Affinement de nuages de points par optimisation des paramètres extrinsèques

Sommaire

Introduction

L'amélioration de la qualité des données en post-traitement, ou **affinement de données**, est un domaine de recherche vaste et très suivi par la communauté : le but est le plus souvent de compenser certaines faiblesses des systèmes d'acquisitions en supprimant des erreurs introduites pendant le

processus d'obtention des données, ou tout du moins en les réduisant. Le but de ces méthodes est généralement de lisser les données, ou de réduire les erreurs introduites lors de l'obtention de ces données : il s'agit de méthodes utilisées en pré-traitement d'autres applications où une bonne précision des données en entrée est requise.
Dans ce chapitre, nous présentons une méthode qui permet d'améliorer la qualité de jeux de données nuages de points : il s'agit de l'affinement de données de type nuages de points par optimisation des paramètres de calibrage extrinsèque. Ce chapitre est organisé de la façon suivante : dans un premier temps, nous montrons les effets d'un mauvais calibrage extrinsèque sur le rendu et l'exploitation des données issues d'une cartographie mobile ; ensuite, nous présentons une synthèse des travaux existants dans le domaine de l'optimisation du calibrage extrinsèque ; la section suivante traitera de la méthode d'optimisation que l'on a mise au point et de son principe, suivi d'une section présentant différents résultats d'optimisation ; enfin, nous évoquerons quelques facteurs limitants de notre algorithme. Aussi, il convient de noter que la méthode d'optimisation que l'on va présenter est applicable à des systèmes équipé de plusieurs LIDARs, ou d'un capteur LIDAR multi-fibres car nous utilisons la redondance de données entre plusieurs sources pour optimiser les paramètres de calibrage extrinsèque.

3.1 Importance des paramètres de calibrage extrinsèque

Pour avoir une cartographie correcte à l'issue d'une acquisition mobile, le calibrage extrinsèque occupe une place centrale : en effet, sur un véhicule mobile de cartographie équipé d'un capteur LIDAR, le capteur laser permet d'acquérir les données, qui sont référencées en coordonnées sphériques le plus souvent ; une fois que les données sont référencées dans le repère cartésien du capteur, il y a une transformation **extrinsèque** pour représenter les données dans le repère du véhicule, le repère « body ». La figure 2.6 illustre cette transformation, avec la rotation R et la translation T entre les deux repères. Avec un mauvais calibrage extrinsèque, après projection dans les repères body puis monde, les données ne seront pas correctement géoréférencées, faussant la cartographie et l'ensemble des traitements prévus sur les données.

3.1.1 Définition du calibrage extrinsèque pour un système mobile LIDAR

Le calibrage extrinsèque décrit un changement de repère, généralement entre le repère « capteur » - qui est lié au capteur LIDAR - et le repère « body » - qui lui est lié au véhicule mobile -. La transformation qui projette un point $p'(t)$ du repère capteur en un point $p(t)$ du repère body est la suivante :

$$p(t) = R_{ext} * p'_t + T_{ext} \tag{3.1}$$

$$\text{avec :} \begin{cases} T_{ext} = T(t_x, t_y, t_z) = [t_x \quad t_y \quad t_z]^T \\ R_{ext} = R(\alpha, \beta, \gamma) = R_{\mathbf{z}}(\gamma) * R_{\mathbf{y}}(\beta) * R_{\mathbf{x}}(\alpha) \\ \\ R_{\mathbf{x}}(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \\ R_{\mathbf{y}}(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \\ R_{\mathbf{z}}(\gamma) = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{cases}$$

$R_{\mathbf{z}}(\gamma)$, $R_{\mathbf{y}}(\beta)$, et $R_{\mathbf{x}}(\alpha)$ représentent respectivement les matrices de rotations autour des trois axes d'un repère cartésien orthonormé : dans notre cas, il s'agit des rotations autour des trois axes

du repère body permettant de projeter les données du repère capteur au repère body. Les rotations sont représentées par les 3 angles α, β et γ, que l'on appelle aussi « roulis, tangage et lacet ». Pour les translations selon chacun des trois axes, elles sont représentées par le vecteur T_{ext}. On appelle paramètres extrinsèques les 6 paramètres dont les valeurs sont obtenues par étalonnage extrinsèque du système d'acquisition, et qui caractérisent la transformation que l'on vient de décrire.

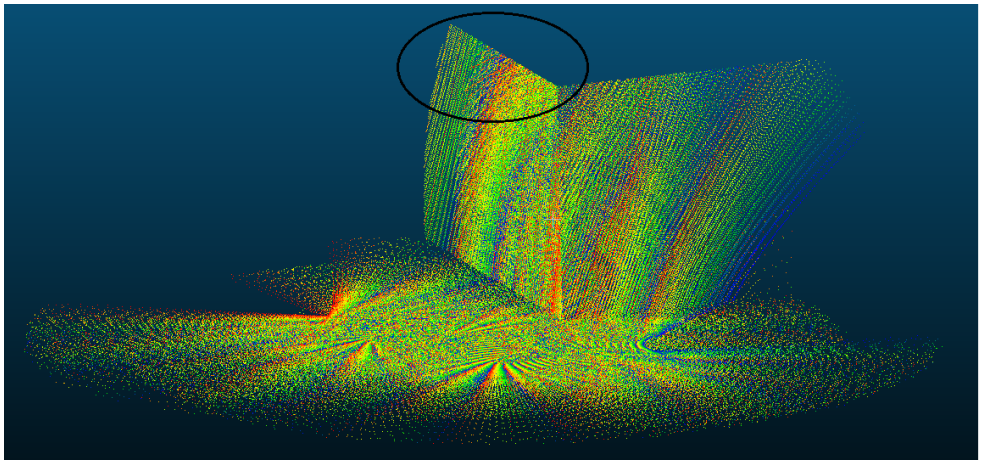
3.1.2 Effets d'un mauvais étalonnage des paramètres extrinsèques sur les données

Un mauvais étalonnage des paramètres extrinsèques implique une projection des données dans le repère body erronée. Dans un premier temps, si l'on a une erreur au niveau des paramètres de translation, le problème qui apparaît est une translation de l'ensemble des données avec un offset constant, qui est égal à la valeur de l'erreur dans le cas d'un capteur mono-fibre ; dans le cas d'un capteur multi-fibres, comme présenté en figure 3.1, on a le même résultat pour chaque fibre du capteur. La sous-figure a) présente le nuage avec un étalonnage correctement effectué, et la sous-figure b) un zoom sur un plan vertical du nuage. On voit avec la sous-figure c) les effets d'une erreur sur les paramètres de translations : le plan est déformé, mais les données de chaque fibre restent consistantes entre elles. L'erreur appliquée est de quelques dizaines de centimètres, entre 20 et 30 centimètres selon l'axe. Avec la sous-figure d), on montre les déformations introduites avec une erreur sur les paramètres de rotations extrinsèque. Chaque point du nuage est acquis dans une certaine direction du capteur LIDAR, et une erreur sur la rotation extrinsèque déforme les données de manière non consistante, au contraire d'une erreur sur la translation extrinsèque. La sous-figure d) illustre ce type d'erreur : cette fois-ci, on remarque très nettement la déformation par rapport à la sous-figure b). L'erreur que l'on illustre sur cette figure est de quelques degrés, entre 2 et 3 degrés.

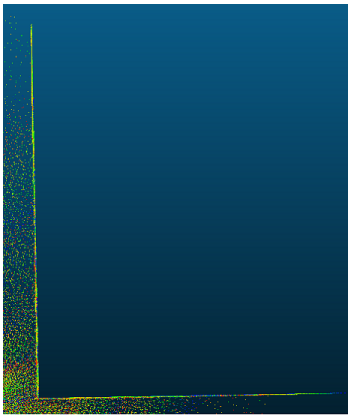
3.2 Etat de l'art

3.2.1 Calibrage automatique de différents systèmes d'acquisitions

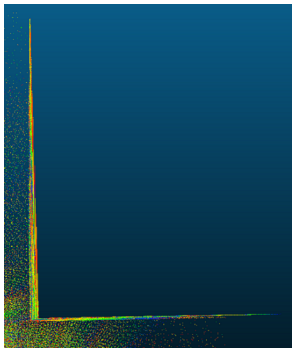
Il existe plusieurs types de capteurs LIDAR, que l'on peut séparer en deux catégories principales comme présenté en section 2.2.3. Par conséquent, on trouve aussi de nombreuses méthodes de calibrage des systèmes d'acquisitions équipés de capteur LIDARs, car la plupart des systèmes embarquent un ou plusieurs capteurs, et nécessitent un calibrage spécifique. En effet, si on s'intéresse aux systèmes comportant au moins un capteur laser, on trouve quelques configurations particulières : par exemple, Huang [Huang et Barth, 2009] présente un système utilisé pour effectuer des acquisitions, qui est composé d'une caméra et de deux capteurs lasers 3D. Pour l'étalonnage, une mire est utilisée, et les paramètres des deux types de capteurs sont estimés en même temps. Dans le même style, Mazzei [Mazzei *et al.*, 2012] propose une méthode d'étalonnage des paramètres de calibrage pour un système hybride caméras/LIDAR, composé de sept caméras et quatre capteurs lasers 2D et 3D. Leur approche utilise des marqueurs au sol dont la position est connue précisément, pour effectuer l'étalonnage des paramètres de calibrage extrinsèque du système complet. L'idée particulière est que pour les lasers, il y en a un « principal », qui est positionné à l'avant du véhicule, et il y a les autres lasers : l'étalonnage des lasers restants se fait en s'aidant des données issues du capteur laser principal, qui est calibré à l'aide d'une vérité terrain. Enfin, on peut aussi noter l'article de Maddern [Maddern *et al.*, 2012], qui présente un système composé d'un capteur laser 3D et de deux capteurs laser 2D. Selon les auteurs, leur méthode fonctionne pour des configurations composées d'un capteur laser 3D et de n capteurs laser 2D. L'étalonnage du système se fait en deux étapes : d'abord, celle du capteur 3D, qui est automatique, et qui ne nécessite une intervention humaine que pour l'initialisation des paramètres ; puis celle du/des capteur(s) 2D, qui se base sur la



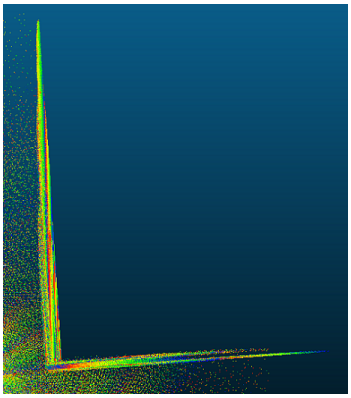
(a)



(b)



(c)



(d)

FIGURE 3.1 – Nuages de points avec différents étalonnages extrinsèques : a) Paramètres corrects ; b) Paramètres corrects, avec zoom sur la zone d'intérêt ; c) Erreurs sur les paramètres de translations ; d) Erreurs sur les paramètres de rotations

structure du nuage créé par le capteur laser 3D. Le nuage produit par les capteurs 2D est comparé avec celui produit par le capteur 3D déjà calibré.

3.2.2 L'optimisation des paramètres de calibrage extrinsèque d'un capteur LIDAR

Dans la section 3.2.1, nous avons présenté quelques méthodes d'étalonnage automatique des paramètres de calibrage de systèmes de cartographie équipés de capteurs LIDAR mono-fibres principalement : ce type de capteur est très utilisé pour plusieurs raisons, comme par exemple leur précision correcte et leur coût faible pour une grande partie des modèles. De nombreuses méthodes d'optimisation des paramètres de calibrage existent, mais ces capteurs bas coûts ne sont pas très bien adaptés à l'acquisition d'informations 3D, sauf pour les modèles RIEGL, mais dont la gamme de prix se rapproche de celle des capteurs multi-fibres. Par exemple, dans [Gao et Spletzer, 2010], les auteurs présentent un système composé de plusieurs LIDAR 2D, et proposent une optimisation des paramètres extrinsèques pour chaque capteur en se servant de la réflexion des impacts lasers sur des cibles réfléchissantes posées sur des poteaux verticaux. Dans [Lin *et al.*, 2013], les auteurs

présentent un système composé d'un capteur LIDAR mono-fibre monté sur une plateforme pivotante, ce qui permet d'avoir des données 3D : avec la rotation induite par la plateforme, le système nécessite un calibrage spécifique, qui est présenté dans l'article. Dans [Maddern *et al.*, 2012], les auteurs présentent un système composé d'un capteur LIDAR 3D et de 2 capteurs mono-fibres : la particularité est que le capteur 3D est composé de 3 capteurs mono-fibres placés sur une plateforme tournante.

D'un autre côté, on trouve aussi des capteurs permettant d'acquérir des données 3D, mais multi-fibres, c'est-à-dire qu'ils englobent plusieurs « lasers », comme par exemple les capteurs Velodyne [Velodyne LIDAR sensors, 2017] ou Quanergy [Quanergy product page, 2017], sorti plus récemment. Peu de méthodes d'optimisation des paramètres extrinsèques pour des systèmes équipés de ce genre de capteur existent dans la littérature. Dans [Zhu et Liu, 2013], les auteurs proposent une méthode d'optimisation des 3 paramètres de rotation en 2 étapes : tout d'abord, l'optimisation des angles de roulis et de tangage en estimant des paramètres planaires pour les sols, puis l'angle de lacet en mettant en correspondance des objets de forme cylindrique. La méthode est non supervisée et ne nécessite pas de mire de calibrage comme la plupart des routines de calibrage de systèmes d'acquisitions, mais il y a une limitation du fait que seul les paramètres de rotation sont estimés. Dans [Huang *et al.*, 2013], les auteurs proposent une optimisation de l'ensemble des paramètres extrinsèques d'un capteur LIDAR multi-fibres, mais pour cela, ils utilisent une mire de calibrage et une caméra infrarouge qui leur permet de récupérer une image des impacts lasers sur la mire, et l'optimisation vise à avoir une correspondance correcte entre les nuages de points et les images infrarouges. Enfin, dans [Elseberg *et al.*, 2013], les auteurs cherchent à optimiser les paramètres de calibrage induits par leur système composé de plusieurs capteurs LIDARs. Ils mesurent la compacité de leurs nuages de points avec une fonction qui est une somme de fonctions de densités de points qu'ils cherchent à minimiser. La méthode est non supervisée, n'utilise pas de mire de calibrage et est appliquée en post-traitement.

Dans la littérature, deux articles ont retenus notre attention : [Levinson et Thrun, 2010], dans lequel les auteurs présentent une méthode non supervisée d'optimisation des paramètres extrinsèques d'un système équipé d'un capteur multi-fibres, sans utilisation de mire de calibrage et en post-traitement, ainsi que [Mengwen *et al.*, 2014], où une optimisation des paramètres extrinsèques entre un capteur LIDAR et le repère body est présenté, aussi en post-traitement. Nous expliquerons plus en détails en section 3.3 pourquoi ces deux articles nous intéressent.

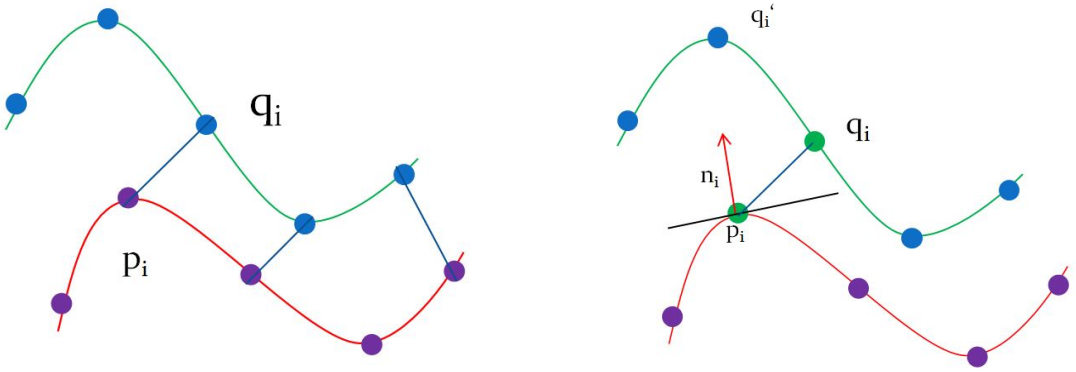
3.2.3 Le recalage de données

Le recalage de données occupe une place importante dans de nombreux domaines : la reconstruction 3D ; la cartographie - mobile ou statique - ; les méthodes de type SLAM orienté trajectoire. Dans le cas de la cartographie (et aussi dans celui du SLAM), le scan matching est une étape importante, puisqu'elle permet d'affiner le géo-référencement des scans, et par conséquent la localisation du véhicule lors de l'acquisition. Selon la fréquence des points de contrôle – c'est-à-dire des instants où l'on cherche à déterminer le vecteur d'état du véhicule -, la localisation sera plus ou moins précise, au détriment de l'application temps réel. Dans notre cas, le recalage de données est important car comme nous allons le présenter en section 3.3, l'optimisation que nous avons choisi d'appliquer s'appuie sur du recalage de données.

L'algorithme de scan matching le plus connu et le plus utilisé est l'Iterative Closest Point (ICP). Cet algorithme est apparu conjointement en 1991 et 1992, respectivement présenté par Chen et Medioni [Chen et Medioni, 1991], et par Besl et Mckay [Besl et McKay, 1992]. En effet, des algorithmes de mise en correspondance de scans (scan registration) existaient déjà, mais Chen et Medioni sont les premiers à utiliser une distance point à un plan pour recaler les deux scans, et Besl et Mckay ont démontré la convergence de l'algorithme lorsqu'une configuration initiale proche de la solution était connue.

L'algorithme d'ICP permet de mettre en correspondance des scans pris à des instants consécutifs différents, ou acquis à des positions différentes. Pour cela, les scans doivent être référencés par rapport à une origine commune et dans un repère commun : en effet, pour effectuer le recalage entre les deux scans, l'algorithme cherche à mettre en correspondance des points appariés issus de chacun des deux scans, d'où la nécessité d'avoir une référence commune, et un des deux scans est pris comme référence, le deuxième scan étant recalé par rapport à cette référence. Une fonctionnelle avec laquelle la distance moyenne entre les deux scans est minimisée itérativement ; la fonctionnelle a en général cette forme :

$$J(R,T) = \frac{1}{N} * \sum_{i=1}^{N} [d_{R,T}(p_i, q_i)]^2 \tag{3.2}$$



(a) Illustration de la distance point à point

(b) Illustration de la distance point à plan

FIGURE 3.2 – Schéma des deux types de distances principalement utilisées pour l'ICP

Dans l'équation (3.2), N représente le nombre de points appariés entre les deux scans et i itère sur l'ensemble des points p_i que l'on a sélectionné du scan de référence ; les points q_i sont les points mis en correspondances avec les points p_i. Enfin, la distance qui est utilisée suit principalement deux métriques différentes : une distance point à point (3.3), introduite par Besl et Mckay [Besl et McKay, 1992], et une distance point à plan (3.4), introduite par Chen et Medioni [Chen et Medioni, 1991].

$$d_{R,T}(p_i, q_i) = \|(R * p_i + T) - q_i\| \tag{3.3}$$

avec R la rotation et T la translation estimées entre les deux scans à mettre en correspondance

$$d_{R,T}(p_i, q_i) = \|n_i \cdot [(R * p_i + T) - q_i]\| \tag{3.4}$$

avec R la rotation et T la translation estimées entre les deux scans à mettre en correspondance ; n_i représente la normale au plan local auquel appartient le point p_i.

Sur la figure 3.3, on peut voir un exemple de recalage entre deux scans du lapin de Stanford. Dans ce cas là, on a un recouvrement total des deux scans, ce qui rend la mise en correspondance plus simple, puisque chaque point d'un scan est apparié à un point de l'autre scan. On peut tout de même noter que, bien que l'ICP soit très utilisé pour aligner des scans entre eux, d'autres approches existent. On trouve par exemple l'algorithme de **Normal Distribution Transform** (NDT), introduit par Peter Biber [Biber et Straber, 2003] : l'idée de base ressemble à celle de la carte d'occupation, puisque l'on a une carte 2D divisée en plusieurs cellules de taille réglable. Cependant, la probabilité de mesurer un échantillon dans une cellule est modélisée par une distribution normale. Le scan matching est effectué plus ou moins de la même manière que l'ICP : les deux scans sont

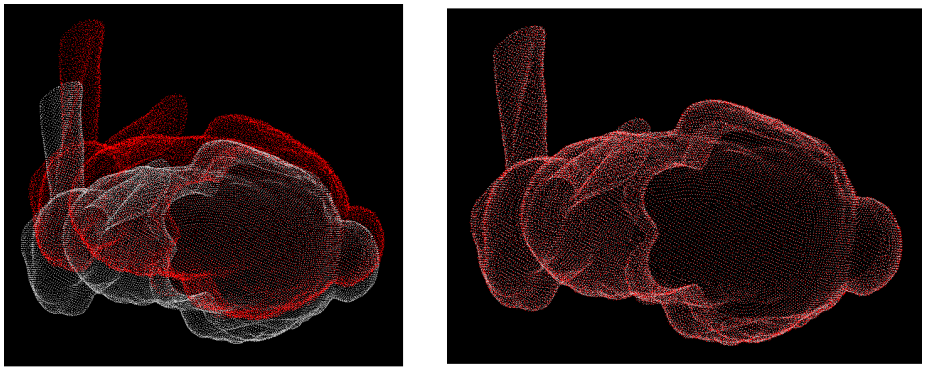


FIGURE 3.3 – Exemple de recalage entre deux scans

référencés dans un même référentiel, les NDT sont calculées pour les deux scans, puis comparées entre elles, et le processus est itéré jusqu'à convergence. Toutefois, au niveau de la comparaison, la fonctionnelle à minimiser n'est pas la même : là où pour l'ICP, la fonctionnelle à minimiser est une distance point à point ou point à plan (équations (3.3) et (3.4)), pour le NDT, on cherche à minimiser une fonctionnelle f(X), qui vaut :

$$f(R,T) = -\sum_{i=1}^{n} h(T[(R,T), x_i]) \tag{3.5}$$

où R et T sont les paramètres de calibrage à optimiser, x_i les points à recaler et $T[(R,T)x_i]$ les points « corrigés », après application de la transformation (rotation et translation) trouvée. Cette fonctionnelle est basée sur des densités de probabilités de mesures d'échantillons, comme indiqué ci-dessus. Cette approche est au départ introduite pour effectuer du 2D-SLAM, mais Magnusson [Magnusson *et al.*, 2007] proposent une extension en 3 dimensions. Magnusson propose ensuite deux articles qui traitent tous les deux de l'algorithme de NDT [Magnusson *et al.*, 2009b], [Magnusson *et al.*, 2009a]. Dans le premier, l'efficacité de l'ICP et la NDT sont comparés : il se trouve que la NDT permet de converger plus vite et a un meilleur taux de réussite que l'ICP, mais n'est pas assez précis. En effet, les problèmes de convergence ne peuvent pas réellement être prédit (il y a convergence malgré un grand écart d'alignement, mais l'algorithme échoue pour un plus petit écart) et lorsque la méthode NDT échoue à converger, l'erreur résiduelle est plus élevée qu'avec l'ICP. Dans le deuxième article, une fermeture de boucle effectuée avec un algorithme de NDT est présentée, afin de voir si les résultats sont satisfaisants, comme avec des systèmes visuels par exemple. Pour les auteurs, la réponse est positive, ce qui est assez encourageant pour de futures améliorations, comme celles qu'a eu l'algorithme d'ICP.

3.2.3.1 L'algorithme d'ICP

Comme le résument Rusinkiewicz et Levoy [Rusinkiewicz et Levoy, 2001], l'algorithme d'ICP est composé de quatre étapes (il y en a six dans leur article, mais quatre sont plus importantes que les autres, les deux autres sont généralement intégrées à l'une de ces quatre étapes) :

- Tout d'abord, il faut sélectionner un scan qui servira de référence (généralement, celui acquis en premier sur les deux considéré) et sélectionner des points pour la mise en correspondance. Effectivement, entre les deux scans, il n'y a qu'un recouvrement partiel la plupart du temps. Ainsi, au niveau de la sélection des points, on trouve comme approche l'utilisation de tous les points disponibles, qui est proposée par Besl et Mckay [Besl et McKay, 1992], ou des échantillonnages uniformes ou aléatoires. Il est aussi possible dès cette première étape de rejeter certains points, notamment ceux présents sur les bords du scan.

- Ensuite, on va chercher à mettre en correspondance les points sélectionnés avec d'autres points du deuxième scan considérés. On trouve la méthode de base proposée par Besl et Mckay, qui est de chercher le point le plus proche sur l'autre scan : en effet, comme indiqué plus haut, on suppose que les deux scans sont très proches. Une alternative est la métrique normal-shooting, proposée par Chen et Medioni [Chen et Medioni, 1991], et qui consiste à « suivre » la normale du point que l'on cherche à apparier pour avoir l'intersection avec le deuxième scan. Cette métrique fonctionne moins bien sur les structures un peu complexes.
- L'étape suivante voit le rejet de certaines paires créés. En effet, cette étape permet d'améliorer la convergence de l'algorithme : avec trop de paires sélectionnées, la convergence sera lente et l'algorithme sera plus sensible au bruit. Ainsi, une première idée est de rejeter les paires présentes sur les « bords » des scans : c'est par exemple ce que fait Turk [Turk et Levoy, 1994], afin de réduire le nombre de faux appariements. D'autres approches consistent à rejeter par exemple les paires dont la distance entre les deux points est trop élevée, mais cette condition est généralement intégrée au processus d'appariement de points entre deux scans, ou encore le rejet d'un certain pourcentage de paires. Toutefois, lorsque le déplacement entre deux scans est élevé, mis à part le rejet des paires sur les bords, il n'y a pas d'améliorations visibles.
- Enfin, la dernière étape consiste en le choix d'une métrique de correspondance entre scans à minimiser : l'idée est que plus les scans seront correctement mis en correspondance, plus cette distance sera faible. Au niveau des métriques pour les distances, on en trouve deux utilisées principalement : la distance point à point, présentée en équation (3.3), et la distance point à plan, présentée en équation (3.4).

Pour la distance point à point, des solutions analytiques existent pour la minimisation de la fonctionnelle présentée en équation 3.2, les deux plus utilisées étant la décomposition en SVD, introduite par Arun et utilisée par exemple dans [Nüchter *et al.*, 2007], et résolution avec une représentation des rotations par des quaternions, introduite par Horn [Horn, 1987] et par exemple utilisée dans [Nüchter *et al.*, 2004]. La méthode par SVD a l'avantage d'être robuste et facile à implémenter, tandis que les quaternions sont plus adaptés pour la représentation des rotations dans l'espace. Pour la distance point à plan, il n'est pas possible de la résoudre linéairement. Deux possibilités sont alors à envisager : utiliser des méthodes de descente de gradient, type Gauss-Newton ou Levenberg-Marquardt, ou sinon linéariser la fonctionnelle que l'on cherche à minimiser, notamment au niveau des matrices de rotations. Andrea Censi [Censi, 2008], dans son article, reprend cette métrique en la linéarisant, ce qui a pour effet d'accélérer l'appariement de points de manière non négligeable. Enfin, Zhu [Zhu *et al.*, 2009] proposent une amélioration de la distance point à plan, qui est selon les auteurs plus robuste pour la mise en correspondance de points entre scans. En effet, là où la métrique de Censi n'est pas robuste au large déplacement entre scans, celle-ci, tout en conservant la rapidité de la métrique point à plan, donne de meilleurs résultats. Le plus souvent, c'est la distance point à plan qui est utilisée car elle donne une meilleure convergence que la distance point à point, comme le montrent Rusinkiewicz et Levoy [Rusinkiewicz et Levoy, 2001] dans leur comparaison de différentes versions de l'algorithme d'ICP.

L'étape qui n'est pas détaillée ici est celle de l'application d'un poids aux paires créées lors de l'appariement. En effet, d'après Rusinkiewicz et Levoy, cette étape n'influe ni sur le temps de calcul, ni sur la convergence de l'algorithme. De plus, elle peut être intégrée à l'étape de mise en correspondance des points, et n'est généralement pas considérée comme une étape de l'algorithme de l'ICP à part.

3.2.3.2 Accélération de l'algorithme

Enfin, il convient de noter que d'autres améliorations ont été apportées à l'ICP, notamment au niveau du temps de calcul. Effectivement, dans l'algorithme, c'est la mise en correspondance des points qui prend le plus de temps, car, naïvement, on cherche à sélectionner la meilleure correspondance pour un point en testant toutes les correspondances possibles. Plusieurs étapes de l'ICP

qui n'influent pas (ou peu) sur la convergence ont ainsi pu être améliorées, permettant de réduire grandement les temps de calcul. On trouve entre autres :

- Le nombre de points des scans en entrée. Lorsque les scans possèdent un trop grand nombre de points, des échantillonnages sont effectués : comme expliqué dans [Nüchter *et al.*, 2004], une méthode dite de « force brute », qui prendrait en compte tous les points des deux scans et tous les appariements possibles, aurait une complexité en $O(n^2)$. C'est pourquoi dans [Nüchter *et al.*, 2004] et [Nüchter *et al.*, 2007], il utilise des filtres pour réduire le bruit au niveau de ces données et les échantillonner : comme il l'explique, pour un scan laser, lorsque la densité est élevée dans une certaine zone, les points sont remplacés par leur barycentre ; des filtres médians sont aussi appliqués pour échantillonner uniformément les scans.
- D'autres approches concernent le format de données utilisées. En effet, l'utilisation d'un kd-tree pour structurer les données et accéder aux plus proches voisins (pour l'appariement, qui est l'étape la plus lente de l'algorithme d'ICP) rapidement est devenue indispensable pour envisager des applications temps réel sur des jeux de données volumineux. La recherche des plus proches voisins se fait par approximation, à l'aide de l'algorithme de recherche **Approximate Nearest Neighbour** (ANN). En effet, l'arbre est parcouru. Ainsi, on trouve Nüchter [Nüchter *et al.*, 2007] qui utilise dans son article le **Cached kd-tree** : la nouveauté par rapport au kd-tree est qu'entre chaque itération, la recherche du plus proche voisin n'est pas entièrement faite. Des pointeurs vers les feuilles visitées sont gardés en mémoire, et constituent le point de départ de la recherche, ce qui a pour effet d'accélérer la recherche des plus proches voisins. Choi [Choi *et al.*, 2012] propose lui aussi une modification des kd-trees, en utilisant les **Approximate cached kd-trees**, qui ont les avantages des **Approximate kd-trees** et des **Cached kd-trees**, et qui selon lui seraient 24 fois plus rapide que les kd-trees standards.

3.2.3.3 Amélioration de l'ICP

A côté des modifications qui ne concernent qu'une étape de l'ICP, on trouve des modifications un peu plus importantes. Ainsi, Fabrice Monnier [Monnier *et al.*, 2013] propose une modification de l'algorithme d'ICP afin de chercher une transformation non rigide entre deux nuages de points : l'objectif donné par l'auteur est de mettre à jour des bases de données existantes, notamment en termes de précision, niveau de détails et diversité d'objets représentés. La non-rigidité de la transformation vient du fait qu'au cours du temps, l'erreur de positionnement de la centrale inertielle évolue de manière non linéaire : même lorsque le véhicule est à l'arrêt, l'erreur augmente tout de même.

D'autre part, on trouve des approches de type « n-scan matching », où le but n'est plus seulement d'aligner deux scans consécutifs entre eux, et remonter de proche en proche sur l'ensemble des scans, mais de contraindre l'ensemble des scans à être alignés en une seule passe. C'est ce qu'ont introduits Lu et Millios [Lu et Milios, 1997], dans le but de réduire l'erreur d'alignement entre scans qui s'accumule au cours de l'acquisition. Leur approche est effectuée en post-traitement. Dans le même style, on trouve un article de Nüchter plus récent, [Nüchter *et al.*, 2010], où une linéarisation de rotations est effectuée dans une approche de n-scans matching, et ceci dans le but d'accélérer la procédure, mais aussi de pouvoir appliquer une solution analytique pour résoudre l'équation du système considéré.

Une dernière modification assez importante, qui permet d'améliorer l'alignement des scans entre eux est la fermeture de boucle, ou plus généralement l'ajustement de faisceaux. En effet, la fermeture de boucle consiste à revenir à un emplacement déjà visité, à scanner l'environnement et à détecter ce bouclage sur les acquisitions pour réduire l'erreur d'alignement accumulée lors des précédents alignements. En effet, comme l'explique Gérossier dans sa thèse [Gérossier, 2012], en repérant une fermeture de boucle entre deux scans, l'ensemble des alignements effectués avant le bouclage peuvent être affinés.

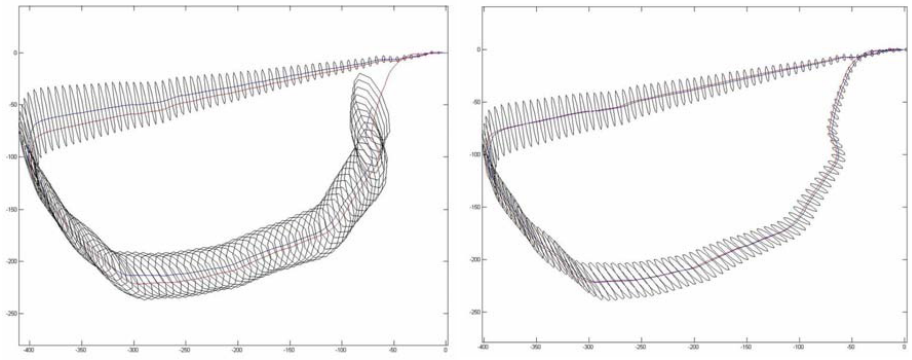


FIGURE 3.4 – Comparaison des incertitudes sur les mesures [Gérossier, 2012]

Sur la figure 3.4, les ellipses représentent les incertitudes sur les mesures. A gauche, on a les incertitudes avant la fermeture de boucle, et à droite après : on voit bien que la fermeture de boucle permet d'améliorer la précision des mesures. L'ajustement de faisceaux, qui vient du domaine de la photogrammétrie et qui a pour but d'affiner les coordonnées des points 3D obtenus par reconstruction 3D à partir des images en repérant les recouvrements de données entre images, est tout aussi applicable avec des données issues de capteur laser 2D ou 3D. On trouve ainsi Lu et Milios [Lu et Milios, 1997], qui, dans leur approche de recalage global, effectuent de l'ajustement de faisceaux.

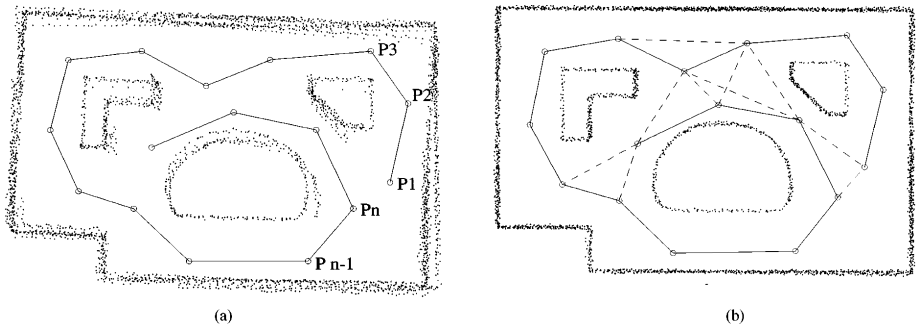


FIGURE 3.5 – Recalages successifs de scans et recalage global [Lu et Milios, 1997]

Sur la figure 3.5, la figure de gauche donne la carte obtenue par recalage successifs de scans entre deux positions consécutives, et la figure de droite présente la même carte, mais où le recalage des scans a été effectué de manière globale. On voit très bien que l'ajustement de faisceaux, au même titre que la fermeture de boucle qu'il généralise, permet d'affiner les mesures et améliore considérablement les cartes. C'est aussi ce qu'utilise Nüchter [Nüchter *et al.*, 2007], pour affiner l'alignement entre les scans, en comparant le dernier scan acquis avec les précédents, mais sous certaines conditions pour ne pas avoir à tester l'ensemble des scans disponibles, ou plus récemment [Nüchter *et al.*, 2010] avec son approche de n-scan matching. Dans cette approche de n-scan matching, l'ensemble des scans qui ont un pourcentage de recouvrement suffisant sont alignés, ce qui permet de détecter automatiquement les recouvrements lorsque deux scans se superposent en partie. Newman [Newman *et al.*, 2006] propose dans son article une méthode qui permet de détecter les recouvrements à partir d'images, approche plus intuitive qu'avec des nuages de points car les points d'intérêts sont extraits plus facilement des images que des nuages. De plus, il détecte et élimine les fausses alarmes en prenant en compte divers paramètres, comme l'écart temporel entre les deux images par exemple.

3.3 Méthode d'optimisation proposée

Dans la section 3.2.2, nous avons introduit 2 méthodes qui sont proches de nos travaux, notamment par rapport à l'orientation que nous avons pris pour l'optimisation des paramètres extrinsèques. La première, présentée dans [Levinson et Thrun, 2010], permet d'optimiser les 6 paramètres extrinsèques d'un système LIDAR multi-fibres séquentiellement et itérativement : l'optimisation se fait en post-traitement, sans mire de calibrage et en se basant sur l'observation qu'avec des paramètres corrects, les données issues de chaque fibre du capteur se superposent comme l'illustre la figure 3.6, et en utilisant une hypothèse de monde plan localement - qui est vraie au vu de la densité de points fournis par le capteur -, les auteurs cherchent à minimiser une fonctionnelle qui pénalise les points éloignés des plans locaux. La résolution est de la forme « grid search », et l'optimisation est séparée pour les paramètres de rotations et de translations : à chaque itération, une énergie est calculée en modifiant chaque paramètre dans un voisinage de l'estimation initiale et selon un pas donné, qui diminuent à chaque itération pour converger vers les paramètres « optimaux ».
Le deuxième article proche de nos travaux est [Mengwen *et al.*, 2014], où un système composé de plusieurs LIDARs est calibré en deux temps : en premier, un LIDAR de référence est étalonné en effectuant plusieurs acquisitions et en recalant les différentes données entre elles. Pour cela, des caractéristiques planaires sont extraites manuellement des nuages et mis en correspondance en jouant sur les paramètres extrinsèques. Ensuite, les autres LIDARs sont étalonnés en recalant chaque nuage de points par rapport au nuage issu du LIDAR de référence, en extrayant aussi manuellement des caractéristiques planaires : par conséquent, chaque LIDAR à son propre calibrage extrinsèque par rapport au véhicule mobile, et pour les résultats, une initialisation proche de la vérité terrain est choisie.

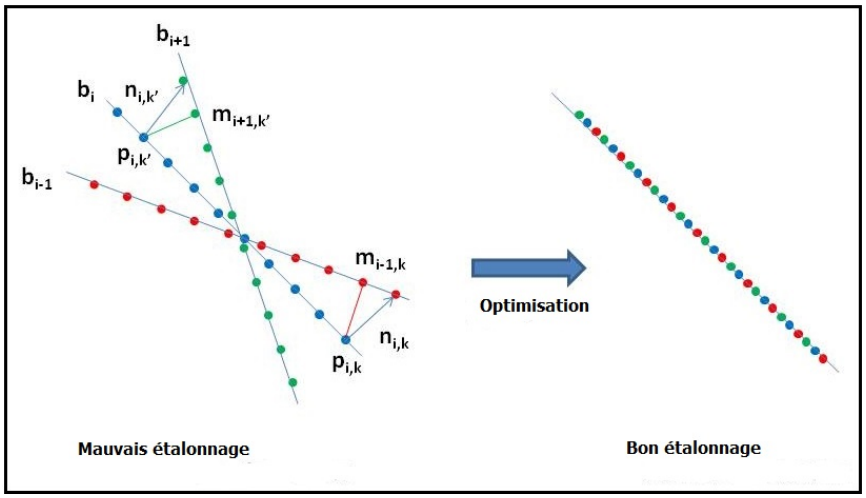


FIGURE 3.6 – Vue de côté d'une surface planaire acquise avec un LIDAR multi-fibres

L'optimisation que nous avons choisi d'effectuer et qui est différente des 2 algorithmes que l'on vient de présenter pour répondre à nos besoins. En effet, nous cherchons aussi à optimiser les 6 paramètres de calibrage extrinsèque d'un capteur LIDAR multi-couches, mais au contraire des 2 méthodes précédentes, notre méthode est automatique, même lorsque nous effectuons une extraction de caractéristiques planaires comme cela est expliqué en section 3.3.5. Notre méthode d'optimisation est rapide, et robuste à une mauvaise initialisation des paramètres de calibrage. Pour effectuer nos expérimentations, nous avons utilisé des jeux de données obtenus à l'aide d'un véhicule mobile équipé d'un capteur LIDAR Velodyne 32 fibres, présenté en figure 2.6. Le capteur LIDAR permet d'obtenir jusqu'à 700 000 points à la seconde, et couvre un champ vertical de 40°(de -8°à +32°) et un champ horizontal de 360°, puisque la base du capteur tourne à une fréquence pouvant aller jusqu'à 10 Hz. Dans la suite de ce chapitre, nous supposons que l'étalonnage intrinsèque du capteur

est effectué et donne des valeurs correctes et précises, et que l'ensemble des capteurs embarqués sur le véhicule permettent d'avoir le positionnement précis du véhicule à chaque instant de contrôle voulu.

3.3.1 Présentation de l'algorithme mis au point

Notre point de départ est le même que celui introduit dans [Levinson et Thrun, 2010] : nous avons un nuage de point obtenu après une acquisition avec notre véhicule mobile, et les valeurs des paramètres de calibrage extrinsèque qui sont utilisés pendant l'acquisition sont erronées, ce qui donne un nuage de point bruité et difficilement exploitable. Nous avons choisi de définir une fonctionnelle à minimiser, similaire à du recalage de données avec une distance point à plan : en effet, comme le montre la figure 3.6, avec un bon calibrage, les données issues de fibres voisines d'un capteur multi-fibres devraient être superposées, et un recalage de données issues de fibres voisines permet d'obtenir ce résultat en optimisant les paramètres extrinsèques. L'optimisation se fait en post-traitement, de façon non supervisée et sans mire de calibrage, et vise à corriger les paramètres de calibrage pour un nuage de points. L'énergie que l'on veut minimiser est définie par l'équation (3.6) :

$$J(R,T) = \frac{\sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_{k} w_{i,j,k} * d_{i,j,k}^{2}(R,T)}{\sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_{k} w_{i,j,k}} \tag{3.6}$$

avec :

$$\begin{cases} d_{i,j,k}(R,T) = n_{i,k}^{T} \times (p_{i,k}(R,T) - m_{j,k}(R,T)) \\ R(\alpha,\beta,\gamma) \text{ et } T(t_x,t_y,t_z) \\ p_{i,k}(R,T) = R_{nav}(p'_{i,k}) \times (R \times p'_{i,k} + T) + T_{nav}(p'_{i,k}) \\ m_{j,k}(R,T) = R_{nav}(m'_{j,k}) \times (R \times m'_{j,k} + T) + T_{nav}(m'_{j,k}) \end{cases}$$

Dans l'énergie définie en (3.6), les différents termes utilisés sont :

- B est un sous-ensemble des fibres du capteur multi-fibres, avec $B \subset [\![0;31]\!]$
- 2*N représente le nombre de fibres voisines à la fibre i que l'on prend en compte pour le recalage
- k itère sur un sous-ensemble des points acquis par la fibre i
- $w_{i,j,k}$ est un poids dont la valeur vaut 1 en fonction d'un seuil de distance entre les points $p_{i,k}$ et $m_{j,k}$.
- $n_{i,k}$ est la normale au point $p_{i,k}$ du plan tangent au même point $p_{i,k}$.
- $p_{i,k}$ et $m_{j,k}$ sont respectivement le k^{me} point de la fibre i, projeté dans le repère lié à la Terre et son plus proche voisin sur la fibre j, aussi projeté dans le même repère.
- $p'_{i,k}$ et $m'_{j,k}$ sont respectivement le k^{me} point de la fibre i, projeté dans le repère cartésien du capteur et son plus proche voisin sur la fibre j, aussi projeté dans le même référentiel.
- R_{nav} et T_{nav} sont respectivement la matrice de rotation et le vecteur de translation décrivant la transformation entre le repère body du véhicule et le repère monde. Ces transformations sont calculées pour chaque point acquis.
- L'énergie que l'on a définie a un sens physique : nous avons une somme de distances au carré, pondérée par la somme des poids pris en compte, qui dans le cas de l'optimisation des paramètres de calibrage extrinsèque est égal au nombre de points N_t pris en compte dans le calcul de l'énergie. Nous supposons que le bruit du nuage provient de plusieurs sources indépendantes, et qu'il est centré, réduit et que sa distribution suit une loi normale : avec ces hypothèses, l'énergie J suit une loi du χ^2, et lorsque N_t est suffisamment grand - ce qui est le cas avec le capteur utilisé -, l'énergie J donne une estimation de la variance σ^2 du bruit total du nuage.

3.3.2 Optimisation de l'énergie

3.3.2.1 Approximation linéaire

Pour notre optimisation, nous partons de paramètres extrinsèques qui ne sont pas corrects, ou plutôt qui donnent un nuage de « mauvaise » qualité, et que nous voulons optimiser en minimisant l'énergie définie par (3.6). Nous ne sommes pas capable de trouver la solution optimale car l'énergie est non linéaire à cause des paramètres de rotation. Nous avons changé le problème en une minimisation itérative où nous cherchons des biais à ajouter à ces paramètres extrinsèques : nous partons de paramètres connus $(t_x, t_y, t_z, \alpha, \beta, \gamma)$, et nous cherchons les variations $(\delta t_x, \delta t_y, \delta t_z, \delta\alpha, \delta\beta, \delta\gamma)$, qui donnent le résultat suivant :
$\mathrm{J}(\mathrm{R}(\alpha+\delta\alpha, \beta+\delta\beta, \gamma+\delta\gamma),\ \mathrm{T}(t_x+\delta t_x, t_y+\delta t_y, t_z+\delta t_z)) < \mathrm{J}(\mathrm{R}(\alpha,\beta,\gamma),\ \mathrm{T}(t_x,t_y,t_z))$

La matrice $\mathrm{R}(\alpha+\delta\alpha, \beta+\delta\beta, \gamma+\delta\gamma)$ est remplacée par une approximation linéaire $\mathrm{R}(\alpha,\beta,\gamma) + R_\alpha * \delta\alpha + R_\beta * \delta\beta + R_\gamma * \delta\gamma$, en supposant que les variations que l'on cherche sont faibles ; des intermédiaires de calculs sont présentés en Annexe B. Avec cette approximation dans l'équation (3.6), on peut réécrire l'énergie J sous la forme :

$$J(\delta X) = \frac{\sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_k w_{i,j,k} * (D_{i,j,k} + C_{i,j,k}^T \times \delta X + o(\delta X))^2}{\sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_k w_{i,j,k}} \tag{3.7}$$

avec :

$$\begin{cases} \delta X = (\delta t_x \quad \delta t_y \quad \delta t_z \quad \delta\alpha \quad \delta\beta \quad \delta\gamma)^T \\ D_{i,j,k} = n_{i,k}^T \times \begin{pmatrix} T_{nav}(p'_{i,k}) - T_{nav}(m'_{j,k}) \\ + \left[R_{nav}(p'_{i,k}) \times (R(\alpha,\beta,\gamma) \times p'_{i,k} + T(t_x,t_y,t_z))\right] \\ - \left[R_{nav}(m'_{j,k}) \times (R(\alpha,\beta,\gamma) \times m'_{j,k} + T(t_x,t_y,t_z))\right] \end{pmatrix} \\ C_{i,j,k} = \begin{pmatrix} n_{i,k}^T \times \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] \times \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \\ n_{i,k}^T \times \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] \times \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \\ n_{i,k}^T \times \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \\ n_{i,k}^T \times \left[R_{nav}(p'_{k,i}) \times R_\alpha \times p'_{i,k} - R_{nav}(m'_{j,k}) \times R_\alpha \times m'_{j,k}\right] \\ n_{i,k}^T \times \left[R_{nav}(p'_{i,k}) \times R_\beta \times p'_{i,k} - R_{nav}(m'_{j,k}) \times R_\beta \times m'_{j,k}\right] \\ n_{i,k}^T \times \left[R_{nav}(p'_{i,k}) \times R_\gamma \times p'_{i,k} - R_{nav}(m'_{j,k}) \times R_\gamma \times m'_{j,k}\right] \end{pmatrix} \end{cases}$$

La solution qui minimise l'énergie (3.7) est la solution du système linéaire suivant :

$$C \times \delta X = -V \tag{3.8}$$

avec :

$$\begin{cases} C = \sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_k w_{i,j,k} * (C_{i,j,k} \times C_{i,j,k}^T) \\ V = \sum_{i=1}^{B}\sum_{j=i-N}^{i+N}\sum_k w_{i,j,k} * (D_{i,j,k} \times C_{i,j,k}) \end{cases}$$

3.3.2.2 Approche d'optimisation

Pour trouver les paramètres extrinsèques optimaux, et comme nous avons linéarisé certains paramètres, nous calculons δX de façon itérative, jusqu'à convergence des paramètres de calibrage.

A chaque nouvelle itération n+1, les paramètres extrinsèques sont définis comme étant :

$$\begin{cases} t_{x_{n+1}} = t_{x_n} + \delta t_{x_n} \\ t_{y_{n+1}} = t_{y_n} + \delta t_{y_n} \\ t_{z_{n+1}} = t_{z_n} + \delta t_{z_n} \\ \alpha_{n+1} = \alpha_n + \delta\alpha_n \\ \beta_{n+1} = \beta_n + \delta\beta_n \\ \gamma_{n+1} = \gamma_n + \delta\gamma_n \end{cases} \tag{3.9}$$

Pour démarrer l'optimisation, nous avons des paramètres initiaux $(t_{x_0}, t_{y_0}, t_{z_0}, \alpha_0, \beta_0, \gamma_0)$ plus ou moins proches des paramètres optimaux, et à chaque itération, nous résolvons la fonction objectif (3.7). Nous montrerons dans la section 3.4 que les paramètres initiaux ne doivent pas nécessairement être proches de la solution optimale, et que notre optimisation des paramètres extrinsèques est robuste à une grande erreur sur les valeurs de paramètres obtenus par étalonnage.
L'algorithme complet d'optimisation des paramètres extrinsèques est présenté dans l'algorithme 1. Le critère d'arrêt que l'on a choisi concerne les variations des paramètres δXentre deux itérations : l'optimisation s'arrête lorsque $\|\delta X\|_{max}$ est inférieur à un seuil δ, ou dans certains cas si le nombre d'itérations est trop élevé.

Algorithme 1 Optimisation linéaire itérative des paramètres extrinsèque

Data: Un nuage de point avec des paramètres extrinsèques initiaux, choisis arbitrairement, et une trajectoire du véhicule connue
Result: Un nuage pour lequel lequel les paramètres de calibrage extrinsèque sont optimisés
Lecture et sous-échantillonnage du nuage de point ;
Paramètres extrinsèques initiaux ;
repeat
  Projection des points acquis dans le repère global avec les paramètres extrinsèque actuels $(t_{x_n}, t_{y_n}, t_{z_n}, \alpha_n, \beta_n, \gamma_n)$;
  Sélection d'un ensemble de points $p_{i,k}$;
  Construction des paires de points $p_{i,k}$ appartenant à la fibre i et $m_{j,k}$, leur plus proche voisin appartenant à une des fibres voisines j ;
  Calculs des normales aux plans en chaque point $p_{i,k}$;
  Construction de l'équation (3.8), et résolution qui permet d'obtenir les variations $(\delta t_{x_n}, \delta t_{y_n}, \delta t_{z_n}, \delta\alpha_n, \delta\beta_n, \delta\gamma_n)$;
until $\|\delta X\|_{max} < \delta$, *ou* $n_{iter} \geq n_{max}$;
Enregistrement des paramètres extrinsèque obtenus par optimisation ;

3.3.3 Validation du résultat de l'optimisation

Pour valider le résultat d'optimisation, l'énergie doit diminuer et atteindre un minimum global, ce qui dans le cas du recalage de données n'est pas trivial, notamment lorsque l'initialisation est un peu éloignée du résultat global. Aussi, un des résultats attendu est que notre valeur d'énergie en fin d'optimisation soit inférieure à sa valeur avant optimisation, mais pour valider l'optimisation, la valeur de l'énergie doit aussi être inférieure à un seuil donné : comme nous l'avons expliqué dans la section 3.3.1, notre énergie suit une distribution du χ^2, et lorsque le nombre de points N_t pris en compte dans son calcul est suffisamment grand, l'énergie donne un estimateur du bruit du nuage. Un seuil de validation pour cette énergie à 97% est $3\sigma^2$, avec σ^2 la valeur de l'énergie. Par exemple, en considérant un nuage provenant d'une acquisition réelle, et si l'on veut un nuage de bonne qualité, comme le bruit provient de plusieurs sources telles que l'acquisition même, la centrale inertielle ou encore le GPS, un bruit de déviation standard inférieur à 5cm est acceptable ; cela donne un seuil

pour la valeur finale de l'énergie J d'environ 75 cm^2, qui est le seuil que l'on va prendre en compte pour la validation des résultats d'optimisation sur des jeux de données réelles.

3.3.4 Précision des paramètres de calibrage extrinsèque

Avec les tests qui vont être présentés en section 3.4, nous allons voir que l'optimisation des paramètres extrinsèques donne de bons résultats à quelques exceptions près : dans la littérature concernant l'optimisation des paramètres de calibrage extrinsèque, le résultat est en général comparé à une vérité terrain. Cela peut prendre du temps de construire correctement cette vérité terrain, notamment dans le cas d'acquisitions réelles où l'on n'est pas assuré d'avoir une vérité terrain correcte. C'est pour cela qu'en plus du critère de validation défini en section 3.3.3, nous avons aussi défini un paramètre de précision concernant les valeurs de paramètres extrinsèques optimisés ; la matrice C de l'équation (3.8) peut-être vue comme une matrice de covariance des biais δX que l'on calcule à chaque itération, comme expliqué dans [Press *et al.*, 1992] dans le cas d'un système des moindres carrés linéaires, et elle nous sert à donner la « précision » des paramètres obtenus en fin d'optimisation, que l'on peut considérer comme un indice de confiance sur les valeurs obtenues :

$$\begin{cases} \text{Pour les translations : } \sigma_x(m) = \sqrt{(C^{-1})_{1,1}} \\ \text{Pour les rotations : } \sigma_\alpha(rad) = \sqrt{(C^{-1})_{4,4}} \end{cases} \tag{3.10}$$

Les précisions restantes pour les paramètres de translations et de rotations sont définies de la même manière. Nous verrons dans les sections 3.4 et 3.5 que selon le type de résultat souhaité, la précision des paramètres finaux est importante. En effet, ce qui nous intéresse est l'affinement des nuages, et le critère qui valide ou non cet affinement est lié la valeur finale de l'énergie ; toutefois, selon le type de jeux de données et de la trajectoire du véhicule, le critère de l'énergie peut être validé, mais la précision peut être mauvaise pour un ou plusieurs paramètres, ce qui poserait problème dans le cas où le but recherché avec notre algorithme est de se rapprocher de la vérité terrain des paramètres de calibrage extrinsèque, que l'on va supposer non connue pour nos tests car comme expliqué précédemment pour des jeux de données réels, elle est difficile à obtenir et peu fiable dans le cas où les paramètres sont mesurés à « la main ».

3.3.5 Définition des poids de la fonctionnelle à minimiser

Les poids $w_{i,j,k}$ définis dans la fonctionnelle (3.6) valent par défaut 1 ou 0 selon que la distance entre les deux points $p_{i,k}$ et $m_{j,k}$ soit inférieure à un seuil ou non. Dans [Demantke *et al.*, 2011], des attributs de dimensionnalité sont présentés avec une façon particulière de les calculer. Les attributs de dimensionnalité sont trois valeurs qui peuvent s'écrire sous la forme :

$$\begin{cases} a_{1D} = \frac{\sigma_1 - \sigma_2}{\sigma_1} \\ a_{2D} = \frac{\sigma_2 - \sigma_3}{\sigma_1} \\ a_{3D} = \frac{\sigma_3}{\sigma_1} \end{cases} \tag{3.11}$$

Pour calculer ces attributs pour un point du nuage, un voisinage du point est choisi, puis une analyse en composante principale est effectuée sur ce voisinage. Les valeurs propres obtenues sont classées par ordre décroissant, et numérotées de 1 à 3 : les σ_1 à σ_3 de l'équation 3.11 représentent les racines carrés de ces valeurs propres ordonnées. Les attributs a_{iD} ensuite calculés permettent de décrire la nature de la surface à laquelle le point concerné appartient : si a_{1D} a la plus grande valeur, cela signifie que le point appartient à un objet plutôt linéaire comme un poteau, et la dimensionnalité du point est alors de 1 ; si a_{2D} est le plus élevé, cela signifie que le point appartient à une surface

planaire, comme le sol ou une façade par exemple, et la dimensionnalité du point est alors de 2 ; enfin, si c'est a_{3D} qui est le plus élevé, le point devrait appartenir à un objet volumique, comme le feuillage d'un arbre, et la dimensionnalité est alors de 3. De plus, ces paramètres sont construits de sorte que leur valeur soit comprise entre 0 et 1 : nous avons donc choisi de les utiliser comme poids dans notre optimisation, puisque notre optimisation utilise une distance point à plan pour effectuer le recalage des données, et que l'on peut donner plus ou moins de poids aux associations avec ces attributs, selon que les points appartiennent à des surfaces planaires. Nous montrerons dans la section 3.4.4 que l'utilisation des attributs de dimensionnalité permet de réduire la valeur de l'énergie en fin d'optimisation.

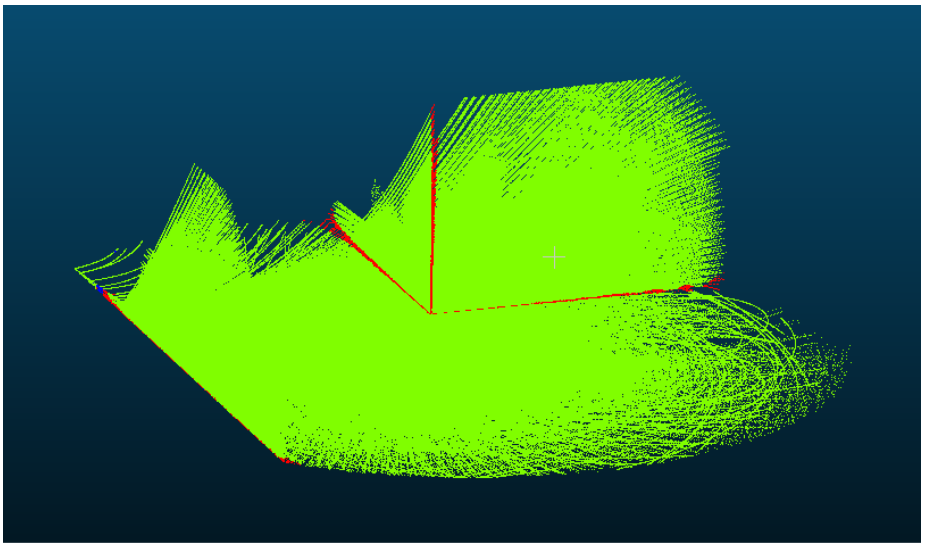


FIGURE 3.7 – Dimensionnalité calculée pour un nuage simulé : en vert, la dimensionnalité vaut 2, et en rouge elle vaut 3

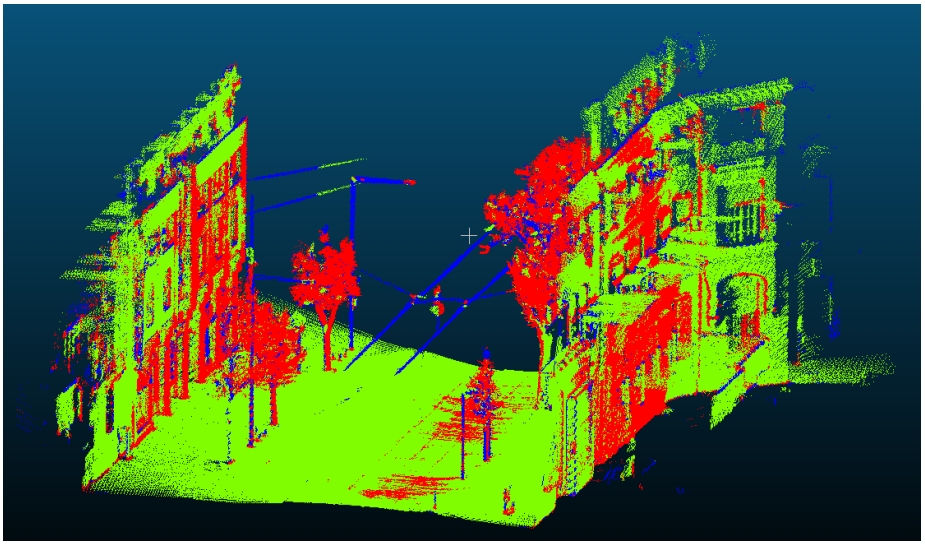


FIGURE 3.8 – Dimensionnalité calculée pour un nuage réel : en bleu, la dimensionnalité vaut 1 ; en vert, elle vaut 2, et en rouge elle vaut 3

Les figures 3.7 et 3.8 montrent la répartition des dimensionnalités des points sur un nuage simulé et un nuage issu d'une acquisition réelle. Pour les jeux de données simulés, les points ont une

dimensionnalité de 2 en majorité, sauf sur les bords, ce qui est logique car il s'agit de l'intersection de 2 plans orthogonaux. Pour les jeux de données réelles, la répartition est moins homogène : en effet, on peut voir que l'on a bien le feuillage des arbres avec une dimensionnalité de 3 en majorité, mais certains éléments du sol et des façades ont aussi une dimensionnalité de 3. Cela est dû aux erreurs provenant de différentes sources, et prendre en compte la valeur de l'attribut de dimensionnalité a_{2d} permet de donner un poids plus petit à ce genre de point. C'est ce que l'on appelle la planéité en un point, à savoir la valeur de l'attribut de dimensionnalité a_{2D} calculé en ce point : plus la valeur est proche de 1, plus le point concerné est à même d'appartenir à une surface planaire, et plus le rôle que l'on donne à ce point dans l'optimisation est important. Les poids $w_{i,j,k}$ utilisés qui valent :

$$w_{i,j,k} = \begin{cases} 1 \ si \ ||p_{i,k} - m_{j,k}|| < d_{max} \\ \\ 0 \ sinon \end{cases}$$

deviennent alors en utilisant la planéité en un point :

$$w_{i,j,k} = \begin{cases} max(a_{2D}(p), a_{2D}(m)) \ si \ ||p_{i,k} - m_{j,k}|| < d_{max} \\ \\ 0 \ sinon \end{cases}$$

Enfin, comme lors du calcul de la distance point à plan, 2 points $p_{i,k}$ et $m_{j,k}$ sont concernés, nous avons testé de prendre la valeur minimale entre les deux attributs, ou encore la moyenne, mais c'est la valeur maximale qui permettait d'avoir les meilleurs résultats de convergence.

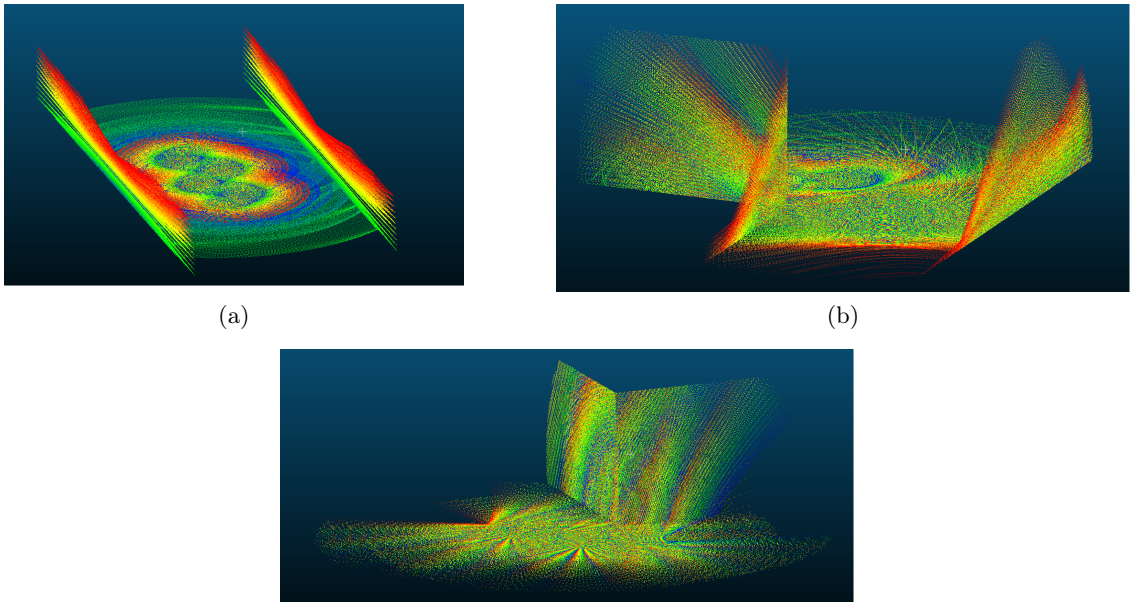
3.4 Résultats expérimentaux

L'optimisation des paramètres extrinsèques a été testée sur plusieurs jeux de données, simulés et issus d'acquisitions réelles. Dans un premier temps, nous allons présenter l'ensemble des jeux de données utilisés pour nos expérimentations, ainsi que les différents paramètres que l'on a dû fixer pour l'optimisation, puis nous présenterons plusieurs résultats d'optimisation obtenus avec les différents jeux de données.

3.4.1 Jeux de données utilisés pour les expérimentations

Les jeux de données simulés sont des nuages qui représentent une acquisition en environnement urbain. L'environnement est composé de plans verticaux (représentant des façades d'immeubles) et horizontaux (représentant le sol, ou la route dans ce cas). Les 3 nuages de points que l'on va présenter ont à peu près le même nombre de points, environ 5 millions de points, et la différence entre ces jeux de données est multiple : l'environnement change, le nombre et la position des plans est différente, mais la trajectoire du véhicule change aussi. Ces jeux de données sont utilisés pour valider notre approche car ils nous fournissent une vérité terrain avec laquelle comparer le résultat d'optimisation. En effet, les paramètres issus de calibrage extrinsèque sont précisément connus pour ces jeux de données, et pour tester notre optimisation, des erreurs ont été ajoutées aux paramètres extrinsèques, et ces erreurs devaient être corrigées par l'optimisation. Les 3 jeux de données simulées ont les propriétés suivantes :

- Le nuage #1, présenté en figure 3.9 a) est composé d'un sol et de 2 plans verticaux. Pour palier certains problèmes d'observabilité qui vont être détaillés en section 3.5, le véhicule a une trajectoire oscillante. Aussi, il n'y a pas de variation d'altitude.
- Le nuage #2 est présenté en figure 3.9 b), et est composé d'un sol, ainsi que de 3 plans verticaux. Le véhicule effectue un virage, et il y a une variation d'altitude.



(a) (b)

(c)

FIGURE 3.9 – Jeux de données simulées utilisées pour les expérimentations : a) jeu de données simulées #1 ; b) jeu de données simulées #2 ; c) jeu de données simulées #3

- Le nuage #3 est présenté en figure 3.9 c), et est assez proche du nuage # 2 : il n'y a que 2 plans verticaux cette fois, et pas de variation d'altitude par contre.

Les 2 jeux de données réelles que l'on a utilisés proviennent de 2 campagnes d'acquisitions différentes, et sont utilisées pour confirmer notre optimisation sur des jeux de données réelles où l'on ne maîtrise pas la valeur des paramètres extrinsèques de départ. Les deux nuages sont les suivants :

— Le nuage #4 est présenté en figure 3.10 a), et provient d'une acquisition effectué à Montbéliard, en France. Le nuage est composé de plusieurs façades, et le véhicule effectue quelques virages ; il y a aussi une légère variation d'altitude. Le nuage est environ composé de 10 millions de points.

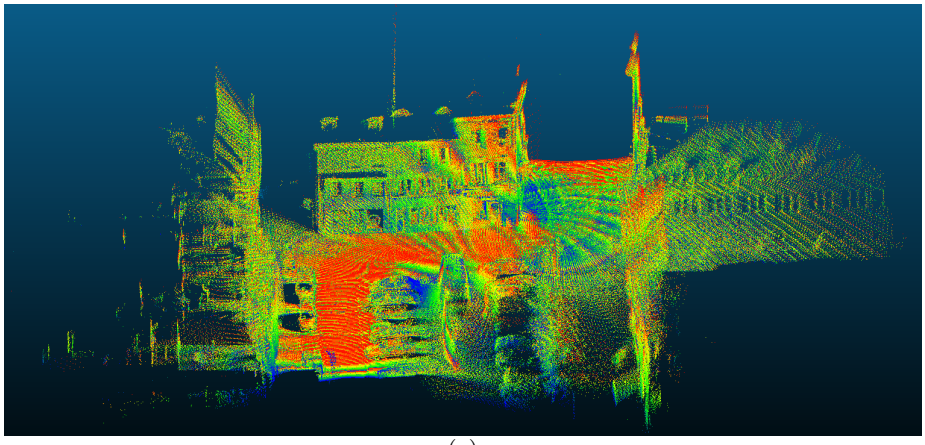
— Le nuage #5 est présenté en figure 3.10 b), et provient d'une acquisition effectué à Dijon, en France. Il n'y a que 2 façades parallèles entre elles, et une légère variation d'altitude, mais pas de virage : par contre, il y a aussi beaucoup de végétation, ainsi que quelques câbles électriques, qui sont des éléments non planaires. Le nuage de points est environ composé de 5 millions de points.

3.4.2 Choix des différents paramètres de l'optimisation

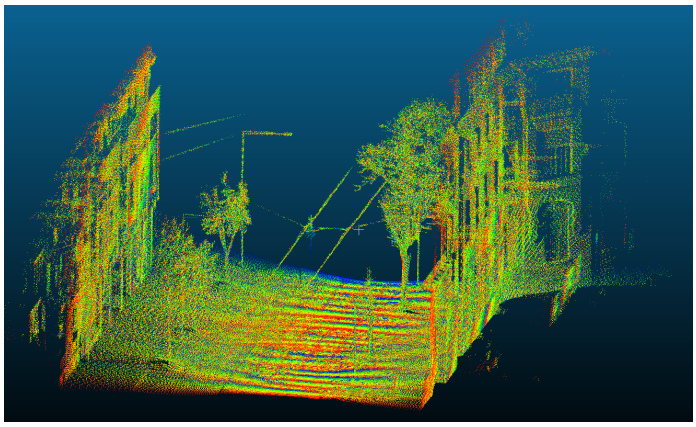
Notre optimisation a été codée en C++. La librairie EIGEN [Eigen library, 2017] a été utilisée pour toutes les opérations matricielles ou vectorielles, et la librairie FLANN [FLANN library, 2017] (Fast Library for Approximated Nearest Neighbor) a été utilisée pour la recherche des plus proches voisins, notamment lors de l'appariement des points ou le calcul des normales. L'algorithme a été exécuté sur un ordinateur avec un processeur de fréquence 3.40 GHz.

De nombreux paramètres avaient besoin d'être fixés dans notre algorithme :

- Pour commencer, nous avons sous-échantillonné les nuages de points à 1 point sur 3, car la



(a)



(b)

FIGURE 3.10 – Jeux de données réelles utilisées pour les expérimentations : a) jeu de données réelles #4 ; b) jeu de données réelles #5

densité des nuages provenant du capteur Velodyne est très élevée, et le sous-échantillonnage permet d'accélérer l'optimisation sans perdre en précision.

- Ensuite, nous avons pris une distance maximale entre deux points appariés $p_{i,k}$ et $m_{j,k}$ de 20cm : plusieurs valeurs ont été testées, et ce seuil donnait les meilleurs résultats d'optimisation. Les poids $w_{i,j,k}$ valent la valeur de l'attribut de dimensionnalité a_{2D} maximum entre les points $p_{i,k}$ et $m_{j,k}$ si le seuil est respecté, 0 sinon.
- Nous faisons du recalage de données entre fibres, et nous avons fixé le nombre de fibres voisines à la fibre b_i à 4, c'est à dire N=2. Une justification du choix de cette valeur est donnée en Annexe B.
- Aussi, il nous a fallu fixer la taille du voisinage pris en compte pour le calcul des normales. Un nombre de 150 voisins a été retenu : le choix de cette valeur est aussi justifié en Annexe B.
- Pour les attributs de dimensionnalité, il a fallu choisir une taille de voisinage pour le calcul des attributs, le nombre de voisins pris en compte influant directement sur la dimensionnalité des points. Dans [Demantke *et al.*, 2011], la taille du voisinage est automatiquement optimisée pour qu'une valeur d'entropie liée aux valeurs des attributs soit la plus petite possible : pour des soucis de temps de calcul, nous avons préféré fixer le nombre de points voisins pour le calcul des attributs. La figure 3.11 présente la répartition des dimensionnalité en fonction du nombre de voisins. On voit que la répartition est correcte à partir d'un voisinage supérieur à 100 points : le sol et les plans verticaux ont des dimensionnalités de 2 en majorité, ce qui est

ce que l'on attend pour des surfaces planaires. Aussi, le temps de calcul des attributs n'étant pas très différent entre un voisinage de 100 et de 150, nous avons fixé le nombre de voisins pris en compte pour le calcul des attributs à 100.

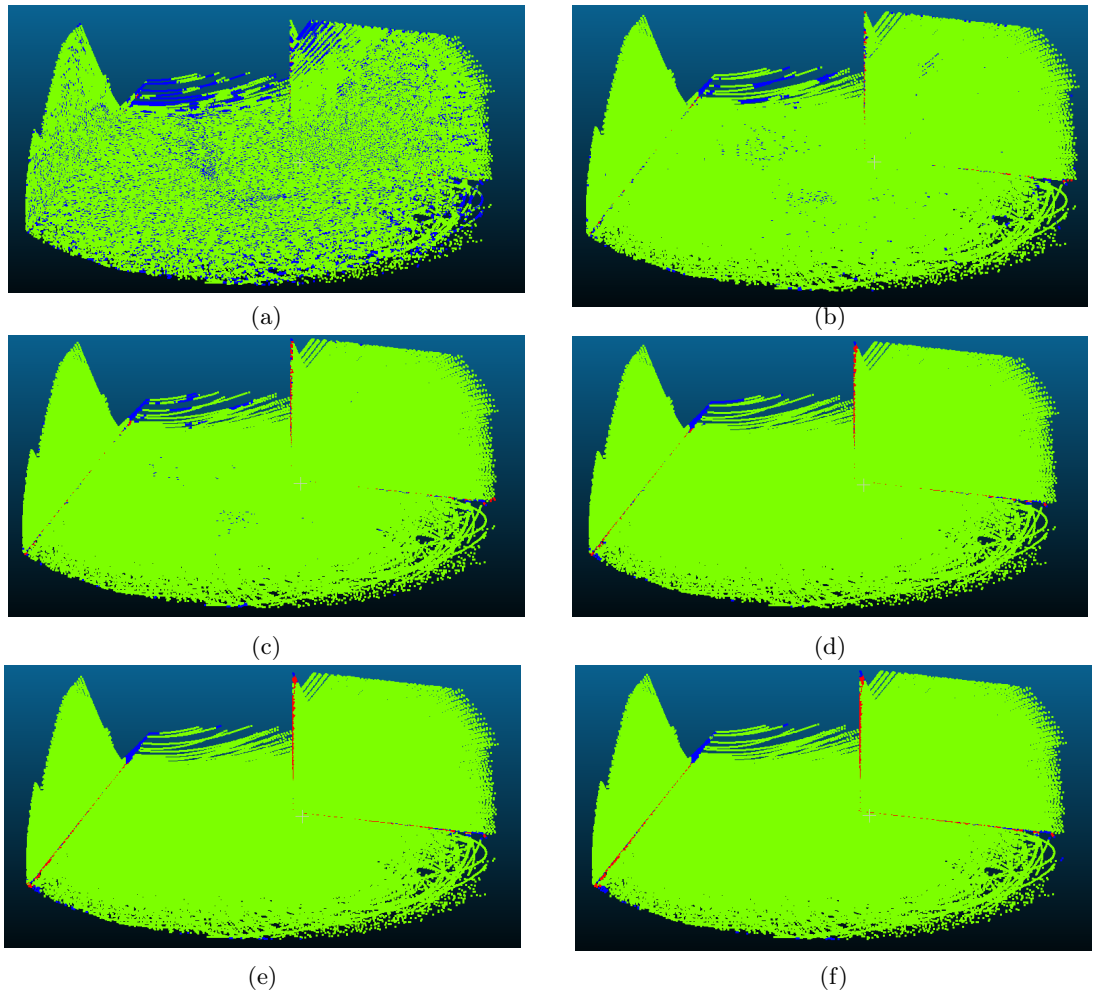


FIGURE 3.11 – Répartition des dimensionnalités en fonction du nombre de voisins pris en compte pour le calcul des attributs : a) 10 points les plus proches ; b) 30 voisins ; c) 50 voisins ; d) 100 voisins ; e) 150 voisins ; f) 200 voisins

- Pour les critères d'arrêts de l'optimisation, nous avons pris un seuil δ de 1cm pour les translations et 0.01° pour les rotations ; l'optimisation s'arrête si l'ensemble des paramètres extrinsèques sont inférieurs à ces seuils. Sinon, le nombre d'itérations maximums a été fixé à 40 ; au cours des expérimentations, nous avons remarqué que l'énergie n'évolue plus de façon significative après un certain nombre d'itérations, mais qu'à cause d'une mauvaise observabilité de certains paramètres pour certains jeux de données, les paramètres continuaient à être modifiés, sans que cela ne change le résultat d'affinement.

Enfin, au niveau des paramètres extrinsèques utilisés pour initialiser notre algorithme, nous avons ajouté des erreurs aux paramètres extrinsèques connus pour les jeux de données simulées, et pour les jeux de données réelles, nous avons pris des paramètres choisis arbitrairement pour l'initialisation de l'optimisation : d'une part, il n'est pas nécessaire d'avoir une initialisation précise des paramètres extrinsèques, et d'autre part, cela à aussi permis de tester la robustesse de notre optimisation à une initialisation qui serait éloignée de la solution optimale.

3.4.3 Comparaison de notre optimisation avec une méthode de l'état de l'art

Dans une première expérimentation, nous avons voulu comparer notre méthode d'optimisation avec une méthode tirée de l'état de l'art. Pour cela, nous avons pris l'optimisation des paramètres extrinsèques présentée dans [Levinson et Thrun, 2010], qui utilise aussi des données provenant d'un capteur Velodyne. Leur méthode d'optimisation est différente de la notre, comme nous l'avons présenté en section 3.3 ; pour comparer leur méthode à la notre, nous avons fixé plusieurs paramètres :

- La taille du voisinage pris en compte dans la recherche des paramètres optimaux a été fixée à + ou - 3 mètres autour des paramètres initiaux, car l'erreur que l'on rajoute aux paramètres de translations est de l'ordre de 3 mètres ; pour les paramètres de rotations, le voisinage était compris entre + ou - 7°, pour les même raisons.
- Le pas de discrétisation pour la recherche des valeurs des paramètres de calibrage optimaux a été fixée à 5, pour réduire le temps de calcul, et le nombre d'itérations choisi est de 10 : à chaque itération, la taille du voisinage dans lequel les recherches sont effectuées est divisée par 2, et avec les tailles de voisinages initiales, au bout de 10 itérations, cela permettait de finir avec une largeur de recherche inférieure au centimètre pour les translations, et inférieure au centième de degré pour les rotations, ce qui correspond aux critères d'arrêt que nous avons défini en section 3.4.2.

Le nuage de point utilisé pour la comparaison est le nuage simulé #1, présenté en section 3.4.1. En fin d'optimisation, on peut voir avec la figure 3.12 que visuellement, les deux méthodes d'optimisations donnent un nuage de points affiné, ce qui montre que les paramètres extrinsèques ont bien été optimisés. Le résultat est meilleur avec notre approche, où les plans verticaux sont correctement reconstruits.
La table 3.1 présente les erreurs que l'on a au niveau des paramètres extrinsèques avant optimisation, puis après optimisation avec la méthode de Levinson et la notre. Avec notre optimisation, les paramètres de calibrage extrinsèque sont très proches de la vérité terrain après optimisation, sauf pour le paramètre de translation z, mais cela n'est pas surprenant car l'observabilité est nulle dans cette direction, le nuage ne présentant pas de variation d'altitude. Avec l'optimisation comparée, les résultats vont dans le même sens, mais sont moins bons, ce qui explique la différence que l'on voit avec la figure 3.12.

La figure 3.13 présente l'évolution des énergies pour les 2 optimisations comparées. Les énergies démarrent à une valeur de 217.41 cm^2, et décroissent jusqu'à une valeur de 7.20 cm^2 pour la méthode comparée, et jusqu'à une valeur de 0.29 cm^2 pour notre optimisation. L'évolution des énergies confirme l'observation faite précédemment, qui est que notre optimisation donne un meilleur affinement que la méthode comparée.

Enfin, un dernier point que l'on peut relever est le temps de calcul : notre optimisation a mis 2 minutes pour converger et donner le résultat final, tandis que la méthode comparée à mis environ 1 heure pour le même nuage : là aussi, notre optimisation est meilleure que la méthode comparée. La figure 3.13 montre que notre optimisation nécessite un plus grand nombre d'itérations que l'optimisation de Levinson, mais le temps de calcul est bien plus faible : en effet, l'optimisation comparée est une recherche exhaustive, pour laquelle l'énergie est calculée à plusieurs reprises à chaque itération.

3.4.4 Optimisation des paramètres extrinsèques

Nous allons maintenant présenter plusieurs résultats d'optimisation des paramètres extrinsèques. Dans un premier temps, nous allons présenter deux résultats avec des jeux de données simulées, puis deux résultats avec des jeux de données réelles. Des comparaisons entre deux approches d'optimisations sont effectuées : en effet, nous cherchons aussi à montrer que l'utilisation des attributs de

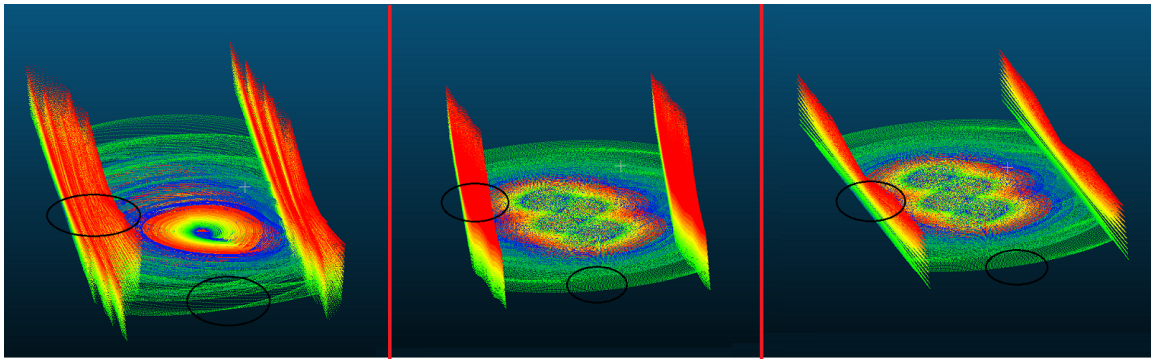


FIGURE 3.12 – Nuage de points simulé #1 : sur la gauche, on a le nuage de points avec des paramètres erronés, avant optimisation ; au milieu, le résultat de l'optimisation avec la méthode de Levinson ; à droite, le résultat d'optimisation avec notre approche. Les 3 nuages sont vus avec la même orientation

	t_x(cm)	t_y(cm)	t_z(cm)	roulis $\alpha(^\circ)$	tangage $\beta(^\circ)$	lacet $\gamma(^\circ)$
Erreur ajoutée aux paramètres extrinsèques	-150.000	250.000	-200.000	5.000	-7.000	-5.500
Différence finale entre la vérité terrain et l'optimisation comparée	6.445	8.594	-500.000	-0.004	0.000	-9.875
Différence entre la vérité terrain et notre optimisation	**0.001**	**-0.012**	-200.000	**0.000**	**0.000**	**0.000**

TABLE 3.1 – Erreurs finales des optimisations des paramètres extrinsèques pour le nuage #1

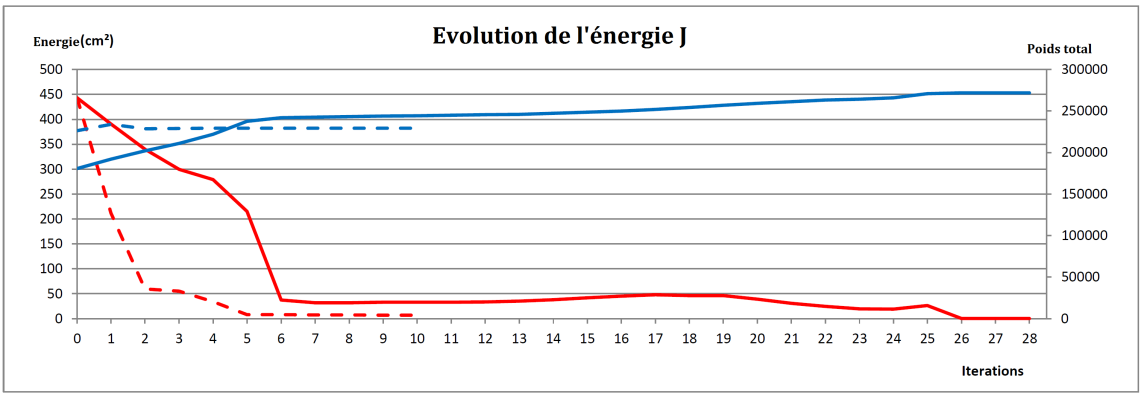


FIGURE 3.13 – Évolution des énergies pour le nuage #1 : en rouge et en trait plein, nous avons notre énergie ; en rouge et en pointillé, nous avons l'énergie de la méthode comparée ; en bleu et en trait plein, nous avons le nombre de points appariés pris en compte dans notre optimisation ; en bleu et en pointillé, le nombre de points appariés pour l'optimisation comparée

dimensionnalité dans les poids utilisés pendant l'optimisation permettent de réduire la valeur finale de l'énergie minimisée. Pour les mises à jour des valeur des attributs de planéité, elles sont effectuées toutes les 7 itérations, afin d'avoir des attributs qui évoluent avec l'affinement des données, et afin de ne pas trop ralentir l'optimisation.

3.4.4.1 Résultats sur des jeux de données simulées

Le premier nuage utilisé pour les tests est le nuage simulé #2. Il y a une variation d'altitude dans le nuage, et le véhicule effectue un virage. La figure 3.14 présente le même nuage de points avec la même orientation pour la visualisation, mais avec deux ensembles de paramètres extrinsèques différentes :

- En haut, le nuage de point est créé avec les paramètres initiaux que l'on donne en entrée à l'optimisation.
- En bas, le même nuage avec les paramètres extrinsèques optimisés

Un seul résultat d'optimisation est présenté car en fin d'optimisation, il n'y a pas de différence visuelle entre les deux nuages affinés avec ou sans l'utilisation des attributs de dimensionnalité. La table 3.2 donne les erreurs ajoutés aux paramètres extrinsèques du nuage de points simulé #2, et donne aussi la différence entre les paramètres extrinsèques optimisés pour le nuage simulé et la vérité terrain. Dans les deux cas d'optimisations, nous avons des paramètres extrinsèques très proches de la vérité terrain en fin d'optimisation, malgré l'initialisation assez éloignée de la vérité terrain : pour les translations, l'erreur finale est inférieure au millimètre, et pour les translations, l'erreur est environ de $1/1000^{\circ}$. Les erreurs sont globalement les plus petites lorsque les attributs de dimensionnalité sont pris en compte. La figure 3.15 donne l'évolution des énergies d'optimisation, avec et sans utilisation des attributs de dimensionnalité : sans les attributs, l'énergie démarre à une valeur de 243.07 cm^2, et diminue jusqu'à atteindre 0.58 cm^2 ; avec les attributs, l'énergie évolue d'une valeur de 86.63 cm^2 à une valeur de 0.46 cm^2. L'énergie est plus faible avec l'utilisation des attributs de dimensionnalité, et la convergence est aussi plus rapide en terme de nombre d'itérations. Comme l'énergie est normalisée, nous donnons aussi sur la même figure l'évolution du facteur de normalisation qui est la somme des poids pour chaque paire de points : avec l'utilisation des attributs, ce facteur augmente considérablement à chaque mise à jour, ce qui illustre le fait que le recalage des données entre fibres se fait correctement.
L'optimisation donne de bons résultats pour chacun des paramètres extrinsèques, et la table 3.3 donne des valeurs de précisions qui vont dans le même sens : en effet, pour chaque paramètre, la précision est très bonne, ce qui va dans le sens des erreurs très faibles par rapport à la vérité terrain décrites dans la table 3.2. Un dernier résultat que l'on peut présenter est le temps de calcul des optimisations : sans l'utilisation des attributs de dimensionnalité, l'optimisation prend 1 minute et 15 secondes pour converger. Avec l'utilisation des attributs de dimensionnalité, l'optimisation prend environ 5 minutes à cause du calcul des attributs de dimensionnalité, mais le temps de calcul reste assez faible par rapport au résultat fourni qui est meilleur que la première optimisation.

	t_x(cm)	t_y(cm)	t_z(cm)	roulis $\alpha(^{\circ})$	tangage $\beta(^{\circ})$	lacet $\gamma(^{\circ})$
Erreurs initiales par rapport à la vérité terrain	-150.00	250.00	-200.00	5.00	-7.00	-5.50
Différence entre la vérité terrain et notre résultat d'optimisation (sans utilisation de la dimensionnalité)	-0.003	-0.016	0.033	-0.000	0.000	0.001
Différence entre la vérité terrain et notre résultat d'optimisation (avec utilisation de la dimensionnalité)	-0.001	-0.023	0.008	-0.000	0.000	0.000

TABLE 3.2 – Erreurs sur les paramètres extrinsèques avant et après optimisation par rapport à la vérité terrain pour le nuage #2

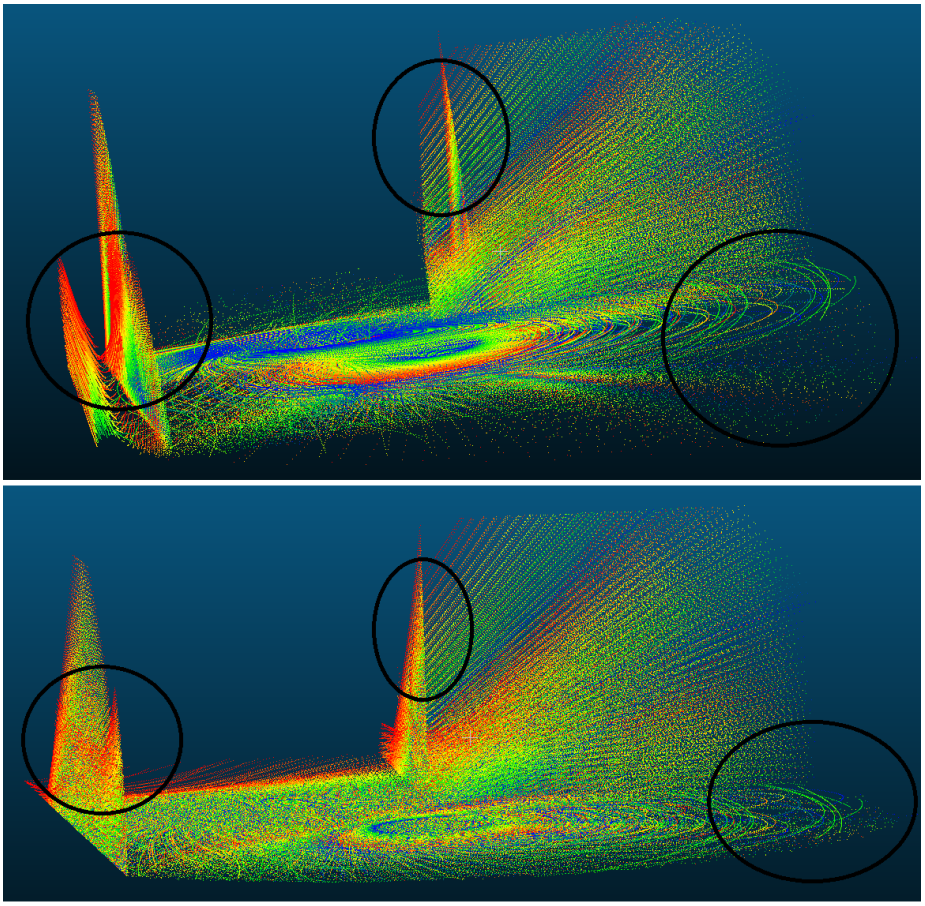


FIGURE 3.14 – Nuage de points simulé #2 : en haut, nuage avec les paramètres extrinsèques initiaux ; en dessous, le résultat de notre optimisation. Les deux images sont vues depuis le même point.

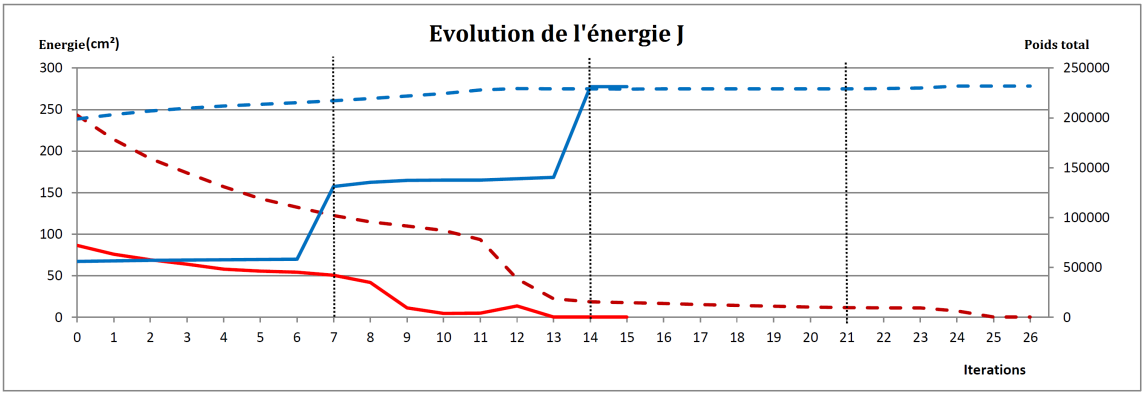


FIGURE 3.15 – Évolution de l'énergie au cours de l'optimisation pour le nuage simulé #2. La ligne rouge pleine représente l'évolution de l'énergie avec l'utilisation des attributs de dimensionnalité ; la ligne rouge pointillé représente la même énergie sans l'utilisation des attributs ; la ligne bleue pleine représente l'évolution du poids total utilisé pour normaliser notre énergie, avec l'utilisation des attributs de dimensionnalité ; la ligne bleue pointillé représente la même donnée, mais sans l'utilisation des attributs de dimensionnalité ; enfin, les lignes noires verticales marquent la mise à jour des attributs de dimensionnalité, toutes les 7 itérations.

	σ_{t_x}(cm)	σ_{t_y}(cm)	σ_{t_z}(cm)	$\sigma_\alpha(^\circ)$	$\sigma_\beta(^\circ)$	$\sigma_\gamma(^\circ)$
Nuage de points simulé #2	1.00	2.18	3.43	0.06	0.04	0.07

TABLE 3.3 – Précision des paramètres extrinsèques pour le nuage de points simulé #2

Un deuxième jeu de données simulées a été utilisé pour valider notre optimisation des paramètres extrinsèques, le nuage #3 : pour les mêmes raisons qu'avec le nuage #2, un seul résultat d'optimisation est présenté. Les résultats sont similaires à ceux présentés pour le nuage #2 ; plus de détails sur les résultats d'optimisation pour le nuage #3 sont donnés en Annexe C.

3.4.4.2 Résultats sur des jeux de données réelles

Dans cette section, nous présentons des résultats d'optimisation des paramètres extrinsèques sur des jeux de données réelles. Le premier nuage de points que l'on utilise est le nuage #4, qui provient d'une acquisition à Montbéliard. La figure 3.17 présente en haut le nuage de points avec les paramètres de calibrage initiaux donnés en entrée de l'optimisation, et sans hypothèse ou connaissance des paramètres extrinsèques comme pour les jeux de données simulées, nous présentons dans la sous-figure du bas le même nuage de points avec des paramètres extrinsèques optimisés par notre méthode.
Comme pour les jeux de données simulées, nous présentons aussi une comparaison entre deux optimisations, une sans l'utilisation des attributs de dimensionnalité et une autre avec leur utilisation. La figure 3.16 présente l'évolution des énergies pour les deux optimisations : l'énergie évolue d'une valeur de 224.81 cm^2 à une valeur de 59.26 cm^2 sans l'utilisation des attributs, et d'une valeur de 145.11 cm^2 à une valeur de 43.26 cm^2, qui est plus faible. Si l'on considère le critère de validation défini en section 3.3.3, avec une déviation standard de 5 cm pour le bruit du nuage, cela nous donne un seuil de 75 cm^2 pour la valeur finale de l'énergie, et l'optimisation respecte ce critère dans les deux cas présentés. Comme pour les jeux de données simulées, nous mettons à jour les attributs de dimensionnalité toutes les 7 itérations, et on peut voir que les effets de ces mises à jour sont similaires : cela se traduit par une diminution importante de l'énergie et une augmentation non négligeable du poids total pris en compte. Cette fois-ci, et contrairement aux jeux de données simulées, le poids total avec utilisation des attributs de dimensionnalité reste toujours plus petit que le poids total sans, ce qui n'est pas surprenant vu que l'on considère un jeu de données réelles et que de nombreux éléments non plans sont présents dans le nuage : pour ces éléments non plans, les valeurs des attributs de planéité sont faibles. Aussi, l'énergie converge assez rapidement vers la valeur optimale : le nombre d'itérations continue d'augmenter parce que les paramètres extrinsèques sont toujours modifiés. La précision des paramètres est présenté dans la table 3.5, et on peut voir que globalement, les valeurs sont correctes pour l'ensemble des paramètres, sauf pour la translation en z, mais cela est normal vu que la variation d'altitude est assez faible. Aussi, les valeurs de précisions sont plus mauvaises que pour les jeux de données simulées, ce qui justifie le fait que les énergies présentées avec la figure 3.16 ne diminuent plus après un certain nombre d'itérations : l'optimisation a convergé, mais à cause des précisions, une variation des paramètres extrinsèques ne modifie pas l'énergie, et c'est aussi pour cela que l'on a limité le nombre d'itérations à 40.
La table 3.4 présente les paramètres extrinsèques utilisés en entrée de l'optimisation, et les valeurs optimisées pour le nuage #4. Les lignes donnent :

— les paramètres initiaux, choisis arbitrairement afin de ne pas avoir besoin d'effectuer une initialisation précise.

— les paramètres extrinsèques, mesurés à la « main » sur notre véhicule d'acquisition mobile lors d'un premier étalonnage, afin d'avoir une idée du voisinage dans lequel les paramètres optimisés devraient se trouver si l'observabilité est bonne dans toutes les directions. Ces paramètres ne sont pas optimaux, et c'est pour cela que l'on a une initialisation arbitraire : le résultat d'optimisation ne doit pas nécessairement être proche de ces valeurs, puisque le but de notre

optimisation est d'affiner le nuage de point et de trouver les paramètres extrinsèques qui permettent d'avoir le meilleur affinement possible.

— les résultats d'optimisation pour les deux approches, avec et sans utilisation des attributs de dimensionnalité

Les paramètres extrinsèques optimisés sont assez proches pour les deux approches, et sont aussi assez proches des paramètres mesurés sur le véhicule : c'est le genre de résultats que l'on veut avoir avec notre optimisation, et c'est le cas ici parce que les précisions sont globalement assez bonnes pour l'ensemble des paramètres. Pour le paramètre de translation en z, la différence est très élevée, mais c'est le résultat que l'on pouvait attendre au vu de la mauvaise précision dans cette direction : l'erreur finale est de plusieurs mètres par rapport aux paramètres mesurés, alors que la précision dans cette direction est de l'ordre du mètre. Enfin, pour les temps de calculs, il est de 2 minutes environ sans les attributs de dimensionnalité et de 7 minutes environ avec, qui sont des temps de calculs acceptables au vu de la taille du nuage considéré.

	t_x(m)	t_y(m)	t_z(m)	roulis $\alpha(^\circ)$	tangage $\beta(^\circ)$	lacet $\gamma(^\circ)$
Valeurs des paramètres extrinsèques initiale	0.00	0.00	0.00	0.00	-45.00	90.00
Paramètres mesurés à la main sur notre véhicule d'acquisition	-0.21	-1.22	0.95	0.00	-60.00	90.00
Paramètres obtenus après optimisation (sans dimensionnalité)	-0.46	-1.56	-7.21	4.49	-60.06	89.59
Paramètres obtenus après optimisation (avec dimensionnalité)	-0.48	-1.37	-7.60	-0.27	-59.84	93.74

TABLE 3.4 – Erreurs sur les paramètres extrinsèques après optimisation pour le nuage #4

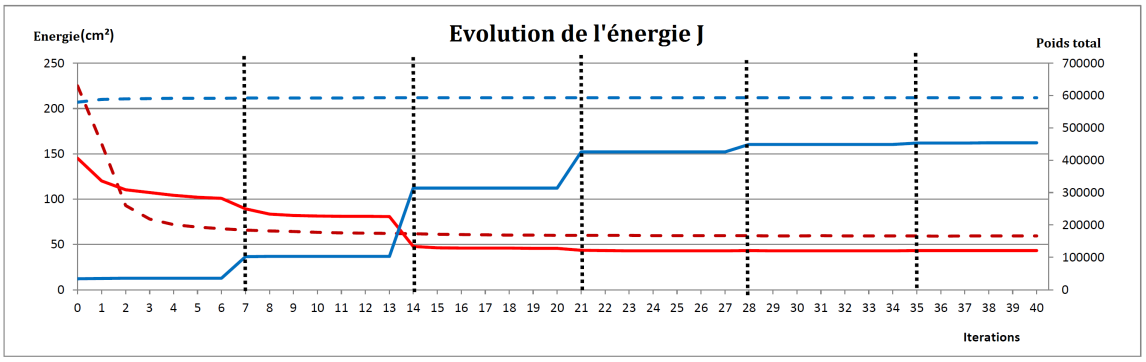


FIGURE 3.16 – Évolution de l'énergie au cours de l'optimisation pour le nuage réel #4. La ligne rouge pleine représente l'évolution de l'énergie avec l'utilisation des attributs de dimensionnalité ; la ligne rouge pointillé représente la même énergie sans l'utilisation des attributs ; la ligne bleue pleine représente l'évolution du poids total utilisé pour normaliser notre énergie, avec l'utilisation des attributs de dimensionnalité ; la ligne bleue pointillé représente la même donnée, mais sans l'utilisation des attributs de dimensionnalité ; enfin, les lignes noires verticales marquent la mise à jour des attributs de dimensionnalité, toutes les 7 itérations.

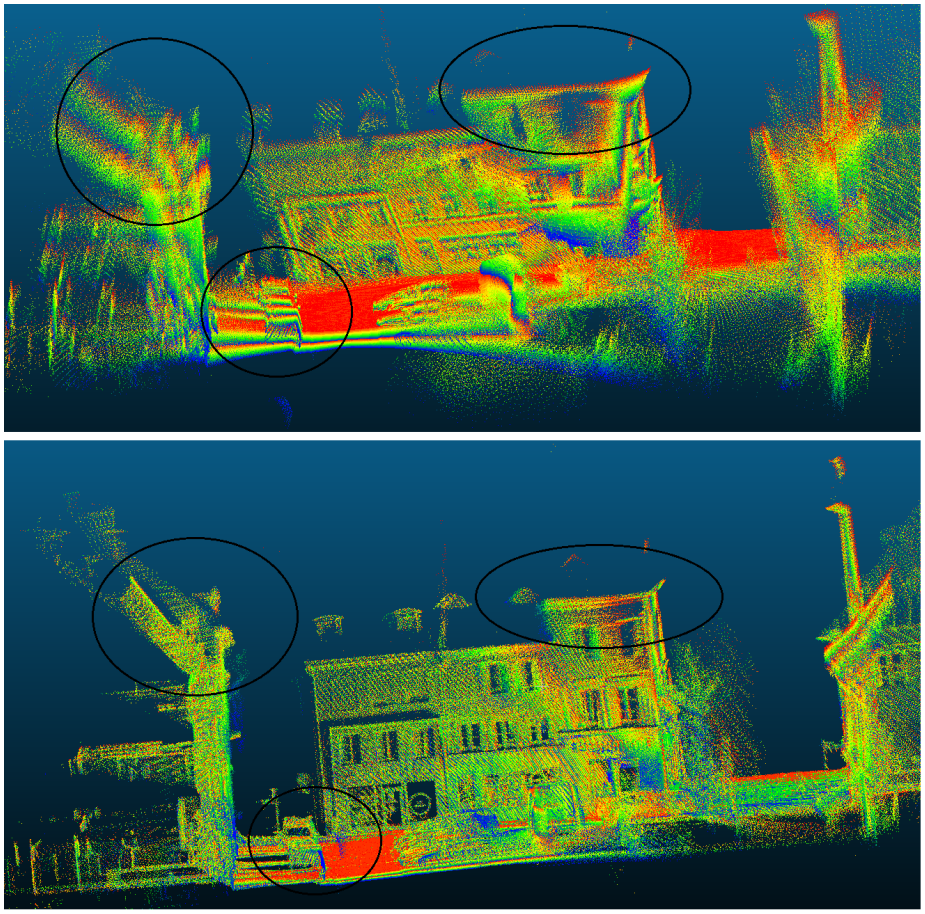
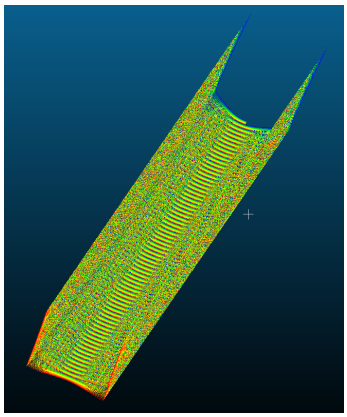


FIGURE 3.17 – Nuage de points réel #4 : en haut, nuage avec les paramètres de calibrage extrinsèque initiaux ; en dessous, le résultat de notre optimisation. Les deux images sont vues depuis le même point

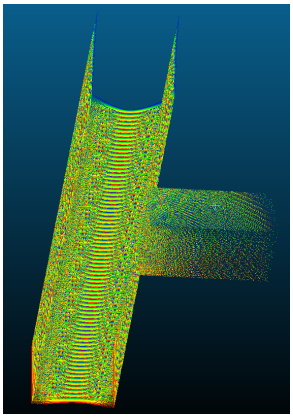
	σ_{tx}(cm)	σ_{ty}(cm)	σ_{tz}(cm)	$\sigma_{\alpha}(^{\circ})$	$\sigma_{\beta}(^{\circ})$	$\sigma_{\gamma}(^{\circ})$
Nuage réel #4	7.77	3.10	157.66	0.50	0.33	0.47

TABLE 3.5 – Précision des paramètres extrinsèques pour le nuage de points réel #4

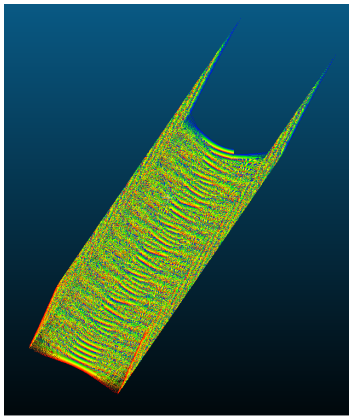
Un deuxième jeu de données réelles est utilisé, le nuage #5. Les résultats sont similaires à ceux présentés pour le nuage #4, et sont détaillés en Annexe C.



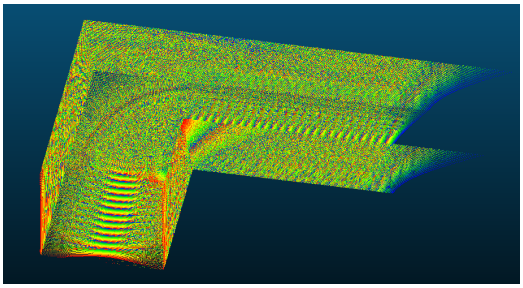
(a) Nuage simulé #a



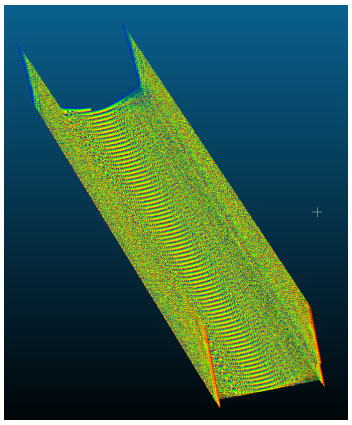
(b) Nuage simulé #b



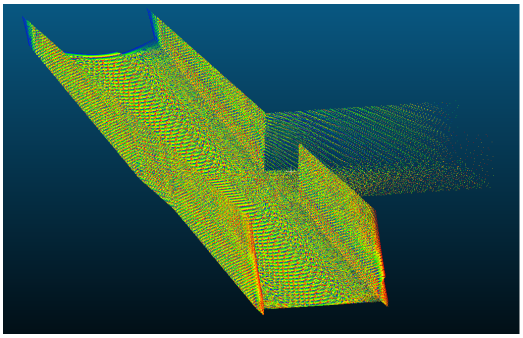
(c) Nuage simulé #c



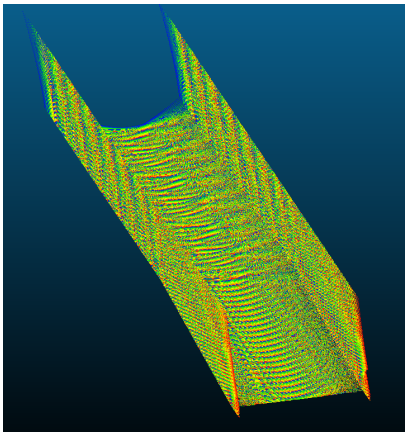
(d) Nuage simulé #d



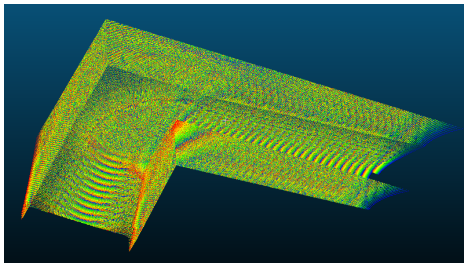
(e) Nuage simulé #e



(f) Nuage simulé #f



(g) Nuage simulé #g



(h) Nuage simulé #h

FIGURE 3.18 – Nuages simulés utilisés pour l'étude de l'observabilité lors de l'optimisation des paramètres extrinsèques

3.5 Observabilité des paramètres extrinsèques

Dans les sections 3.3 et 3.4, nous avons vu une méthode d'optimisation automatique des paramètres de calibrage extrinsèque d'un système LIDAR multi-couches, avec des résultats simulés et réels où le système concerné est un capteur LIDAR multi-fibres, le Velodyne HDL-32E. Nous avons montré des résultats satisfaisants pour nos objectifs, mais on a tout de même pu relever certains problèmes, notamment par rapport à ce que l'on a appelé « l'observabilité » des paramètres.

L'observabilité d'un paramètre permet de traduire la « capacité » à détecter les variations de ce paramètre, et plus un paramètre est observable, plus il est par exemple facilement optimisable ou plus il aura d'influence sur un système donné. Si on reprend les résultats présentés en section 3.4.4.2, avec le jeu de données réelles #4, les paramètres de précision présentés dans la table 3.5 sont inversement liés à l'observabilité : plus la précision est faible, plus un paramètre est observable, et inversement. Du coup, comme nous l'avons expliqué, les 3 paramètres de rotations extrinsèques sont les paramètres les plus observables, et pour les translations, ce sont les translations selon les axes x et y qui sont correctement observables ; la translation selon l'axe z n'est pas du tout observable. L'effet de l'observabilité est visible sur les résultats d'optimisation des paramètres extrinsèques : si l'observabilité d'un paramètre est bonne, le paramètre optimisé sera proche de la vérité terrain, ou tout du moins proche de la valeur réelle ; à l'inverse, si un paramètre n'est pas observable, la valeur optimisée par notre algorithme ne modifiera pas la structure du nuage, mais sera éloignée de la valeur réelle.
L'observabilité d'un paramètre dépend de plusieurs choses, des données mais aussi du système numérique dans lequel le paramètre intervient.

- La structure des données influe directement sur l'observabilité des paramètres. Si on reprend le nuage réel #4 utilisé pour les tests de notre algorithme, ce nuage comporte plusieurs bâtiments, ainsi que certains objets non linéaires et quelques véhicules ; il y a aussi une légère variation d'altitude durant l'acquisition. Le véhicule mobile d'acquisition effectue quelques virages. Ces différentes caractéristiques influent donc directement sur l'observabilité des paramètres : les virages permettent d'avoir une bonne observabilité dans les directions de translations x et y ; les bâtiments et leur structure non uniforme, ainsi que les déplacements du véhicule permettent d'avoir une bonne observabilité pour les trois paramètres de rotations ; par contre, pour le paramètre de translation selon l'axe z, l'observabilité est mauvaise car il y a peu de changements dans la direction verticale et l'altitude ne varie pas assez. Avec le nuage réel #5, dont les résultats d'affinement sont présentés en Annexe C, l'observabilité est différente car les caractéristiques du nuage ont changé : la trajectoire du véhicule est une ligne droite, et cela suffit à rendre l'observabilité dans plusieurs directions mauvaises.
- Le système numérique dans lequel les paramètres interviennent est aussi important : dans notre cas, nous avons une fonctionnelle à minimiser et une résolution qui nous permet d'obtenir les valeurs de précisions comme présenté dans la section 3.3.4 ; un autre choix de résolution où des paramètres à optimiser différents n'auraient pas conduits à la même précision puisque la matrice de covariance que l'on utilise aurait été différente.

Avec notre résolution numérique, nous avons testé plusieurs configurations d'acquisitions pour voir comment évolue l'observabilité des paramètres extrinsèques en fonction des caractéristiques des nuages de points et de la trajectoire du véhicule. Pour cela, nous avons simulé 8 acquisitions, avec différentes trajectoires du véhicule et caractéristiques ; ces nuages sont présentés dans la figure 3.18 :

- Le jeu de données #a est composé d'un sol et de 2 plans verticaux. Il n'y a pas de variation d'altitude, et le véhicule a une trajectoire rectiligne entre les 2 murs.
- Le jeu de données #b est composé d'un sol et de 5 plans verticaux. Le véhicule mobile a aussi une trajectoire rectiligne, et il n'y a pas de variation d'altitude lors de l'acquisition non plus.
- Le jeu de données #c est le même que le jeu de données #a, à ceci près que le véhicule n'a pas une trajectoire rectiligne cette fois-ci.

- Le jeu de données #d est composé d'un sol et de 4 plans verticaux représentants des façades. Il n'y a pas de variations d'altitude, et le véhicule mobile effectue un virage pendant l'acquisition.
- Les jeux de données #e à #h sont les mêmes que les jeux de données #a à #d, mais avec une variation de l'altitude cette fois-ci.

Les paramètres extrinsèques utilisés pour initialiser l'optimisation sont les mêmes pour l'ensemble des jeux de données simulées : des biais présentés sur la première ligne du tableau 3.6 ont été ajoutés à la vérité terrain, qui est la même pour tous les nuages. Sur le même tableau, les erreurs après optimisation par rapport aux paramètres extrinsèques de la vérité terrain sont données pour chaque nuage. Si l'on s'intéresse au nuage #a, les paramètres de translation ne devraient pas être observables, et l'observabilité pour les paramètres de rotations devrait être correcte : en effet, la structure de l'environnement est du type couloir et le véhicule a une trajectoire rectiligne selon l'axe des x entre les 2 « murs » parallèles ; de plus, il n'y a pas de variation d'altitude, ce qui enlève toute observabilité dans les directions des trois translations. Pour les rotations, comme le capteur Velodyne tourne et que celui-ci est penché sur le toit du véhicule mobile, comme le présente la figure 2.4, on a une observabilité correcte pour chacune des trois rotations. C'est ce que l'on peut voir avec la table 3.6 qui présente les erreurs des paramètres extrinsèques après optimisation par rapport à la vérité terrain, et la table 3.7 qui présente la précision des paramètres extrinsèques avec notre optimisation : l'erreur après optimisation est la même qu'avant optimisation pour les paramètres de translations, et elle est inférieure au degré pour les rotations sauf pour le tangage, qui est la rotation autour de l'axe des y, transverse au déplacement du véhicule. Pour les précisions liées aux paramètres extrinsèques, elles sont présentées dans la table 3.7 et suivent les observations faites plus tôt : pour les translations, les précisions sont mauvaises, alors qu'elles sont correctes pour les rotations. Le jeu de données #b est très proche du jeu de données #a, ce qu'il y a de différent étant une variation de structure pour une des deux façades. La variation de structure devrait ajouter de l'observabilité pour certaines translations : les tableaux d'erreurs et de précisions donnent des résultats similaires et proches de ceux obtenus pour le jeu de données #a. Cela s'explique du fait qu'à l'échelle de l'acquisition, la variation de structure de la façade est assez négligeable.
Le jeu de données #c présente le même environnement que celui présenté avec le jeu de données #a, mais cette fois-ci, le véhicule mobile a une trajectoire non rectiligne. Cela apporte de l'observabilité selon les directions horizontales de translations x et y : la table 3.6 confirme ce résultat, avec des erreurs par rapport à la vérité terrain inférieure au centimètre pour les translations selon les axes x et y ; la translation selon l'axe z n'est toujours pas observable puisque il n'y a pas de variation d'altitude. Pour les rotations, les résultats sont sensiblement meilleurs pour le roulis et le tangage grâce à la variation d'orientation du véhicule pendant l'acquisition. Les précisions pour chacun des paramètres confirment ces observations. Finalement, le jeu de données #d présente une acquisition où le véhicule a une trajectoire non rectiligne, et où l'environnement à une variation de structure bien marquée ; nous ne sommes plus dans une configuration de type couloir, et les résultats d'optimisation sont meilleurs. Les erreurs des paramètres optimisés par rapport à la vérité terrain sont globalement meilleurs que pour le jeu de données #c, toujours sauf pour la translation selon l'axe des z car il n'y a pas de variation d'altitude. Les précision pour le jeu de données #d vont aussi dans le sens de cette observation, avec en général des meilleures précisions que pour le jeu de données #c.

Le jeu de données #e est le même que le jeu de données #a, avec une variation d'altitude en plus. Le résultat attendu est que l'on ajoute de l'observabilité dans la direction de la translation en z avec la variation d'altitude : or, d'après les tableaux d'erreurs et de précisions, ce n'est pas le cas et les résultats sont similaires à ceux obtenus pour le jeu de données #a. Au contraire, pour le jeu de données #f qui est le même que le jeu de données #b avec une variation d'altitude en plus, les 2 tableaux montrent que l'on a ajouté de l'observabilité dans les directions x et z ; il semblerait qu'une structure différente d'un couloir pour le nuage couplée à une variation d'altitude permette d'apporter de l'observabilité dans la direction de la marche du véhicule, ainsi que dans la direction verticale. Par contre, la translation selon l'axe des y n'est toujours pas observable car la trajectoire du véhicule est rectiligne selon l'axe des x : il n'y a pas de variations selon l'axe des y, et

l'environnement est assez proche d'un couloir malgré la variation de structure d'une des 2 façades parallèles. Pour le jeu de données #g, nous avons les mêmes caractéristiques que pour le jeu de données #c, avec une variation d'altitude en plus. L'environnement est toujours de type couloir, mais comme nous avons des variations de trajectoire dans chacune des directions de translations, on s'attend à avoir une observabilité correcte pour les paramètres de translations : c'est le cas, comme présenté avec les tableaux d'erreurs et de précisions. Pour les erreurs, elles sont assez faibles pour chacun des paramètres optimisés, et pour les précisions, nous avons une très bonne valeur pour la translation selon l'axe des x et pour les rotations ; pour les translations selon les axes y et z, la précision est moins bonne, mais les résultats d'optimisations sont corrects avec des erreurs de l'ordre de quelques millimètres après optimisation par rapport à la vérité terrain : cela est normal, parce que l'on travaille avec des données simulées « parfaites », et où seuls des plans sont présents. Sur des données réelles, avec ces mêmes valeurs de précisions, les erreurs pour les paramètres de translations par rapport à une vérité terrain seraient plus élevées. Enfin, le jeu de données #h est le même que le jeu de données #d, avec une variation d'altitude en plus. Comme pour les données sans variation d'altitude, les erreurs sont globalement les plus faibles avec cette optimisation, pour tous les paramètres cette fois-ci. En effet, comme pour le jeu de données #g, et puisque nous avons une variation de la direction d'acquisition dans toutes les directions de translation, aussi parce que la structure de l'environnement n'est pas uniforme comme pour un couloir, les erreurs après optimisation pour le jeu de données #h sont très faibles pour tous les paramètres extrinsèques. Aussi, on peut voir que les précisions sont très bonnes pour l'ensemble des paramètres, ce qui va dans le sens des résultats obtenus sur l'erreur entre les paramètres extrinsèques optimisés et la vérité terrain.

Ce que l'on peut voir en comparant les tables 3.6 3.7 est qu'il y a une forte corrélation entre les résultats d'optimisation pour un nuage et la précision des paramètres extrinsèques que l'on mesure pour ce même nuage. En effet, pour l'ensemble des nuages simulés qui ont servis à effectuer cette étude, lorsque l'erreur d'un paramètre extrinsèque optimisé avec la vérité terrain est très élevée, on remarque que la précision du paramètre associée est très mauvaise. D'un autre côté, lorsque l'erreur entre un paramètre optimisé et la vérité terrain est faible, la précision associée est très bonne. On peut ainsi s'appuyer sur les valeurs de précisions pour valider ou non les valeurs de paramètres optimisés, puisque une valeur de précision faible signifie que l'on est capable avec notre optimisation de retrouver des paramètres correctements optimisés.

En résumé, pour que les paramètres de translations extrinsèques que l'on optimise soient correctement observables, il est nécessaire que :

- le véhicule ne doit pas avoir une trajectoire rectiligne pour qu'il y ait des changements dans les directions de translations horizontales, dans l'idéal avec un ou plusieurs virages, ce qui permet d'apporter de l'observabilité dans ces directions de translations.
- l'altitude varie au cours de l'acquisition ; plus l'altitude varie fortement, plus le paramètre de translations en z sera observable.

Pour les paramètres de rotations, la configuration du système d'acquisition donne déjà une bonne observabilité pour chacun des paramètres. Une combinaison de ces caractéristiques permet d'avoir des paramètres extrinsèques correctement optimisés et proches de la vérité terrain. Ces observations sont confirmées avec les résultats d'optimisation sur des jeux de données réelles présentés en section 3.4.4.2 : pour le nuage #4, qui comporte une variation d'altitude et pour lequel le véhicule mobile a une trajectoire non rectiligne, les précisions sont correctes pour les rotations, et pour les translations, les précisions sont aussi correctes sauf pour la translation en z car la variation d'altitude. Pour le nuage #5, les précisions pour les rotations sont correctes, et pour les translations, les précisions sont plus mauvaises car l'environnement est de type couloir et la variation d'altitude est assez faible.

	t_x(cm)	t_y(cm)	t_z(cm)	roulis $\alpha(^\circ)$	tangage $\beta(^\circ)$	lacet $\gamma(^\circ)$
Erreurs initiales ajoutées à la vérité terrain	-150.00	250.00	-200.00	5.00	-7.00	-5.50
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #a	**-150.00**	**250.00**	**-200.00**	-0.0465	-4.41	-0.0001
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #b	**-150.00**	**250.00**	**-200.00**	-0.031	-3.60	-0.0013
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #c	-0.0374	0.19	**-200.00**	-0.0077	-0.011	0.0004
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #d	-0.0015	0.025	**-200.00**	-0.0038	0.0027	-0.0016
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #e	**-150.00**	**250.00**	**-200.00**	-0.0068	-4.49	-0.4515
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #f	-2.0063	**250.00**	4.2441	-0.0427	0.188	0.0359
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #g	-0.06	0.0058	0.46	-0.0186	0.015	-0.0014
Différence entre la vérité terrain et le résultat d'optimisation pour le nuage #h	-0.0362	-0.02	0.43	-0.0034	0.0012	0.

TABLE 3.6 – Erreurs sur les paramètres extrinsèques ajoutés à l'ensemble des nuages simulés pour l'étude de l'observabilité

	σ_{t_x}(cm)	σ_{t_y}(cm)	σ_{t_z}(cm)	$\sigma_\alpha(^\circ)$	$\sigma_\beta(^\circ)$	$\sigma_\gamma(^\circ)$
Nuage de points simulé #a	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	1.56	2.47	0.72
Nuage de points simulé #b	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	1.48	2.51	0.75
Nuage de points simulé #c	1.99	45.46	$\mathbf{1.00 * 10^{20}}$	0.41	1.08	0.13
Nuage de points simulé #d	2.90	1.50	$\mathbf{1.00 * 10^{20}}$	0.25	0.13	0.15
Nuage de points simulé #e	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	$\mathbf{1.00 * 10^{20}}$	1.47	2.12	0.67
Nuage de points simulé #f	118.27	$\mathbf{1.00 * 10^{20}}$	91.05	1.34	1.89	0.72
Nuage de points simulé #g	2.27	53.44	345.49	0.62	0.99	0.16
Nuage de points simulé #h	2.88	1.59	16.02	0.22	0.12	0.13

TABLE 3.7 – Précision des paramètres extrinsèques pour les nuages simulés utilisés pour l'étude de l'observabilité

3.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode d'affinement de nuages de points par optimisation des paramètres de calibrage extrinsèque d'un système mobile d'acquisition. La méthode que l'on propose est automatique et n'utilise que des informations extraites des données : nous utilisons la structure des LIDARs multi-couches et la redondance de données entre les couches du capteur pour réaliser notre optimisation. Des résultats d'optimisation sur des jeux de données simulées et réelles ont été présentés, et les affinements ont tous été validés. Nous avons aussi montré que l'utilisation des attributs de dimensionnalité apporte un plus à l'optimisation proposée, notamment du fait que nous effectuons un recalage entre couches du capteur basée sur une distance point-à-plan. L'optimisation que l'on propose est rapide : en effet, les temps de calculs que l'on présente englobent l'ensemble des opérations qui sont effectuées sur les données (lecture, écriture, échantillonnage, optimisation...), et sont de l'ordre de quelques minutes pour plusieurs millions de points. Aussi, notre optimisation est robuste à une mauvaise initialisation des paramètres que l'on corrige : nos tests ont montré que nous n'avons pas besoin d'avoir une initialisation précise des paramètres pour que notre optimisation converge et permette de correctement affiner le nuage de points. Enfin, nous avons aussi traité d'un problème récurrent dans l'optimisation de données qui est l'observabilité des données : dans le cas des paramètres de calibrage extrinsèque, nous avons pu définir les caractéristiques « idéales » (structure du nuage de points et trajectoire du véhicule) qui permettent d'avoir des paramètres extrinsèques correctement optimisés.