Affinement de nuages de points par optimisation des paramètres intrinsèques

Sommaire

Introduction

Dans le chapitre 3, nous avons présenté un algorithme d'optimisation des paramètres de calibrage extrinsèque dans le but d'affiner des nuages de points issus de cartographies mobiles. Si on se réfère à la figure 2.6, il s'agit d'une des trois étapes principales au géoréférencement des données lors d'une cartographie avec un véhicule mobile. Dans ce cas, nous avions supposé que les paramètres intrinsèques du système LIDAR étaient corrects, et que la trajectoire du véhicule à tout instant était connue. Cette hypothèse n'est pas forcément vérifiée, notamment concernant le calibrage intrinsèque d'un système LIDAR. En effet, là où le calibrage extrinsèque dépend du système mobile utilisé et de la configuration choisie par « l'utilisateur », le calibrage intrinsèque est plus spécifique, et dépend de la configuration du capteur utilisé pour l'acquisition : il s'agit de correctement récupérer les données brutes, généralement en coordonnées sphériques, et de référencer ces données dans un repère laser

cartésien.
Dans ce chapitre, nous allons présenter un algorithme permettant aussi d'affiner des nuages de points, mais en jouant sur les paramètres de calibrage intrinsèque cette fois-ci ; l'idée est d'aller dans la continuité de ce qui a été présenté au chapitre 3, voire de proposer une amélioration en optimisant un plus grand nombre de paramètres de calibrage pour un système d'acquisition. La solution que l'on présente dans ce chapitre est applicable dans sa méthodologie à l'optimisation des paramètres de calibrage d'un système mobile équipé soit d'un LIDAR multi-couches, soit de plusieurs LIDARs. Nous allons présenter un modèle d'optimisation pour un LIDAR du type Velodyne 32 fibres, mais en adaptant les équations de calibrage intrinsèque au type de système utilisé, l'optimisation est applicable aux mêmes types de systèmes que ceux présentés dans l'introduction du chapitre 3.

4.1 Importance des paramètres de calibrage intrinsèque

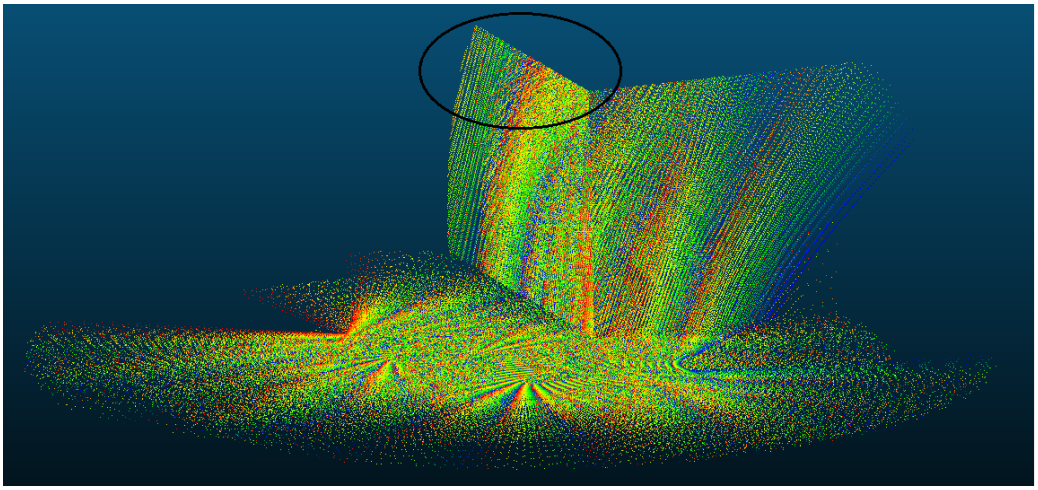
Le calibrage intrinsèque d'un système LIDAR dépend du modèle que le constructeur a choisi pour construire son ou ses capteurs LIDAR. En effet, la configuration « intrinsèque » d'un capteur LIDAR est fixe, et un modèle de calibrage est alors choisi pour représenter au mieux la transformation qui permet d'obtenir des données référencées dans le repère cartésien du capteur à partir des données brutes que le capteur acquiert. Le modèle peut prendre en compte des erreurs potentielles, ou être le plus simple possible : le modèle de calibrage est choisi par l'utilisateur en fonction de la configuration du scanner. Tout comme pour l'optimisation des paramètres de calibrage extrinsèque, l'optimisation des paramètres de calibrage intrinsèque a pour but d'affiner le nuage de point.

4.1.1 Définition du calibrage intrinsèque pour un capteur LIDAR

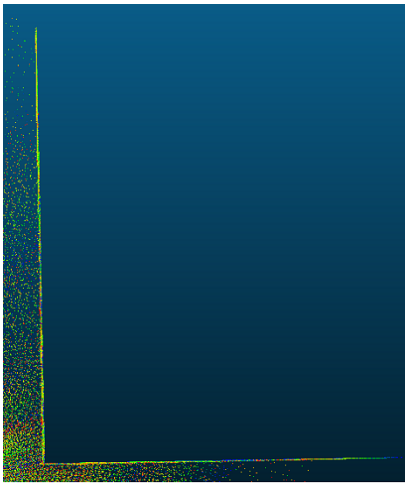
Ce que l'on appelle **calibrage intrinsèque** pour un capteur LIDAR définit la transformation qui permet de référencer des données brutes qui sont en coordonnées sphériques en coordonnées cartésiennes dans le repère lié au capteur. Dans le cas d'un LIDAR mono-fibre, on trouve trois équations qui permettent d'effectuer le calibrage intrinsèque qui sont celles données par l'équation (2.1). Dans le cas d'un capteur multi-fibres, le modèle peut-être le même, que l'on applique à chaque laser du capteur. Ce modèle peut être utilisé par les systèmes LIDARs composés de plusieurs lasers centrés sur un axe vertical et tournants autour de ce même axe, comme le Velodyne 32 fibres par exemple ; pour certains capteurs comme le Velodyne modèle 64 fibres, les lasers sont excentrés et le modèle de calibrage donné par le constructeur est différent de celui-ci, plus complexe car il prend notamment en compte les offsets verticaux et horizontaux de chaque laser, ainsi que les offsets au niveau des rotations. Dans la suite de ce chapitre, pour la construction du modèle corrigé de calibrage intrinsèque que l'on va chercher à optimiser, nous allons détailler les calculs dans le cas du modèle présenté avec l'équation (2.1).

4.1.2 Effets d'un mauvais étalonnage des paramètres intrinsèques sur les données

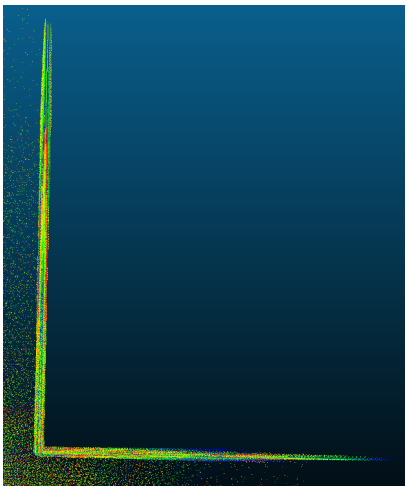
Tout comme pour les paramètres de calibrage extrinsèques, des paramètres de calibrage intrinsèques erronés vont aussi fausser la projection des données dans le repère capteur, ce qui a pour effet d'engendrer un mauvais référencement et une déformation des données dans le repère monde. La figure 4.1 présente le type de déformation que l'on peut obtenir lorsque les paramètres intrinsèques sont erronés pour des fibres d'un capteur multi-fibres : l'erreur est de l'ordre du degré pour les angles et de quelques centimètres pour les distances. La sous-figure a) présente le nuage avec des paramètres de calibrage corrects, 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 calibrage, où les plans verticaux sont déformés.



(a)



(b)



(c)

FIGURE 4.1 – Nuages de point avec plusieurs calibrages intrinsèques : a) Paramètres correctement étalonnés ; b) Paramètres corrects, avec zoom sur la zone d'intérêt ; c) Paramètres erronés

Les erreurs sur les paramètres de calibrage intrinsèque proviennent généralement d'un modèle simplifié induit par la structure du système LIDAR. Dans le cas de capteurs multi-fibres comme le Velodyne 32 ou le Quanergy, on a un faisceau de fibres réparties en éventail sur un axe vertical tournant, et un modèle de calibrage simplifié est celui que l'on donne avec l'équation (2.1). Une correction du modèle permet de corriger une bonne partie des erreurs induites par ce modèle : c'est ce que nous allons présenter dans la suite de ce chapitre, tout d'abord en présentant un état de l'art de différentes méthodes d'optimisation de calibrage intrinsèques, puis le modèle de calibrage corrigé avec lequel on cherche à affiner les données issues d'une acquisition mobile, ainsi que différents résultats expérimentaux permettant de valider le modèle corrigé et l'optimisation effectuée, et nous terminerons par présenter certaines limites que nous avons rencontrés au cours des expérimentations effectuées.

4.2 Etat de l'art sur l'optimisation des paramètres de calibrage intrinsèque

Pour le calibrage intrinsèque de systèmes LIDAR, plusieurs algorithmes d'optimisation des paramètres existent dans la littérature. Ils traitent en majorité de l'optimisation des paramètres intrinsèques d'un capteur LIDAR multi-fibres tournant, et le Velodyne 32 ou 64 fibres est à chaque fois utilisé pour les expérimentations et la validation de leur méthode. Les capteurs Velodynes sont apparus en 2007, et ont connu une popularité croissante, notamment grâce à leur introduction dans le Defense Advanced Research Projects Agency (DARPA) Urban Challenge en 2007, où de nombreux véhicules autonomes qui ont réussi à finir la course étaient équipés d'un capteur Velodyne pour analyser l'environnement. Depuis, de nombreuses méthodes d'optimisation des paramètres intrinsèques ont vu le jour ; en 2010, [Glennie et Lichti, 2010] et [Muhammad et Lacroix, 2010] proposent des optimisations des paramètres intrinsèques pour le modèle 64 fibres du Velodyne, en utilisant un environnement de calibrage particulier : de nombreux plans sont présents dans l'environnement, et ces plans sont extraits des nuages de points acquis pour que leurs structures soient contraintes, ce qui a pour effet de corriger les paramètres intrinsèques et de les optimiser. [Levinson et Thrun, 2010] présente aussi une méthode d'optimisation des paramètres intrinsèques, qui est similaire à leur méthode d'optimisation pour les paramètres extrinsèques : l'optimisation est effectuée en post-traitement, sans mire de calibrage, et n'utilise que les données acquises. Ils optimisent les paramètres intrinsèques de leur LIDAR en ajoutant un paramètre qui permet de diminuer la valeur de leur fonctionnelle.
[Atanacio-jiménez *et al.*, 2011] présente une optimisation du calibrage intrinsèque du Velodyne 64 fibres, où différents offsets sont ajoutés au modèle fourni par le constructeur pour le corriger et améliorer la qualité des nuages qui proviendront d'acquisitions futures. L'optimisation est effectuée à l'aide d'un environnement composé de plusieurs plans, dont la structure est optimisée en ajoutant différents offsets aux paramètres intrinsèques. [Mirzaei *et al.*, 2012] présente une méthode similaire pour optimiser les paramètres intrinsèques d'un LIDAR multi-fibres ; un plan d'étalonnage est utilisé, et les impacts du LIDAR sur ce plan selon plusieurs configurations sont mesurés : une mise en correspondance des différentes acquisitions permet de réévaluer les paramètres intrinsèques du LIDAR. Toujours la même année, dans [Chen *et al.*, 2012], une autre méthode d'optimisation des paramètres intrinsèques est proposée, là aussi en cherchant à extraire des plans des données acquises et à les mettre en correspondance d'un point de vue à l'autre pour optimiser le calibrage intrinsèque. La nouveauté est que les auteurs cherchent aussi à avoir une cohérence temporelle pour l'optimisation, et appliquent donc un recalage spatio-temporel sur les plans, ce qui donne de la consistance temporelle aux paramètres intrinsèques optimisés.
En 2013, plusieurs méthodes d'optimisation des paramètres intrinsèques d'un capteur Velodyne ont vu le jour : [Chan et Lichti, 2013] et [Chan *et al.*, 2013] présentent deux méthodes d'optimisation des paramètres intrinsèques du Velodyne 32 fibres, le premier article en extrayant des plans et des cylindres des données, puis en optimisant les paramètres de calibrage pour que ces éléments soient correctement recalés entre eux, et le deuxième article en n'utilisant que des éléments cylindriques des données. Dans [Huang *et al.*, 2013], une optimisation partielle des paramètres intrinsèques d'un Velodyne 64 fibres est proposée, avec l'utilisation d'une caméra infrarouge pour visualiser les impacts lasers sur un plan.
La majorité de ces méthodes d'optimisation sont appliquées en « mode **statique** », et dans une extension des travaux présentés dans [Chan et Lichti, 2013] est proposée dans [Chan et Lichti, 2015], avec l'utilisation de données acquises en mode **cinématique**, en recalant des plans et cylindres extraits des données.

4.3 Méthode d'optimisation proposée

Nous avons déjà traité de l'optimisation des paramètres extrinsèques en supposant que les paramètres intrinsèques du système LIDAR étaient correctement étalonnés, et aussi que la fusion de données GPS + centrale inertielle nous permettait d'avoir un positionnement précis du véhicule à tout instant. Toutefois, cette hypothèse n'est pas tout à fait vérifiée : il est vrai que l'étalonnage intrinsèque donne des valeurs assez précises, mais le modèle donné par le constructeur peut être amélioré, et c'est ce que nous avons voulu proposer. Dans cette section, nous allons traiter de l'optimisation des paramètres intrinsèques liés au Velodyne HDL-32E, mais l'optimisation se veut « généralisable » : en effet, comme nous allons l'expliquer, le modèle d'optimisation corrigé est spécifique à un capteur ou un système LIDAR ; cependant, en adaptant le modèle corrigé au type de capteur, l'optimisation reste la même.
Pour le Velodyne HDL-32E, nous avons choisi un modèle à corriger qui prenait en compte selon nous la plupart des biais qui pouvaient être introduits par le modèle du constructeur. Le modèle corrigé que l'on a choisi d'utiliser va être présenté dans la section suivante 4.3.1.

4.3.1 Algorithme proposé et optimisation

L'idée est d'aller dans la continuité de l'optimisation des paramètres extrinsèques. Pour cela, nous avons repris l'énergie J(R, T) de l'équation (3.6) avec laquelle nous avons optimisé les paramètres extrinsèques. Dans un premier temps, nous supposons que les paramètres extrinsèques sont connus et précis, et que seuls les paramètres de calibrage intrinsèques ont besoin d'être optimisés.

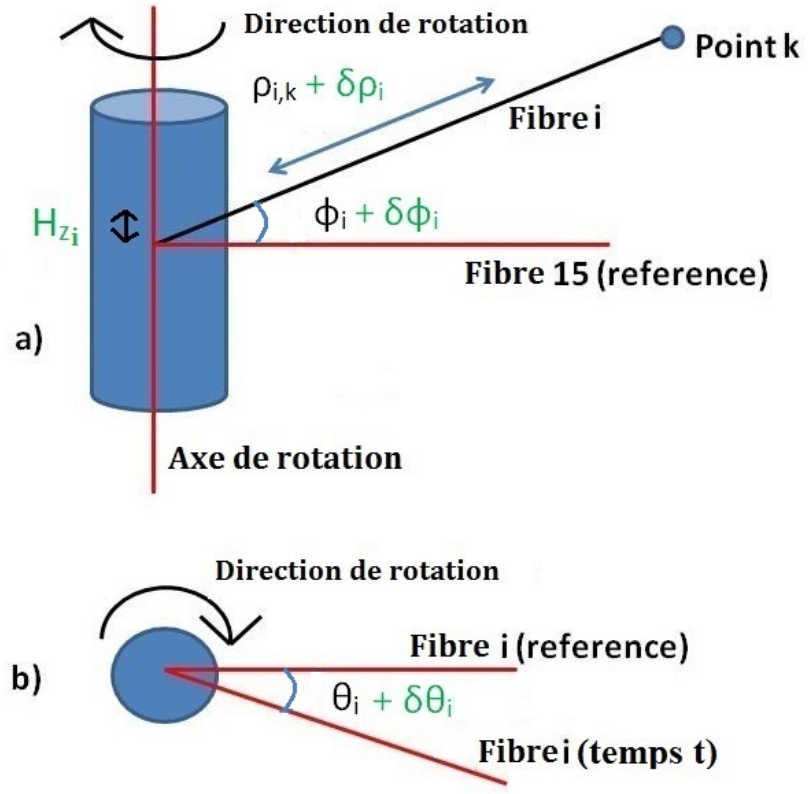


FIGURE 4.2 – Représentation des paramètres intrinsèques pour le capteur Velodyne HDL-32E

La figure 4.2 présente schématiquement la façon dont le capteur Velodyne acquiert les données : le capteur est équipé de 32 fibres placées en éventail sur un plan vertical, mais nous n'en avons représenté que 2 sur la figure a). Aussi, nous y reviendrons plus tard, mais nous considérons la fibre numérotée 15 par le constructeur comme étant une fibre de « référence ». Pour chaque acquisition, le capteur fournit les informations suivantes :

- L'angle vertical ϕ_i de chaque fibre i, par rapport à l'horizontale du capteur.
- La distance $\rho_{i,k}$ entre l'origine de la fibre i et le point acquis k.
- L'angle horizontal θ_i, dû à la rotation du capteur.

Le modèle de calibrage intrinsèque du Velodyne 32 fibres a déjà été présenté dans le chapitre 2 : il s'agit d'un modèle sphérique, puisque l'ensemble des fibres du capteur sont positionnées sur un axe tournant. Les trois équations sont les suivantes :

$$p'_i(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} \rho_i(k) * cos(\theta_i(t)) * cos(\phi_i) \\ -\rho_i(k) * sin(\theta_i(t)) * cos(\phi_i) \\ \rho_i(k) * sin(\phi_i) \end{pmatrix} \tag{4.1}$$

Ce modèle est proche de celui présenté par les auteurs dans [Chan et Lichti, 2013] : les auteurs ne corrigent que la distance mesurée et l'angle vertical de chaque fibre, alors que l'on a ajouté une correction sur l'angle horizontal induit par la rotation du scanner et une correction de positionnement vertical de la fibre au sein du scanner. Il s'agit du modèle que l'on veut corriger, pour plusieurs raisons :

- Le modèle suppose qu'entre chaque fibre du capteur, il y a le même écart angulaire vertical, ce qui n'est pas certain à cause de quelques défauts de construction par exemple. Nous ajoutons un offset $\delta\phi_i$ pour chaque angle vertical ϕ_i afin de corriger des écarts angulaires non uniformes.
- Toutes les fibres sont placées sur le même axe vertical, mais le modèle suppose que les fibres sont toutes orientés dans la même direction. Pour les mêmes raisons que précédemment, nous avons ajouté un offset $\delta\theta_i$ pour chaque angle horizontal θ_i.
- Aussi,pour palier à quelques petites erreurs de mesures de distances, notamment lorsqu'elles sont élevées, nous avons un ajouté offset $\delta\rho_i$ par fibre i (et identique pour tous les points acquis par cette fibre) sur les distances $\rho_i(k)$ mesurés entre le centre de la fibre i et le point k.
- Enfin, les fibres sont supposés concentriques, et nous avons ajouté un offset vertical H_z de position pour chaque fibre, afin de prendre en compte des erreurs dues à une origine différente des fibres.

Avec ces différents offsets, les équations (4.1) de calibrage intrinsèque deviennent :

$$p'_i(t) = \begin{pmatrix} (\rho_i(k) + \delta\rho_i) * cos(\theta_i(t) + \delta\theta_i) * cos(\phi_i + \delta\phi_i) \\ -(\rho_i(k) + \delta\rho_i) * sin(\theta_i(t) + \delta\theta_i) * cos(\phi_i + \delta\phi_i) \\ (\rho_i(k) + \delta\rho_i) * sin(\phi_i + \delta\phi_i) + H_{z,i} \end{pmatrix} \tag{4.2}$$

4.3.1.1 Approximation linéaire

Les équations de calibrage intrinsèque ne sont pas linéaires, et pour simplifier notre problème d'optimisation, nous effectuons une linéarisation au 1^{er} ordre, ce qui donne des équations de calibrage linéarisées pour le point k acquis par la fibre i :

$$\begin{aligned} p'_{i,k}(t) = p'_{1,i,k}(t) + p'_{2,i,k}(t) * \delta\rho_i + p'_{3,i,k}(t) * \delta\theta_i \\ + p'_{4,i,k}(t) * \delta\phi_i + p'_{5,i} \end{aligned} \tag{4.3}$$

avec :

$$\begin{cases} p'_{1,i,k}(t) = \begin{pmatrix} \rho_i(k) * cos(\theta_i(t)) * cos(\phi_i) \\ -\rho_i(k) * sin(\theta_i(t)) * cos(\phi_i) \\ \rho_i(k) * sin(\phi_i) \end{pmatrix} \\ p'_{2,i,k}(t) = \begin{pmatrix} cos(\theta_i(t)) * cos(\phi_i) \\ -sin(\theta_i(t)) * cos(\phi_i) \\ sin(\phi_i) \end{pmatrix} \\ p'_{3,i,k}(t) = \begin{pmatrix} \rho_i(k) * cos(\theta_i(t) + 90) * cos(\phi_i) \\ -\rho_i(k) * sin(\theta_i(t) + 90) * cos(\phi_i) \\ \rho_i(k) * sin(\phi_i) \end{pmatrix} \\ p'_{4,i,k}(t) = \begin{pmatrix} \rho_i(k) * cos(\theta_i(t)) * cos(\phi_i + 90) \\ -\rho_i(k) * sin(\theta_i(t)) * cos(\phi_i + 90) \\ \rho_i(k) * sin(\phi_i + 90) \end{pmatrix} \\ p'_{5,i} = \begin{pmatrix} 0 \\ 0 \\ H_{z,i} \end{pmatrix} \end{cases}$$

$p'_{i,k}(t)$ défini dans l'équation (4.3) est ensuite réinjecté dans la fonctionnelle J définie dans (3.6), et il ne reste plus qu'à minimiser la fonctionnelle pour trouver les différents offsets optimaux.

4.3.1.2 Approche d'optimisation

Pour l'optimisation des offsets que nous avons ajoutés au modèle de calibrage, nous avons choisi de prendre la fibre numérotée 15 comme référence pour plusieurs raisons :

- Tout d'abord, le problème d'optimisation que l'on a est sous-contraint, et une optimisation de l'ensemble des offsets ajoutés aux fibres du Velodyne ne convergerait pas vers une solution optimale puisque ces mêmes offsets peuvent se compenser d'une fibre à l'autre.
- D'autre part, lorsque l'on regarde la datasheet du Velodyne 32 fibres, la structure du Velodyne y est détaillée : les fibres sont numérotées de 0 à 31 en fonction de l'ordre de tir lors d'une acquisition, et l'angle vertical de la fibre 15 est de 0°. Par conséquent, cette fibre est la plus à même d'être la « plus correctement » placée dans le capteur, et le problème d'optimisation devient alors un recalage des nuages issus des autres fibres les unes par rapport aux autres, tout en supposant que le nuage issu de la fibre 15 est optimal.

Au total, il nous reste $4 * 31 = 124$ paramètres à optimiser. Dans l'équation (3.6), ces offsets apparaissent aussi bien avec les termes $p'_{i,k}$ que $m'_{j,k}$. Comme pour l'optimisation des paramètres extrinsèques, nous pouvons réécrire l'énergie J de la façon suivante :

$$J(\delta X_{int}) = \frac{\sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_k w_{i,j,k} * (B_{i,j,k} + A^T_{i,j,k} \times \delta X_{int} + o(\delta X_{int}))^2}{\sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_k w_{i,j,k}} \tag{4.4}$$

avec :

$$\begin{cases} \delta X_{int} = ([\delta\rho_i \quad \delta\theta_i \quad \delta\phi_i \quad H_{z,i}]_{i\in[\![0,31]\!]\setminus 15})^T \\ \\ B_{i,j,k} = n_{i,k}^T \times \begin{pmatrix} T_{nav}(p'_{i,k}) - T_{nav}(m'_{j,k}) \\ + \left(R_{nav}(p'_{i,k}) \times (R(\alpha,\beta,\gamma) \times p'_{1,i,k} + T(t_x,t_y,t_z))\right) \\ - \left(R_{nav}(m'_{j,k}) \times (R(\alpha,\beta,\gamma) \times m'_{1,j,k} + T(t_x,t_y,t_z))\right) \end{pmatrix} \\ \\ A_{i,j,k} = \begin{pmatrix} \dots \\ n_{i,k}^T \times \left(R_{nav}(p'_{i,k}) \times R(\alpha,\beta,\gamma) \times p'_{2,i,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(p'_{i,k}) \times R(\alpha,\beta,\gamma) \times p'_{3,i,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(p'_{i,k}) \times R(\alpha,\beta,\gamma) \times p'_{4,i,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(p'_{i,k}) \times R(\alpha,\beta,\gamma) \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T\right) \\ \dots \\ n_{i,k}^T \times \left(R_{nav}(m'_{j,k}) \times R(\alpha,\beta,\gamma) \times m'_{2,j,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(m'_{j,k}) \times R(\alpha,\beta,\gamma) \times m'_{3,j,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(m'_{j,k}) \times R(\alpha,\beta,\gamma) \times m'_{4,j,k}\right) \\ n_{i,k}^T \times \left(R_{nav}(m'_{j,k}) \times R(\alpha,\beta,\gamma) \times \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T\right) \\ \dots \end{pmatrix} \\ \\ i \text{ et } j \neq 15 \end{cases}$$

$A_{i,j,k}$ est un vecteur rempli de 0, sauf pour les quelques éléments présentés en équation 4.4. Le vecteur des offsets qui minimise la fonction objectif (4.4) est la solution du système linéaire :

$$C_{int} \times \delta X_{int} = -V_{int} \tag{4.5}$$

avec :

$$\begin{cases} C_{int} = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_k w_{i,j,k} * (A_{i,j,k} \times A_{i,j,k}^T) \\ V_{int} = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_k w_{i,j,k} * (B_{i,j,k} \times A_{i,j,k}) \end{cases}$$

Pour trouver les offsets de calibrage optimaux, et comme nous avons linéarisé certains paramètres, nous calculons δX de façon itérative, jusqu'à convergence des offsets. A chaque nouvelle itération n+1, les paramètres de calibrage intrinsèque sont définis comme étant corrigés par les offsets calculés à l'itération précédente :

$$X_{n+1} = X_n + \delta X_n$$

Pour démarrer l'optimisation, nous partons d'offsets nuls car le but de l'optimisation est de calculer et ajouter des offsets aux paramètres de calibrage intrinsèque pour affiner le nuage de points : ces offsets n'existent pas avant l'optimisation. Il est toutefois possible de rajouter quelques offsets pour « déformer » le nuage de points et sortir d'un minimum local : en effet, bien qu'il soit possible d'améliorer le modèle donné par le constructeur, celui-ci est assez précis et certains tests ont montré qu'il valait mieux ajouter des offsets peu élevés au démarrage de l'optimisation pour que la convergence soit optimale.
L'algorithme complet d'optimisation des paramètres de calibrage intrinsèque est similaire à l'algorithme 1 pour l'optimisation des paramètres extrinsèques : ce qui change entre ces deux optimisations sont les paramètres que l'on optimise. Le critère d'arrêt est le même : 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. Ces paramètres δ et n_max sont les mêmes que pour l'optimisation des paramètres extrinsèques.

4.3.2 Validation du résultat de l'optimisation

La validation du résultat d'optimisation est la même que celle présentée dans la section 3.3.3, puisque nous utilisons la même fonctionnelle pour l'optimisation, et que la procédure d'optimisation est aussi similaire.

Aussi, nous avons défini une métrique d'erreur pour chaque type d'offset de paramètres intrinsèques :

$$\begin{cases} \Delta\rho = \sqrt{\frac{1}{31} * \sum_{i=0/i\neq 15}^{31} (\delta\rho_i)^2} \\ \Delta\theta = \sqrt{\frac{1}{31} * \sum_{i=0/i\neq 15}^{31} (\delta\theta_i)^2} \\ \Delta\phi = \sqrt{\frac{1}{31} * \sum_{i=0/i\neq 15}^{31} (\delta\phi_i)^2} \\ \Delta H_z = \sqrt{\frac{1}{31} * \sum_{i=0/i\neq 15}^{31} (\delta H_{z,i})^2} \end{cases} \tag{4.6}$$

Dans le cas où la vérité terrain est connue et que l'on déforme le nuage, on peut comparer ces erreurs avant et après optimisation, et le résultat espéré est une erreur faible pour chaque type de paramètre intrinsèque. Dans le cas où la vérité terrain n'est pas connue, la validation se fait d'une part avec la valeur finale de l'énergie J, mais aussi avec le fait que ces erreurs doivent être faibles : en effet, on ajoute des offsets sur les paramètres de calibrage intrinsèque, mais le modèle du constructeur étant assez proche de la réalité, les offsets ne doivent pas être très grands.

4.4 Résultats expérimentaux

Cette optimisation a été codée en C++ et utilise les mêmes librairies EIGEN et FLANN que pour l'optimisation des paramètres extrinsèques. Pour les paramètres à régler dans notre optimisation, les mêmes que ceux définis en section 3.4.2 ont été utilisés afin d'aller dans la continuité de l'optimisation des paramètres de calibrage du système mobile.

Enfin, au niveau des paramètres intrinsèques avec lesquels nous initialisons l'algorithme, il y a 2 cas de figure :

- les données sont simulées ; nous avons alors une vérité terrain des paramètres intrinsèques, et une erreur est ajoutée pour chacun des 4 paramètres à optimiser de chaque fibre, sauf pour la fibre référence pour laquelle on considère que les paramètres sont optimaux. Le but de l'optimisation est alors d'obtenir des offsets qui donnent des paramètres intrinsèques corrigés aussi proches que possible de ceux donnés par la vérité terrain.
- les données sont issues d'acquisitions réelles ; dans ce cas là, nous n'avons pas accès à la vérité terrain. Pour tester la robustesse de notre optimisation, une erreur est aussi ajoutée à tous les paramètres de chaque fibre sauf pour la fibre référence du Velodyne : en effet, pour certains tests d'optimisation, aucune différence entre le nuage avant et après optimisation n'était visible. Cela nous permettait ainsi de tester notre optimisation dans le cas où les paramètres intrinsèques étaient « trop » erronés.

Dans la suite de cette section, nous allons présenter 4 résultats d'optimisation des paramètres intrinsèques : les résultats concernent les nuages simulés #2 et #3, et les nuages réels #4 et #5 introduits en section 3.4.1.

4.4.1 Résultats sur des jeux de données simulées

Pour les tests d'optimisation sur des jeux de données simulées, nous avons une vérité terrain, qui est composée des paramètres de calibrage du nuage de point. Nous avons ajouté des erreurs

arbitraires à ces valeurs de paramètres intrinsèques, et le but de l'optimisation est d'obtenir au final des offsets les plus proches de 0. Les erreurs ajoutées aux paramètres intrinsèques sont de l'ordre de -3° à 3° pour les paramètres angulaires ϕ et θ, et entre -10 cm et 10 cm pour les paramètres de distance et de translation ρ et H_z : cela a aussi permis de tester la robustesse de notre optimisation à une initialisation trop mauvaise.

Le premier nuage simulé est le nuage #2 présenté en figure 4.3. L'évolution de l'énergie est présentée avec la figure 4.4, et on peut voir que l'énergie n'est pas strictement décroissante : ce n'est pas surprenant car nous utilisons une approche de type recalage de données ; or, dans [Besl et McKay, 1992], l'énergie mesurée à chaque itération du recalage est strictement décroissante dans le cas où le facteur de normalisation est le même à chaque itération, et cela a été prouvé. Dans notre cas, nous autorisons de plus en plus de points appariés à être pris en compte, ce qui a pour effet d'augmenter le poids total utilisé pour la normalisation à chaque itération, et ce qui justifie que l'énergie augmente à certaines itérations, tout en convergeant vers une valeur minimale en fin d'optimisation. L'énergie varie d'une valeur de 265.13 cm^2 à une valeur de 0.45 cm^2 avec l'optimisation. La table 4.1 donne les erreurs définies en section 4.3.2 pour chaque catégorie de paramètres : on peut voir que pour l'ensemble des offsets concernés, nous avons des valeurs très proche de 0, ce qui permet de valider le résultat d'optimisation.

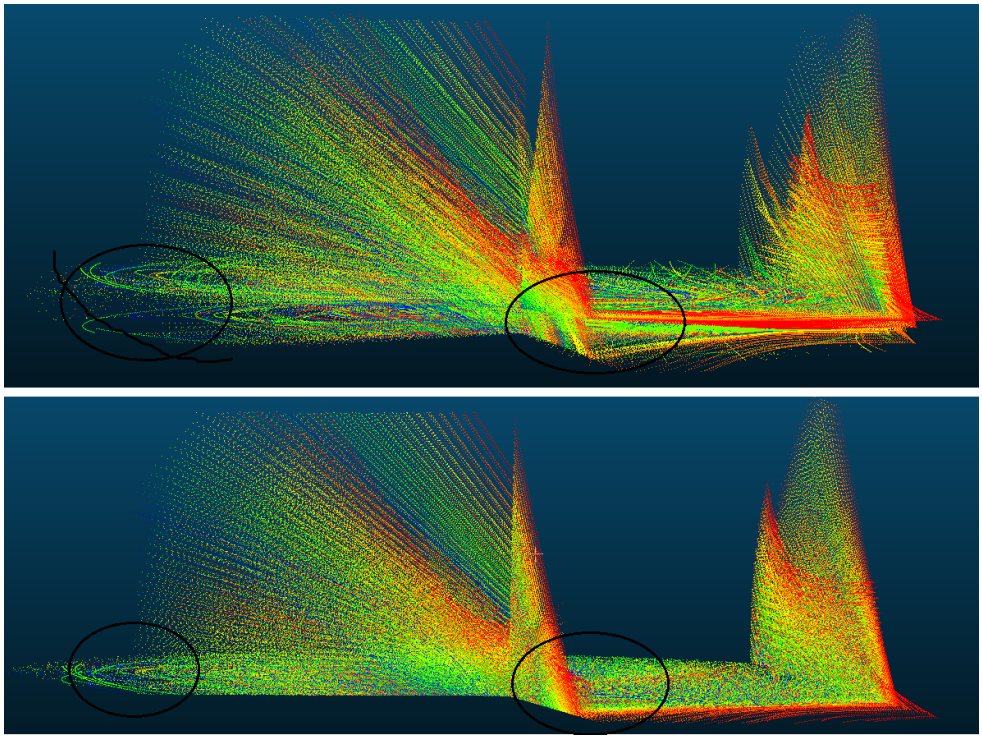


FIGURE 4.3 – Nuage simulé #2 : en haut, nuage avant optimisation des paramètres intrinsèques ; en dessous, le même nuage après optimisation des paramètres intrinsèques. Les deux images sont vues avec la même orientation.

	$\Delta\rho(cm)$	$\Delta\theta(^\circ)$	$\Delta\phi(^\circ)$	$\Delta H_z(cm)$
Erreurs initiales	10	2.5	3	10
Erreurs après notre optimisation	$1.52 * 10^{-2}$	$1.38 * 10^{-3}$	$7.81 * 10^{-4}$	$1.19 * 10^{-2}$

TABLE 4.1 – Erreurs entre les offsets et la vérité terrain pour le nuage simulé #2

Pour le nuage #3, les observations sont les mêmes : le résultat d'optimisation est similaire à

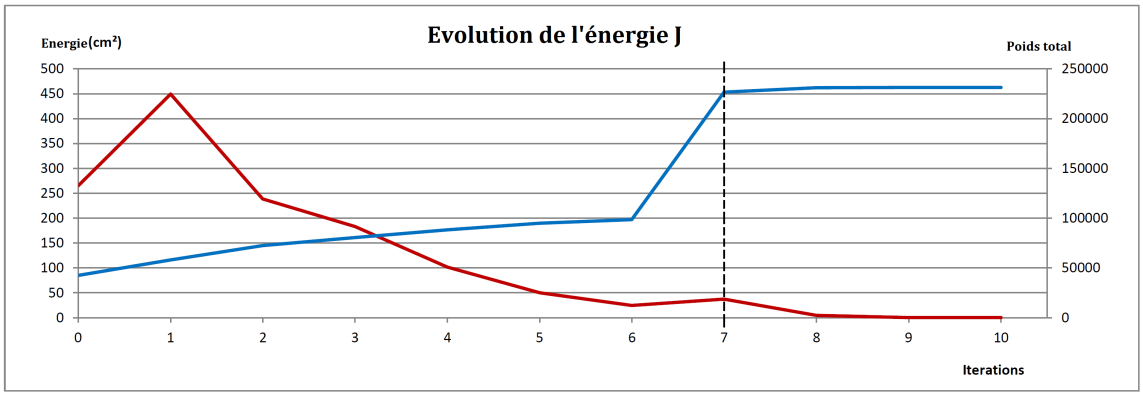


FIGURE 4.4 – Évolution de l'énergie pour le nuage simulé #2 : en rouge, nous avons la courbe d'énergie en fonction des itérations ; en bleu, nous avons la courbe représentant le poids total utilisé pour normaliser l'énergie ; les lignes noires verticales mettent en évidence les itérations où les attributs de dimensionnalité sont mis à jour.

celui présenté avec le nuage #2, et plus de détails sont donnés en Annexe D.

Enfin, concernant les temps de calculs pour les optimisations des nuages #2 et #3, ils sont respectivement de 9 minutes et 5 minutes, ce qui est acceptable au vu du nombre de paramètres optimisés.

4.4.2 Résultats sur des jeux de données réelles

Dans cette section, nous présentons des résultats d'optimisation sur 2 jeux de données réelles. Cette fois-ci, nous n'avons pas de vérité terrain comme pour les jeux de données simulées : nous partons alors du modèle donné par le constructeur en ajoutant les mêmes offsets que pour les jeux de données simulées, et les résultats attendus sont des offsets plus faibles, qui donnent une énergie plus petite et un nuage correctement affiné.

La figure 4.6 montre les améliorations après optimisation des offsets intrinsèques pour le nuage #4 : en haut, nous présentons le nuage avec des offsets intrinsèques initiaux, et la figure du bas présente le même nuage avec des offset optimisés, où le nuage est correctement affiné. La figure 4.5 présente l'évolution de l'énergie au cours de l'optimisation pour le nuage #4 : elle évolue d'une valeur de 278.47 cm^2 à une valeur de 33.40 cm^2, ce qui montre que l'on affine correctement le nuage de points ; aussi, la valeur de l'énergie est validée par notre critère défini en section 3.3.3. On peut remarquer sur cette figure que l'énergie d'optimisation commence par diminuer, puis augmente jusqu'à la mise à jour des attributs de dimensionnalité, avant de diminuer à nouveau jusqu'à convergence ; ce résultat peut-être expliqué du fait qu'entre les itérations 1 à 7, nous utilisons les mêmes valeurs d'attributs de dimensionnalité à chaque itération. Ces attibuts sont calculés à l'aide des normales estimées en chaque point du nuage de point, et à chaque itération, les paramètres de calibrages sont corrigés, ce qui modifie la structure du nuage, ce qui implique que les attributs de dimensionnalité devraient changer. Or, pour des soucis de temps de calculs, nous ne les mettons à jour que toutes les 7 itérations, ce qui explique cette augmentation de l'énergie jusqu'à la première mise à jour des attributs.

Pour le nuage réel #5, nous avons les mêmes observations que pour le nuage #4 : les résultats sont similaires à ceux présentés pour le jeu de données #4, et sont plus détaillés en Annexe D.

Pour les temps de calculs des optimisations des jeux de données réelles, nous avons respectivement 25 minutes pour le nuage #4 et 12 minutes pour le nuage #5, ce qui est acceptable au vu de la taille des nuages et de la quantité de paramètres optimisés.

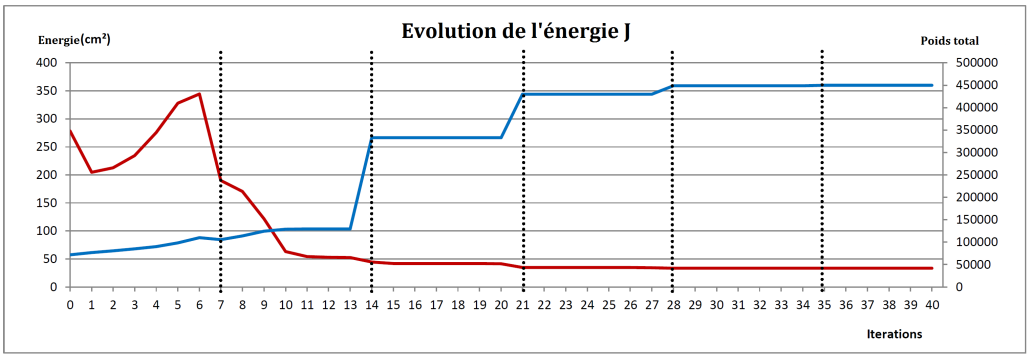


FIGURE 4.5 – Évolution de l'énergie pour le nuage réel #4 : en rouge, nous avons la courbe d'énergie en fonction des itérations ; en bleu, nous avons la courbe représentant le poids total utilisé pour normaliser l'énergie ; les lignes noires verticales mettent en évidence les itérations où les attributs de dimensionnalité sont mis à jour.

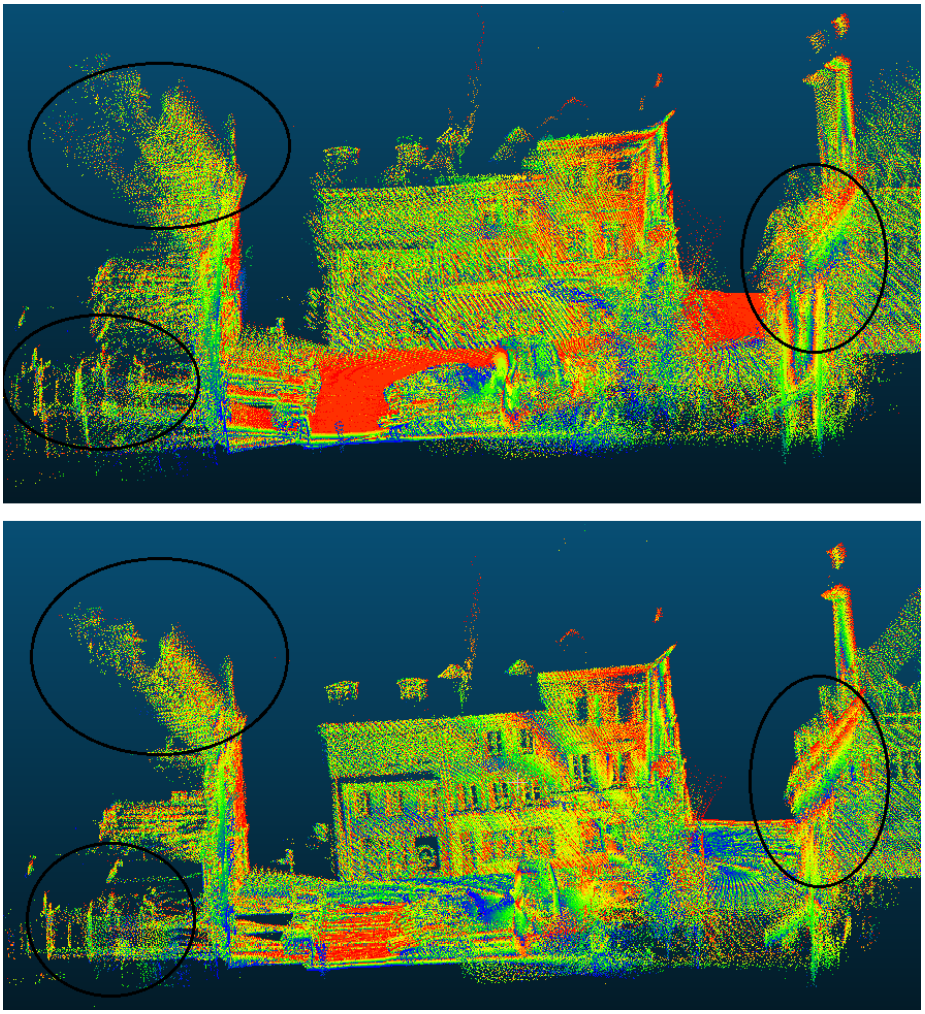


FIGURE 4.6 – Nuage réel #4 : en haut, nuage avant optimisation des paramètres intrinsèques ; en dessous, le même nuage après optimisation des paramètres intrinsèques. Les deux images sont vues depuis le même point.

4.5 Optimisation conjointe avec les paramètres extrinsèques

Dans le chapitre 3, nous avons présenté une optimisation des paramètres extrinsèques en faisant la supposition que les paramètres intrinsèques étaient correctement estimés. Dans ce chapitre, nous avons présenté l'optimisation du modèle de calibrage intrinsèque en supposant que cette fois-ci, il s'agissait des paramètres extrinsèques qui étaient correctement étalonnés. Les deux optimisations utilisent la même fonctionnelle à minimiser, et l'algorithme est le même : nous avons présenté des résultats d'optimisations avec des paramètres égaux. Toutefois, il n'est pas toujours possible de faire ces hypothèses, et en général, les deux étalonnages ne sont pas optimaux. Dans cette section, nous allons proposer une optimisation conjointe des paramètres de calibrage extrinsèque et intrinsèque.

4.5.1 Changements sur la résolution numérique

Pour l'optimisation conjointe des paramètres de calibrage, nous avons repris la fonctionnelle (3.6), et les paramètres à optimiser sont ceux définis dans les sections 3.3.1 et 4.3.1. Nous effectuons aussi une linéarisation au premier ordre, et cela conduit au système linéaire suivant à résoudre :

$$C_{tot} \times \delta X_{tot} = -V_{tot} \tag{4.7}$$

avec :

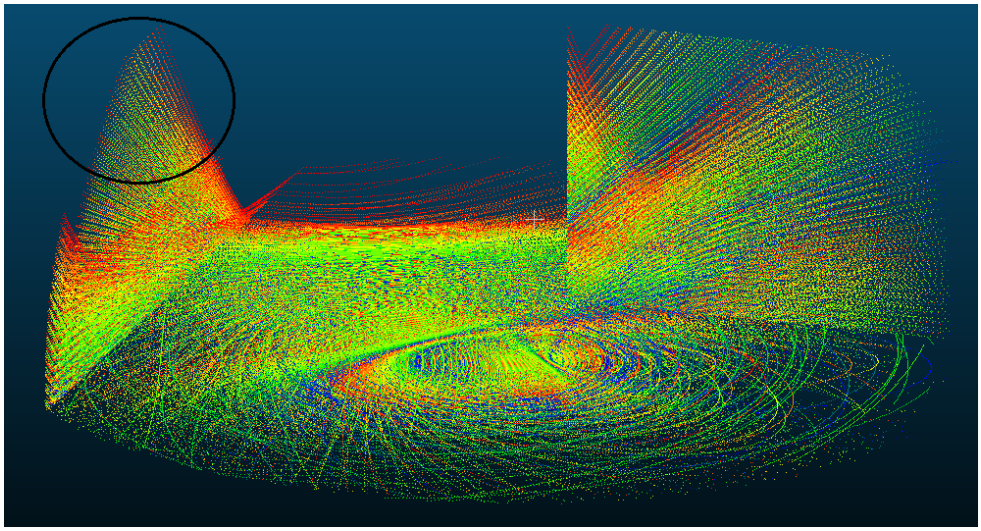
$$\begin{cases} \delta X_{tot} = [\delta X_{int} \quad \delta X_{ext}]^T \\ C_{tot} = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * (E_{i,j,k} \times E_{i,j,k}^T) \\ E_{i,j,k} = [A_{i,j,k}^T \quad C_{i,j,k}^T]^T \\ V_{tot} = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * (B_{i,j,k} \times E_{i,j,k}) \end{cases}$$

Les paramètres intrinsèques et extrinsèques à optimiser sont concaténés en un seul vecteur d'inconnues, qui est itérativement calculé comme cela a été présenté plus tôt.
$C_{i,j,k}$ est le vecteur introduit en équation 3.7. Le terme constant $B_{i,j,k}$ est égal au terme $D_{i,j,k}$ lui aussi introduit en équation 3.7.

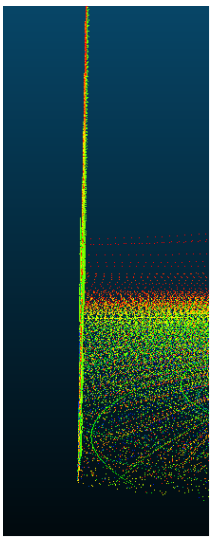
4.5.2 Comparaison des approches d'optimisations successives et d'optimisation conjointe des paramètres intrinsèques et extrinsèques

Dans un premier temps, nous avons comparé des optimisations successives des paramètres extrinsèques et intrinsèques, dans les deux sens, c'est à dire en effectuant l'optimisation extrinsèque suivie de l'optimisation intrinsèque dans un premier temps, et inversement ensuite. Pour comparer ces approches d'optimisation des paramètres de calibrage, nous avons pris les 2 jeux de données simulées #2 et #3, présentés en section 3.4.

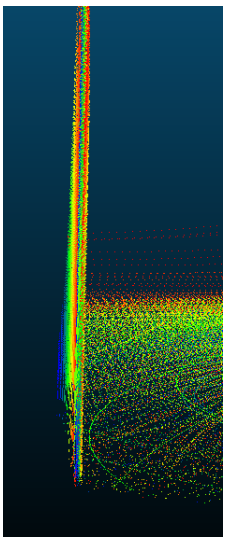
La figure 4.7 présente le résultat d'optimisation avec les 3 approches testées, lorsque au départ les paramètres extrinsèques et intrinsèques sont erronés. Les erreurs sur les paramètres de calibrage extrinsèque sont de l'ordre de 50 cm pour les translations selon les axes x et y, et de 80 cm selon l'axe z, et des erreurs de respectivement 2.5, 3 et -2° pour les rotations roulis, tangage et lacet. Pour les erreurs des paramètres intrinsèques, elles sont de l'ordre de 0.3° pour les paramètres liés aux rotations, et d'environ 3 cm pour les paramètres de distance et d'offset vertical.



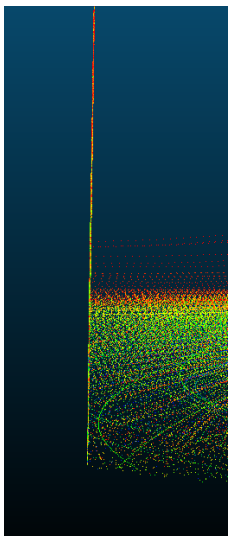
(a)



(b)



(c)



(d)

FIGURE 4.7 – Nuages de point #2 avec plusieurs optimisations des paramètres de calibrage : a) Paramètres corrects ; b) Optimisation extrinsèque puis intrinsèque ; c) Optimisation intrinsèque puis extrinsèque ; d) Optimisation conjointe des paramètres de calibrage

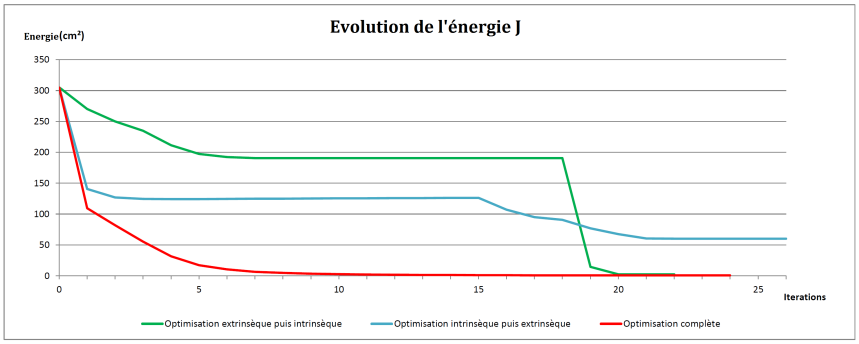


FIGURE 4.8 – Comparaison des énergies avec les 3 approches d'optimisations pour le nuage #2