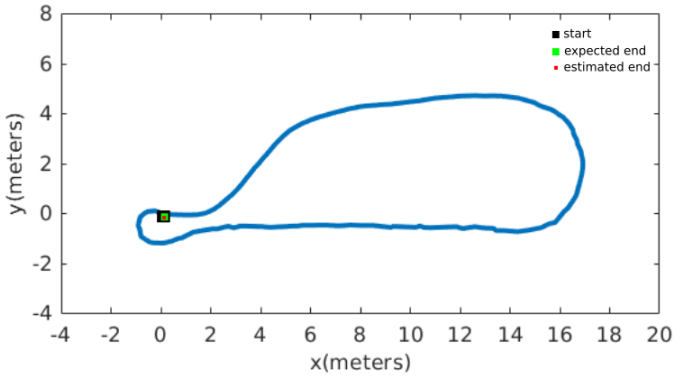
6.4.3 Registration in LOOP'IN dataset and incremental map creation using PAR-ICP

In the following experiment, the sensor trajectories of the LOOP'IN sequences are computed using PAR-ICP. In this context, all ASL sequences were used to train the Random Forest classifier establishing correspondences. While each point cloud is being registered, a so-called incremental map is being built and updated. The registration of new input source scans is not done frame-by-frame but by registering the scan to the incremental map. This allows to match source planes with planes that were met earlier in the sequence, which gives important information in a long sequence, even more if a loop is performed. Planes in the source, that are already known and identified as matches in early steps, are updated in the map, and new unknown planes are added, expecting to be matched further in the reconstruction process. The trajectory on x and y axes for each sequence is presented in Fig. 6.7, with the respective incremental map. In both sequences, PAR-ICP is able to provide a satisfying path, and to close the loop between the starting and the ending poses, when using the incremental map for

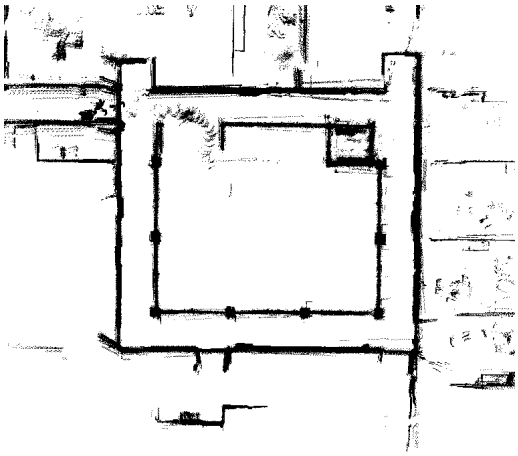
registration. Some inaccuracies in the trajectory can be observed from time to time (visible on the incremental maps), but they do not prevent the algorithm to close the loop.



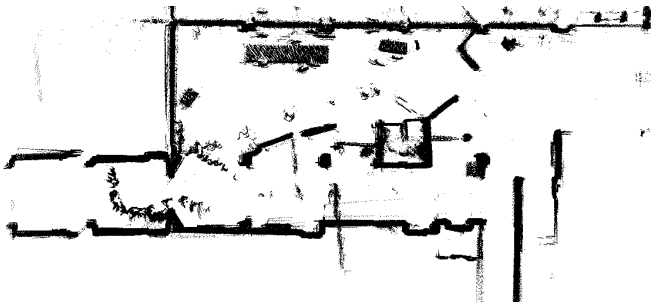
(a) *Balcony loop* sequence



(b) *Coffee Hall* sequence



(c) *Balcony loop* incremental map



(d) *Coffee Hall loop* incremental map

Figure 6.7 – Trajectories in xy-plane computed using PAR-ICP algorithm and incremental maps for each sequence (floor and ceiling were removed to ease reading). Axes are in meters.

The consistency of the incremental maps can be seen on *Balcony loop* with the segmented floor and ceiling. The sequence is acquired on a balcony surrounding stairs, thus a few points from the stairs were captured, and they are clearly visible on the segmented floor in figure 6.8.

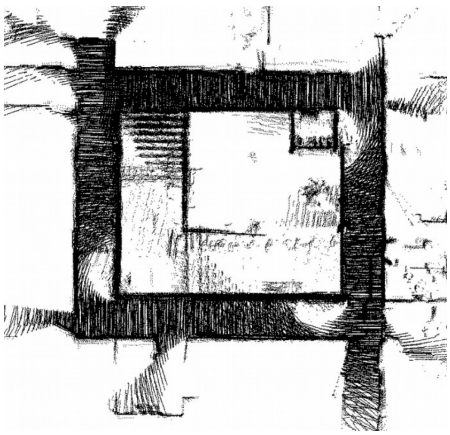


Figure 6.8 – Segmented floor of *Balcony loop* incremental map generated by PAR-ICP.

6.5 Conclusion

In this chapter, a method performing 3D point clouds registration based on plane-to-plane distance, PAR-ICP is presented. This method uses a classifier based on a Random Forest, trained with plane features in order to establish the plane matches. PAR-ICP combines robust plane-to-plane and point-to-plane registrations to ensure an accurate registration.

Registering point clouds using planes in man-made environments allows to reduce the dimensionality of the classic point-to-point problem. However, establishing plane correspondences is challenging, more specifically in case of large motion scenarios. Thus, we trained a Random Forest classifier with plane pairs features in order to perform the classification of plane matches. To train the Random Forest classifier, pairs of matched and unmatched planes were established thanks to the ASL dataset and its ground truth.

The first experiment evaluated the accuracy of the Random Forest classifier. We saw that it was not perfect (87.9% for the worst case), but is easily manageable by the RANSAC method used in PAR-ICP.

In the second experiment, PAR-ICP is compared to state of the art algorithms GICP, NDT and ICP-PCL. This experiment showed that PAR-ICP gives better results in terms of percentage of successful registration than other state of the art algorithms as it succeeds in registering all three indoor sequences of ASL dataset. It also showed to give a better trade-off between processing time and accuracy than the other state-of-the-art algorithms as can be seen in Figure 6.9 with an example on *Apartment* sequence. It also demonstrated the robustness of PAR-ICP to large rotation motion as PAR-ICP succeeds in registering the whole *Apartment* sequence (which is composed of 38% of rotation motion larger than $\pm 35°$).

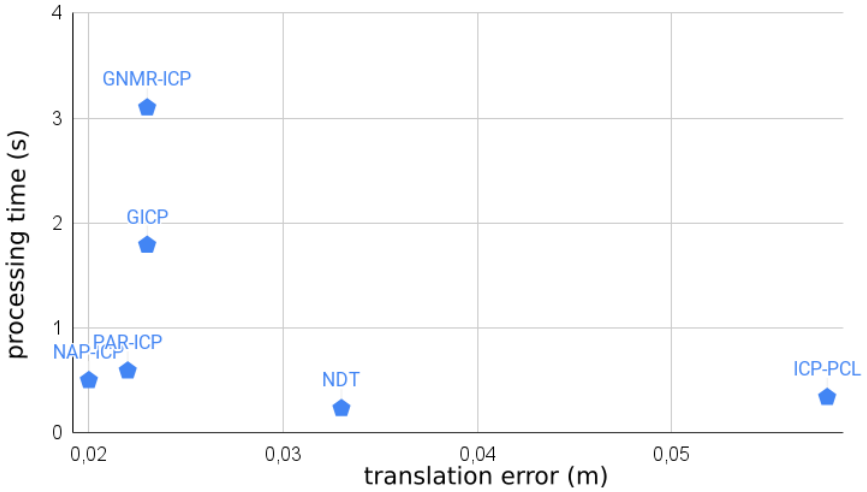


Figure 6.9 – Translation error (m) / processing time (s) for PAR-ICP and state-of-the-art algorithms on *Apartment* sequence.

In the third experiment we showed that PAR-ICP was able to close loops in LOOP'IN dataset. The estimated trajectories are rather smooth, allowing to generate a consistent incremental map.

In this chapter, we outlined that using plane features in a Random Forest based classification, to establish plane correspondences, was an interesting choice. It allows to remove the uncertainties caused by the weighting parameters of the score function presented in Chapter 5. PAR-ICP, by providing consistent map with the LOOP'IN scenarios, proved its ability to be used in mapping scenarios and would be a nice fit for SLAM applications to make the maps more precise.

Conclusion

Contributions

In this thesis we addressed the problem of data registration. Our goal was to improve registration methods based only on LiDAR data, in other words, 3D point clouds. We focused on how to take advantage of structured environments to provide algorithms that allow a trade off between accuracy, robustness and computation time by reducing the dimensionality of the registration problem.

In the first contribution of this thesis we presented LOOP'IN, a new challenging dataset, designed to evaluate the ability of registration algorithms to close loops. This dataset is made of two long sequences. They were made long in the purpose to identify if the evaluated algorithm was subject to drift or not. In order to cope with the fact that we were not able to measure precisely the ground truth of the sensor pose, we added loop to the trajectories, so that we would be able to identify if the algorithms behaved appropriately by closing those loops.

In the second contribution of this thesis we proposed a registration algorithm based on a multi-resolution scheme called GNMR-ICP. To ensure a fast and robust minimization to estimate the transformation parameters, we chose to minimize the point-to-plane distance using a Gauss-Newton method. We showed the importance of using robust functions in the minimization process to reduce the influence of outliers, by adding M-estimators in this key step.

The experiments on ASL dataset showed that GNMR-ICP is more accurate and more robust than its closed-form equivalent using the small angle approximation, regardless the chosen M-estimators. In the second experiment, we saw that using more levels of registration in the multi-resolution process generated more accurate results as well as reducing computation time. We noted that the time reduction notion applied more obviously on very structured data, such as man-made environments. The experiment on LOOP'IN dataset showed that GNMR-ICP is not robust enough to recover from an error in the estimation and thus, to close loops in the provided scenarios.

The second contribution has highlighted the idea of taking advantage of the structured data, thus the third contribution focused on plane-to-plane registration in man-made environments. An algorithm called NAP-ICP designed to register planes was proposed. In order to ensure a robust minimization, this process is performed in two steps: first, a closed-form minimization is made using a RANSAC process to initialize the estimation and to identify the inliers. Then an iterative Gauss-Newton minimization on the plane-to-plane distance is performed to refine the estimation. As using planes tends to smooth the original data and in the sake of accuracy, an additional point-to-plane registration is performed at the end of the process to make the estimation as precise as possible. The plane-to-plane distance minimization was only possible thanks to the score function we designed to build plane correspondences. Plane matching turns out to be a challenging problem, so we decided to create a function based on weighted plane features in order to estimate which planes are correspondents. The function highlighted that the plane features we chose were a good match since the behavior of the algorithm was the one we expected. However, we argue the robustness of such a function as it requires weights settings that were made empirically.

The first experiment proved the importance of the RANSAC step and the point-to-plane additional registration step on NAP-ICP process accuracy. Indeed, NAP-ICP was able to successfully register 100% of the indoor sequences of ASL dataset by combining these steps. The second experiment showed that NAP-ICP gave more successful results and was more accurate than state of the art algorithms, once more, on the ASL dataset. Finally, thanks to the experiment on LOOP'IN dataset, we saw that NAP-ICP was able to close loops and also could recover from an error made in the estimation during the registration of a long sequence.

The method presented in the third contribution was efficient, but making the score function more robust by not applying empirically chosen weights was an interesting perspective. Hence, in the fourth contribution we chose to present a plane-to-plane registration method, PAR-ICP, where plane correspondences were built thanks to a learning method. A Random Forest was trained in order to classify whether two planes were to a valid match or not. To train the Random Forest we kept the features chosen in the previous contribution. The data used for training were the point clouds from ASL dataset. Since we had the ground truth of the sensor pose, it was possible to easily identify the plane matches, and to label the data. Classification on the test dataset showed that the accuracy of the Random Forest classifier was not perfect (87.9% on the *Apartment* test set) but this was manageable by the RANSAC algorithm used

in the minimization step, so that we obtained 100% successful registrations.

As expected, the experiment showed that PAR-ICP was able to successfully register the indoor sequences of the ASL dataset. The algorithm also provided good results on the LOOP'IN dataset as it was able to close the loops and to provide rather consistent incremental maps for both scenarios.

Perspectives

Making the Random Forest classifier more robust

On PAR-ICP, even if we performed data augmentation to generate more data for training, we think that the classifier would be more robust if it were trained on data coming from different types of sensors, and with "real" transformations between successive scans. The difficulty lies on the labeling of the pair correspondences of the data.

Porting on embedded systems

The global context of this thesis was to solve the point clouds registration problem in a navigation intention. This implies using the designed algorithms on mobile platforms, thus embedded systems. Porting GNMR-ICP or PAR-ICP on embedded systems would be very interesting but a significant work on architecture and optimization would have to be made to give the possibility to use the algorithms online in real-time scenarios. More over, the methods designed here are good candidates for GPU acceleration, they would surely give interesting results on a system such as a Jetson AGX Xavier, a GPU for embedded systems designed by NVIDIA. However, due to embedded constraints, the consumption must also be taken into account and might not be always compatible with hardware acceleration.

Large scale datasets

An interesting task that is missing in this work is testing the several proposed algorithms on datasets with larger point clouds, such as cities. The estimation problem and constraints are similar as the ones of the tested datasets yet different, for instance, we can imagine that in city reconstruction the initial solution is further from the optimal solution than in indoor scenarios. Moreover, it would allow to confront the methods robustness to different kinds of sensors.

SLAM applications

Also, an improvement that naturally comes to mind in this context is to adapt PAR-ICP to SLAM applications. We saw that it gave very good results in registration and localization. Its accuracy allowed us to create an incremental map, but the proposed method lacks a step where PAR-ICP questions the previously estimated transformation thanks to loop closure in order to correct this map. Furthermore, a nice application would be to directly feed the algorithm with a map (such as a 3D blueprint of a building, implying precision inaccuracies between real life and the map) and thanks to it, the map could be updated and corrected by what PAR-ICP sees in real-time. A measure of confidence in the data could be given to emphasize whether the map and the estimations are good or if inaccuracies are detected (because of the sensor precision or the provided map).

Appendix A

Notations

General rules

a lower case	:	scalar
$\mathbf{a}$ lower case and bold	:	vector
$\mathbf{A}$ upper case and bold	:	matrix

Transformations

$\mathcal{F}_a$	:	frame a
${}^a\mathbf{t}_b$	:	translation vector linking $\mathcal{F}_b$ to $\mathcal{F}_a$
${}^a\mathbf{R}_b$	:	rotation matrix linking $\mathcal{F}_b$ to $\mathcal{F}_a$
${}^a\mathbf{T}_b$	:	rigid transformation matrix linking $\mathcal{F}_b$ to $\mathcal{F}_a$

Points representation

${}^s\mathbf{P}$, ${}^t\mathbf{P}$	:	respectively, the source and target point clouds
${}^s\mathbf{p}_i$, ${}^t\mathbf{p}_i$	:	the i^{th} point in, respectively, the source and target point cloud
${}^t\mathbf{n}_i$	:	the normal vector of the i^{th} point in the target point cloud
$\overline{\mathbf{p}} = (X, Y, Z)^\top$	:	Cartesian coordinates of a 3D point
$\mathbf{p} = (WX, WY, WZ, W)^\top$	:	homogeneous coordinates of a 3D point

Distances

d_i	:	the Euclidean point to point distance
$d_i^\perp$	:	the point to plane distance
$\mathbf{d}_i^\Pi$	:	the plane to plane distance

BIBLIOGRAPHY

Arun, K. S., T. S. Huang, and S. D. Blostein (May 1987), « Least-squares fitting of two 3-d point sets », eng, *in*: *IEEE transactions on pattern analysis and machine intelligence* 9.*5*, pp. 698–700 (cit. on pp. 54, 95).

Arya, Sunil and David M Mount (1993), « Approximate Nearest Neighbor Queries in Fixed Dimensions. », *in*: *SODA*, vol. 93, Citeseer, pp. 271–280 (cit. on p. 35).

Babin, P., P. Giguère, and F. Pomerleau (May 2019), « Analysis of Robust Functions for Registration Algorithms », *in*: *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1451–1457 (cit. on p. 74).

Beaton, Albert E. and John W. Tukey (1974), « The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data », *in*: *Technometrics* 16.*2*, pp. 147–185 (cit. on p. 46).

Behley, Jens and Cyrill Stachniss (2018), « Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. », *in*: *Robotics: Science and Systems*, vol. 2018 (cit. on p. 59).

Ben-Shabat, Yizhak, Michael Lindenbaum, and Anath Fischer (2019), « Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds Using Convolutional Neural Networks », *in*: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 38).

Besl, P. J. and N. D. McKay (Feb. 1992), « A method for registration of 3-D shapes », *in*: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.*2*, pp. 239–256 (cit. on pp. 15, 49, 50, 53, 71).

Biber, Peter and Wolfgang Straßer (2003), « The normal distributions transform: A new approach to laser scan matching », *in*: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3, IEEE, pp. 2743–2748 (cit. on p. 55).

Blanco, Jose-Luis, Francisco-Angel Moreno, and Javier Gonzalez (2009), « A collection of outdoor robotic datasets with centimeter-accuracy ground truth », *in*: *Autonomous Robots* 27.*4*, pp. 327–351 (cit. on pp. 61, 63).

Breiman, Leo (Oct. 2001), « Random Forests », en, *in*: *Machine Learning* 45.*1*, pp. 5–32 (cit. on p. 117).

Brou, Philippe (1984), « Using the Gaussian image to find the orientation of objects », *in*: *The International Journal of Robotics Research* 3.*4*, pp. 89–125 (cit. on p. 54).

Burel, Gilles and Hugues Henocq (1995), « Three-dimensional invariants and their application to object recognition », *in*: *Signal processing* 45.*1*, pp. 1–22 (cit. on p. 53).

Cazals, F. and M. Pouget (2005), « Estimating differential quantities using polynomial fitting of osculating jets », *in*: *Computer Aided Geometric Design* 22.*2*, pp. 121–146 (cit. on p. 38).

Chatila, Raja and Jean-Paul Laumond (1985), « Position referencing and consistent world modeling for mobile robots », *in*: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, IEEE, pp. 138–145 (cit. on p. 57).

Chen, Songlin et al. (2019a), « PLADE: A Plane-Based Descriptor for Point Cloud Registration With Small Overlap », *in*: *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–11 (cit. on p. 53).

— (Apr. 2020), « PLADE: A Plane-Based Descriptor for Point Cloud Registration With Small Overlap », en, *in*: *IEEE Transactions on Geoscience and Remote Sensing* 58.*4*, pp. 2530–2540 (cit. on pp. 16, 96).

Chen, Xieyuanli et al. (2019b), « Suma++: Efficient lidar-based semantic slam », *in*: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 4530–4537 (cit. on p. 59).

Chen, Xieyuanli et al. (2021), « OverlapNet: Loop closing for LiDAR-based SLAM », *in*: *arXiv preprint arXiv:2105.11344* (cit. on p. 59).

Chen, Yang and Gérard Medioni (Apr. 1992), « Object modelling by registration of multiple range images », *in*: *Image and Vision Computing*, Range Image Understanding 10.*3*, pp. 145–155 (cit. on pp. 15, 51).

Chum, O. and J. Matas (2005), « Matching with PROSAC - progressive sample consensus », *in*: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 220–226 vol. 1 (cit. on p. 45).

Davenport, P. B. (1968), « A vector approach to the algebra of rotations with applications », *in*: (cit. on p. 94).

Debeunne, César and Damien Vivet (2020), « A review of visual-LiDAR fusion based simultaneous localization and mapping », *in*: *Sensors* 20.*7*, p. 2068 (cit. on p. 57).

Dosovitskiy, Alexey et al. (2017), *CARLA: An Open Urban Driving Simulator*, arXiv: 1711.03938 [cs.LG] (cit. on p. 62).

Dubé, Renaud et al. (2017), « Segmatch: Segment based place recognition in 3d point clouds », *in*: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 5266–5272 (cit. on p. 59).

Efron, B. (1979), « Bootstrap Methods: Another Look at the Jackknife », *in*: *The Annals of Statistics* 7.*1*, pp. 1–26 (cit. on p. 117).

Fischler, Martin A. and Robert C. Bolles (June 1981), « Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography », *in*: *Communications of the ACM* 24.*6*, pp. 381–395 (cit. on pp. 44, 45, 96).

Fitzgibbon, Andrew (Jan. 2003), « Robust Registration of 2D and 3D Point Sets », en-US, *in*: *Image and Vision Computing* 21 (cit. on pp. 15, 54, 74).

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012), « Are we ready for autonomous driving? the kitti vision benchmark suite », *in*: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, pp. 3354–3361 (cit. on pp. 62, 63).

Grant, W. Shane, Randolph C. Voorhies, and Laurent Itti (Nov. 2013), « Finding planes in LiDAR point clouds for real-time registration », *in*: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4347–4354 (cit. on pp. 96, 100).

— (June 2019), « Efficient Velodyne SLAM with point and plane features », en, *in*: *Autonomous Robots* 43.*5*, pp. 1207–1224 (cit. on pp. 16, 52, 59, 94, 96).

Griffiths, David and Jan Boehm (2019), « SynthCity: A large-scale synthetic point cloud », *in*: *ArXiv preprint* (cit. on pp. 62, 63).

Grisetti, Giorgio, Cyrill Stachniss, and Wolfram Burgard (2007), « Improved techniques for grid mapping with rao-blackwellized particle filters », *in*: *IEEE transactions on Robotics* 23.*1*, pp. 34–46 (cit. on p. 58).

Grisetti, Giorgio et al. (2010), « A tutorial on graph-based SLAM », *in*: *IEEE Intelligent Transportation Systems Magazine* 2.*4*, pp. 31–43 (cit. on p. 58).

Grisettiyz, Giorgio, Cyrill Stachniss, and Wolfram Burgard (2005), « Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling », *in*: *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, pp. 2432–2437 (cit. on p. 58).

Gschwandtner, Michael et al. (2011), « BlenSor: Blender sensor simulation toolbox », *in*: *International Symposium on Visual Computing*, Springer, pp. 199–208 (cit. on p. 62).

Guo, Yulan et al. (2016), « A comprehensive performance evaluation of 3D local feature descriptors », *in*: *International Journal of Computer Vision* 116.*1*, pp. 66–89 (cit. on p. 53).

Han, Jen-Yu, Nei-Hao Perng, and Yan-Ting Lin (2013), « Feature conjugation for intensity-coded LIDAR point clouds », *in*: *Journal of Surveying Engineering* 139.*3*, pp. 135–142 (cit. on p. 53).

Hartley, R. I. and A. Zisserman (2004), *Multiple View Geometry in Computer Vision*, Second, Cambridge University Press, ISBN: 0521540518 (cit. on p. 40).

Hoppe, Hugues et al. (1992), « Surface reconstruction from unorganized points », *in*: *Proceedings of the 19th annual conference on computer graphics and interactive techniques*, pp. 71–78 (cit. on p. 37).

Horn, Berthold K. P. (Apr. 1987), « Closed-form solution of absolute orientation using unit quaternions », EN, *in*: *JOSA A* 4.*4*, pp. 629–642 (cit. on p. 53).

Horn, Berthold Klaus Paul (1984), « Extended gaussian images », *in*: *Proceedings of the IEEE* 72.*12*, pp. 1671–1686 (cit. on p. 54).

Hornung, Armin et al. (2013), « OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees », *in*: *Autonomous Robots*, Software available at `http://octomap.github.com` (cit. on p. 36).

Huber, P.J., J. Wiley, and W. InterScience (1981), *Robust statistics*, Wiley New York (cit. on p. 46).

Huber, Peter J (1992), « Robust estimation of a location parameter », *in*: *Breakthroughs in statistics*, Springer, pp. 492–518 (cit. on p. 55).

Johnson, Andrew E. and Martial Hebert (1999), « Using spin images for efficient object recognition in cluttered 3D scenes », *in*: *IEEE Transactions on pattern analysis and machine intelligence* 21.*5*, pp. 433–449 (cit. on p. 53).

Jost, T. and H. Hugli (Oct. 2003), « A multi-resolution ICP with heuristic closest point search for fast and robust 3D registration of range images », *in*: *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.* Pp. 427–433 (cit. on p. 74).

Kang, Sing Bing and Katsushi Ikeuchi (1991), « Determining 3-D object pose using the complex extended Gaussian image. », *in*: *CVPR*, vol. 91, pp. 580–585 (cit. on p. 54).

Li, Kailai, Meng Li, and Uwe D. Hanebeck (2021), « Towards High-Performance Solid-State-LiDAR-Inertial Odometry and Mapping », *in*: *IEEE Robotics and Automation Letters* 6.*3*, pp. 5167–5174 (cit. on pp. 62, 63).

Lorusso, A, D W Eggert, and R B Fisher (1995), « A Comparison of Four Algorithms forEstimating 3-D Rigid Transformations », en, *in*: *1995 BMVC* 1, pp. 237–246 (cit. on p. 95).

Low, Kok-lim (2004), *Linear least-squares optimization for point-toplane ICP surface registration*, tech. rep., Department of Computer Science, University of North Carolina at Chapel Hill (cit. on pp. 15, 52, 54, 73, 90).

Lu, Feng and Evangelos Milios (1997), « Globally consistent range scan alignment for environment mapping », *in*: *Autonomous robots* 4.*4*, pp. 333–349 (cit. on p. 58).

Lu, Weixin et al. (2019), « DeepICP: An end-to-end deep neural network for 3D point cloud registration », *in*: *arXiv preprint arXiv:1905.04153* (cit. on p. 56).

Magnusson, Martin, Achim Lilienthal, and Tom Duckett (2007), « Scan registration for autonomous mining vehicles using 3D-NDT », en, *in*: *Journal of Field Robotics* 24.*10*, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20204, pp. 803–827 (cit. on p. 55).

Magnusson, Martin et al. (2009), « Appearance-based loop detection from 3D laser data using the normal distributions transform », *in*: *2009 IEEE International Conference on Robotics and Automation*, IEEE, pp. 23–28 (cit. on p. 60).

Magnusson, Martin et al. (May 2015), « Beyond points: Evaluating recent 3D scan-matching algorithms », *in*: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3631–3637 (cit. on pp. 15, 18, 65, 82, 89, 90, 104, 108, 121).

Malis, Ezio and Eric Marchand (2006), « Experiments with robust estimation techniques in real-time robot vision », *in*: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 223–228 (cit. on p. 80).

Mérigot, Quentin, Maks Ovsjanikov, and Leonidas J. Guibas (2011), « Voronoi-Based Curvature and Feature Estimation from Point Clouds », *in*: *IEEE Transactions on Visualization and Computer Graphics* 17.*6*, pp. 743–756 (cit. on p. 38).

Nüchter, A. et al. (2005), « 6D SLAM with approximate data association », *in*: *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* Pp. 242–249 (cit. on p. 35).

Nuchter, Andreas, Kai Lingemann, and Joachim Hertzberg (Aug. 2007), « Cached k-d tree search for ICP algorithms », *in*: pp. 419–426 (cit. on p. 35).

Pais, G Dias et al. (2020), « 3dregnet: A deep neural network for 3d point registration », *in*: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7193–7203 (cit. on p. 56).

Pandey, Gaurav, James R McBride, and Ryan M Eustice (2011), « Ford campus vision and lidar data set », *in*: *The International Journal of Robotics Research* 30.*13*, pp. 1543–1552 (cit. on pp. 61, 63).

Pathak, K. et al. (Oct. 2009), « Fast 3D mapping by matching planes extracted from range sensor point-clouds », *in*: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1150–1155 (cit. on pp. 16, 96).

Pathak, Kaustubh et al. (June 2010), « Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping », en, *in*: *IEEE Transactions on Robotics* 26.*3*, pp. 424–441 (cit. on pp. 55, 96).

Phillips, Jeff M., Ran Liu, and Carlo Tomasi (2007), « Outlier Robust ICP for Minimizing Fractional RMSD », *in*: *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*, pp. 427–434 (cit. on p. 74).

Pomerleau, F., F. Colas, and R. Siegwart (May 2015), « A Review of Point Cloud Registration Algorithms for Mobile Robotics », English, *in*: *Foundations and Trends® in Robotics* 4.*1*, pp. 1–104 (cit. on p. 51).

Pomerleau, François et al. (Dec. 2012), « Challenging data sets for point cloud registration algorithms », en, *in*: *The International Journal of Robotics Research* 31.*14*, pp. 1705–1711 (cit. on pp. 18, 61–63, 91, 98, 116, 119).

Pomerleau, François et al. (2013), « Comparing ICP variants on real-world data sets Open-source library and experimental protocol », en, *in*: *Autonomous Robots* 34.*3*, pp. 133–148 (cit. on p. 65).

Poppinga, J. et al. (Sept. 2008), « Fast plane detection and polygonalization in noisy 3D range images », *in*: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3378–3383 (cit. on p. 96).

Pottmann, Helmut, Stefan Leopoldseder, and Michael Hofer (2004), « Registration without ICP », *in*: *Computer Vision and Image Understanding* 95.*1*, pp. 54–71 (cit. on p. 51).

Qi, Charles R et al. (2017), « Pointnet++: Deep hierarchical feature learning on point sets in a metric space », *in*: *arXiv preprint arXiv:1706.02413* (cit. on p. 56).

Rabbani, T., F. A. van den Heuvel, and G. Vosselman (2006), « Segmentation of point clouds using smoothness constraints », English, *in*: *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, pp. 248–253 (cit. on pp. 96, 98).

Rusinkiewicz, S. and M. Levoy (May 2001), « Efficient variants of the ICP algorithm », *in*: *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pp. 145–152 (cit. on pp. 51, 54, 72).

Rusu, Radu Bogdan (Nov. 2010), « Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments », en, *in*: *KI - Künstliche Intelligenz* 24.*4*, pp. 345–348 (cit. on pp. 37, 38).

Rusu, Radu Bogdan, Nico Blodow, and Michael Beetz (2009), « Fast point feature histograms (FPFH) for 3D registration », *in*: *2009 IEEE international conference on robotics and automation*, IEEE, pp. 3212–3217 (cit. on pp. 53, 55).

Rusu, Radu Bogdan and Steve Cousins (May 2011), « 3D is here: Point Cloud Library (PCL) », en, *in*: *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 1–4 (cit. on p. 37).

Sanchez, Julia et al. (2017), « Global registration of 3D LiDAR point clouds based on scene features: Application to structured environments », *in*: *Remote Sensing* 9.*10*, p. 1014 (cit. on pp. 54, 62, 63).

Sanchez, Julia et al. (2020), « Robust normal vector estimation in 3D point clouds through iterative principal component analysis », *in*: *ISPRS Journal of Photogrammetry and Remote Sensing* 163, pp. 18–35 (cit. on p. 38).

Segal, A., D. Haehnel, and S. Thrun (2009), « Generalized-ICP », *in*: *Proc. of Robotics : Science and Systems* 2, p. 4 (cit. on pp. 15, 52, 74, 89, 108, 121).

Shorten, Connor and Taghi M Khoshgoftaar (2019), « A survey on image data augmentation for deep learning », *in*: *Journal of Big Data* 6.*1*, pp. 1–48 (cit. on p. 120).

Steder, Bastian, Giorgio Grisetti, and Wolfram Burgard (2010), « Robust place recognition for 3D range data based on point features », *in*: *2010 IEEE International Conference on Robotics and Automation*, IEEE, pp. 1400–1405 (cit. on p. 59).

Steder, Bastian et al. (2011), « Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation », *in*: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 1249–1255 (cit. on p. 59).

Taguchi, Y. et al. (May 2013), « Point-plane SLAM for hand-held 3D sensors », *in*: *2013 IEEE International Conference on Robotics and Automation*, pp. 5182–5189 (cit. on pp. 54, 94–96, 102).

Tombari, Federico, Samuele Salti, and Luigi Di Stefano (2010), « Unique signatures of histograms for local surface description », *in*: *European conference on computer vision*, Springer, pp. 356–369 (cit. on p. 53).

Torr, P.H.S. and A. Zisserman (2000), « MLESAC: A New Robust Estimator with Application to Estimating Image Geometry », *in*: *Computer Vision and Image Understanding* 78.*1*, pp. 138–156 (cit. on p. 45).

Turk, Greg and Marc Levoy (1994), « Zippered polygon meshes from range images », *in*: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 311–318 (cit. on p. 61).

Vlaminck, M., H. Luong, and W. Philips (May 2017), « Multi-resolution ICP for the efficient registration of point clouds based on octrees », *in*: *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pp. 334–337 (cit. on pp. 54, 74, 75, 90, 91).

Wang, Fang et al. (2016), « Point cloud registration by combining shape and intensity contexts », *in*: *2016 9th IAPR Workshop on Pattern Recogniton in Remote Sensing (PRRS)*, IEEE, pp. 1–6 (cit. on p. 53).

Wright, Stephen and Jorge Nocedal (1999), « Numerical optimization », *in*: *Springer Science* 35.*67-68*, p. 7 (cit. on p. 95).

Yue, Pan et al. (2018), « Iterative Global Similarity Points: A robust coarse-to-fine integration solution for pairwise 3D point cloud registration », *in*: *2018 International Conference on 3D Vision (3DV)* (cit. on p. 122).

Zhang, Ji and Sanjiv Singh (2014), « LOAM: Lidar Odometry and Mapping in Real-time. », *in*: *Robotics: Science and Systems*, vol. 2, 9 (cit. on p. 53).

Zheng, Yinglong et al. (2018), « Rolling normal filtering for point clouds », *in*: *Computer Aided Geometric Design* 62, pp. 16–28 (cit. on p. 38).

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun (Oct. 2016), « Fast Global Registration », *in*: *Computer Vision – ECCV 2016*, vol. 9906, pp. 766–782 (cit. on pp. 16, 122).

Zong, Wenpeng et al. (Dec. 2019), « A Fast and Accurate Planar-Feature-Based Global Scan Registration Method », *in*: *IEEE Sensors Journal* 19.*24*, pp. 12333–12345 (cit. on pp. 16, 89, 94, 96, 100, 122).

Acronyms

ASL Autonomous Systems Lab. 18, 20–23, 61, 62, 65, 67, 70, 76, 80, 82, 89, 91, 98, 103, 108, 112, 119, 121, 124, 126, 129–131

BFGS Broyden–Fletcher–Goldfarb–Shanno. 95

BSP Binary Space Partitionning. 34

CNN Convolutional Neural Network. 56

DNN Deep Neural Network. 56

FGR Fast Global Registration. 122

FPFH Fast Point Feature Histogram. 53, 55

GICP Generalized ICP. 52, 89, 90, 108, 110–113, 121, 122, 124, 126

GNMR-ICP Gauss-Newton based Multi-Resolution ICP. 17, 18, 22, 26, 28, 71, 75–78, 80, 82–85, 88–91, 129, 131

GPS Global Positionning System. 30, 31, 57

IC3PO Iterative Closest Point Plus Plane Optimization. 52, 53, 94

ICP Iterative Closest Point. 15, 28, 49–51, 53, 55, 56, 60, 71–75, 82–84, 88–91, 108, 110–113, 121, 122, 124, 126

ILS Iterative Least Squares. 43

IMU Inertial Measurement Unit. 30, 62

IRLS Iteratively Re-weighted Least Squares. 45

LiDAR Light Detection And Ranging. 14–16, 22, 28–34, 36, 51, 56–60, 62, 64, 67, 69, 70, 91, 117

MLESAC Maximum Likelihood Estimation SAmple and Consensus. 45

MSAC M-estimator SAmple and Consensus. 45

MUMC Minimally Uncertain Maximum Consensus. 55, 96

NAP-ICP New Accurate Plane-based ICP. 18–21, 23, 27, 28, 93, 97, 103, 104, 108, 110–113, 115–117, 119, 121, 122, 130

NDT Normal Distribution Transform. 15, 55, 60, 89, 90, 108, 110–113, 121, 122, 126

PAR-ICP Plane-based Accurate Registration ICP. 21, 23, 27, 28, 115–117, 121, 122, 124, 126, 127, 130–132

PCA Principal Component Analysis. 37, 38, 53, 77, 98

PCL Point Cloud Library. 37, 89, 99

PCV Pair Consistency Voting. 38

PGR Planar Global Registration. 122

PROSAC PROgressive SAmple Consensus. 45

RANSAC RANdom SAmple Consensus. 19, 21, 27, 44–46, 56, 59, 96, 97, 99, 102, 104, 112, 121, 126, 130

SAC-IA SAmple Consensus Initial Alignment. 55

SHOT Signature of Histograms of OrienTations. 53

SLAM Simultaneous Localization And Mapping. 24, 28, 49, 56–60, 62, 127, 132

SSFR Structured Scene Features-based Registration. 54

SVD Singular Value Decomposition. 42, 54, 94, 95

VCM Voronoï Covariance Measure. 38

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Recalage de nuages de points issus de LiDAR pour la localisation dans des environnements intérieurs.

Mot clés : nuage de point, recalage, scènes intérieures, localisation

Résumé : Cette thèse traite du problème du recalage de nuages de points 3D dans des environnements intérieurs. Tout d'abord nous présentons l'algorithme multi-résolution GNMR-ICP, minimisant de manière robuste la distance point-à-plan entre deux nuages de points à l'aide d'une méthode de Gauss-Newton. La multi-résolution est faite grâce à un octree. Sur le jeu de données de référence ASL, GNMR-ICP donne des résultats plus précis que son équivalent utilisant l'approximation des petits angles (81% de succès contre 43%). Les temps de calculs dans les environnements structurés sont réduis (jusqu'à un facteur 2). Ensuite nous présentons NAP-ICP, un algorithme basé sur le recalage de plans. La mise en correspondance des plans est effectuée à l'aide d'une fonction de score basée sur les caractéristiques de paires de plans. Un recalage point-à-plan supplémentaire est effectué pour assurer un maximum de précision. NAP-ICP recale 100% des scènes intérieures du jeu de données ASL, est plus précis que les fonctions de l'état de l'art évaluées et est capable de fermer les boucles du jeu de données LOOP'IN. Enfin, PAR-ICP, une méthode plan-à-plan où la mise en correspondance est faite à l'aide d'un Random Forest est présentée. PAR-ICP recale 100% des scènes intérieures du jeu de données ASL et est capable de fermer les boucles de LOOP'IN, permettant de générer des cartes incrémentales.

Title: LiDAR-based point clouds registration for localization in indoor environments.

Keywords: point cloud, registration, indoor scenes, localization

Abstract: This thesis deals with the problem of registration of 3D point clouds in indoor environments. Registration methods are proposed to obtain a compromise between time and accuracy. First, GNMR-ICP, a multi-resolution algorithm which robustly minimizes the point-to-plane distance between two point clouds using a Gauss-Newton method. The multi-resolution is done using an octree. On the ASL benchmark dataset, GNMR-ICP gives more accurate results than its equivalent using the small angle approximation (81% success rate against 43%). Computation times in structured environments are reduced (up to a factor of 2). Next we present NAP-ICP, an algorithm based on plane matching. Planes are matched using a score function based on the characteristics of pairs of planes. An additional point-to-plane registration is performed to ensure maximum accuracy. NAP-ICP registers 100% of the interior scenes of the ASL dataset and is more accurate than the evaluated state-of-the-art functions and is able to close the loops of the LOOP'IN dataset. Finally, PAR-ICP, a plane-based method where the matching is performed using a Random Forest is presented. PAR-ICP registers 100% of the interior scenes of the ASL dataset and is able to close the loops of LOOP'IN, allowing to generate incremental maps.