Une Méthode Plug-and-play pour le Recalage de Nuages de Points

M. BOUZEID[1,2] P. BRUEL[3] S. LABSIR[4] V. POULAIN[3] J. TACHELLA[5] J.-Y. TOURNERET[1,2] D. YOUSSEFI[6]

[1]TéSA laboratory, Toulouse, France

[2]IRIT-ENSEEIHT, Univ. Toulouse, Toulouse, France

[3]Thales Services Numériques, Toulouse, France

[4]IPSA, Toulouse, France

[5]CNRS, ENS de Lyon, Lyon, France

[6]CNES, Toulouse, France

Résumé – Cet article présente une extension d'une approche plug-and-play pour le recalage de nuages de points 3D. Le problème de recalage de nuages de points 3D est formulé comme un problème inverse, et une approche plug-and-play est utilisée pour conjointement débruiter et recaler les nuages de points. Dans cet article, nous proposons d'optimiser la transformation de recalage en exploitant la structure de groupe de Lie de la transformation rigide $SE(3)$. Des expériences menées sur des nuages de points LiDAR sont présentées mettant en évidence l'amélioration de la méthode par rapport à une méthode existante.

Abstract – This paper presents an extension of a plug-and-play approach for 3D point cloud registration. The 3D point cloud registration problem is formulated as an inverse problem, and a plug-and-play approach is used to jointly denoise and register two point clouds.This work proposes to optimize the registration transformation by leveraging the Lie group structure of the rigid transformation $SE(3)$. Experiments conducted on LiDAR point clouds demonstrate the improvement of the proposed method with respect to an existing method.

1 Introduction

Les technologies comme le LiDAR, la photogrammétrie et la reconstruction 3D ont amélioré la modélisation des environnements urbains. Pour pouvoir traiter et fusionner les différentes données issues de ces capteurs, il est nécessaire de recaler des nuages de points issus de capteurs différents, ce qui reste un défi majeur pour ces données. Le recalage consiste à trouver la transformation géométrique (rotation, translation) entre deux nuages de points. Les différentes méthodes de recalage de nuages de points 3D peuvent se diviser en deux catégories. La première consiste à minimiser une fonction de coût sans connaître les correspondances entre les points des deux nuages, ces fonctions de coût variant d'une méthode à une autre [1] [2] [3]. L'ICP (Iterative Closest Point) [4] est la méthode de référence dans cette catégorie. L'autre catégorie regroupe les méthodes se concentrant sur l'association de points afin de trouver la transformation optimale les reliant. Ces méthodes utilisent notamment la détection de points d'intérêt pour réaliser ces associations [5].

Les méthodes plug-and-play (PnP), largement utilisées dans la résolution de problèmes inverses pour le traitement d'images [6], permettent d'intégrer implicitement des informations a priori en exploitant un débruiteur pré-entraîné. Dans un précédent article [7], nous avons adapté une méthode PnP au problème de recalage de nuages de points 3D. Cette méthode utilise notamment une estimation de la matrice de rotation entre les deux nuages à l'aide d'une méthode de descente de gradient. Cependant, cette approche n'utilise pas de paramétrisation pour la matrice de transformation et n'exploite donc pas les propriétés spécifiques de la rotation. Dans cet article, nous proposons de recourir à la théorie des groupes de Lie [8] [9] pour paramétrer la transformation entre les deux nuages en exploitant pleinement la géométrie du problème de recalage. Ainsi, on garantit que la transformation préserve les propriétés de la rotation et de la translation la définissant. De plus, cette théorie fournit une estimation plus précise de la transformation entre les deux nuages de points avec un coût calculatoire réduit.

Cet article est organisé de la manière suivante. La section 2 explique comment la méthode PnP peut être utilisée pour le recalage de nuages de points et comment elle peut être améliorée en utilisant la théorie des groupes de Lie. La section 3 compare les performances de la méthode proposée avec l'état de l'art. Nos conclusions sont indiquées dans la section 4.

2 Recalage de nuages de points 3D

Les méthodes PnP reposent sur l'utilisation d'un débruiteur pour intégrer une forme de régularisation implicite dans la résolution de problèmes inverses. Plutôt que d'employer des termes de régularisation classiques, ces approches exploitent les propriétés du débruiteur pour guider la solution. Cette partie présente tout d'abord le problème de recalage et la méthode de résolution proposée dans [7]. Ensuite, nous expliquons comment exploiter la théorie des groupes de Lie pour améliorer les performances de l'algorithme initial.

2.1 Formulation du problème

Dans un premier temps, nous allons formuler notre problème de recalage de nuages de points 3D comme un problème inverse. Cette formulation s'inspire des travaux de [10] et adapte

leur méthode au recalage de nuages de points. On considère trois nuages de points définis par :

$$\boldsymbol{y}_1 = \{\boldsymbol{y}_{11}, \dots, \boldsymbol{y}_{1N}\}, \boldsymbol{y}_{1i} \in \mathbb{R}^4,$$
$$\boldsymbol{y}_2 = \{\boldsymbol{y}_{21}, \dots, \boldsymbol{y}_{2M}\}, \boldsymbol{y}_{2i} \in \mathbb{R}^4,$$
$$\boldsymbol{x} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\}, \boldsymbol{x}_i \in \mathbb{R}^4,$$

où $\boldsymbol{y}_1$ et $\boldsymbol{y}_2$ sont les données observées, $\boldsymbol{x}$ est le nuage de points recherché et, N et M sont les nombres de points des nuages. Bien qu'un point 3D se définisse par trois coordonnées dans l'espace, nous utilisons les coordonnées homogènes qui ajoutent la valeur 1 à chaque vecteur de l'ensemble. Ce choix permet de définir la matrice de transformation comme suit :

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & R_3 & t_1 \\ R_4 & R_5 & R_6 & t_2 \\ R_7 & R_8 & R_9 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathcal{M}_4(\mathbb{R}). \quad (1)$$

Le problème de recalage peut alors s'écrire :

$$\begin{cases} \boldsymbol{y}_{1i} &= \boldsymbol{T}\boldsymbol{x}_{m(i)} + \boldsymbol{\epsilon}_{1i}, \\ \boldsymbol{y}_{2i} &= \boldsymbol{x}_i + \boldsymbol{\epsilon}_{2i}, \end{cases} \quad (2)$$

où $m(i)$ ($\forall i = 1, \dots, M$) renvoie un nombre compris entre 1 et N représentant l'indice du point de $\boldsymbol{y}_1$ associé à $\boldsymbol{y}_{2i}$, $\boldsymbol{\epsilon}_{1i} \sim \mathcal{N}(\boldsymbol{0}, \sigma_1^2\mathbf{I})$ et $\boldsymbol{\epsilon}_{2i} \sim \mathcal{N}(\boldsymbol{0}, \sigma_2^2\mathbf{I})$. On précise que la fonction d'association m n'est pas connue. Similairement à ce qui a été réalisé dans [10], on définit notre problème de recalage comme un problème inverse nécessitant d'estimer $\boldsymbol{x}$ et $\boldsymbol{T}$:

$$\underset{\boldsymbol{x},\boldsymbol{T}}{\operatorname{argmin}} \left[f(\boldsymbol{x}, \boldsymbol{T}; \lambda_1) - \lambda_2 \log p_X(\boldsymbol{x})\right], \quad (3)$$

avec

$$f(\boldsymbol{x}, \boldsymbol{T}; \lambda_1) = d(\boldsymbol{y}_1, \boldsymbol{T}\boldsymbol{x}) + \lambda_1 d(\boldsymbol{y}_2, \boldsymbol{x}),^{1}$$

où d est une mesure de dissimilarité, $p_X(\boldsymbol{x})$ est la loi a priori de $\boldsymbol{x}$ et λ_1 et λ_2 sont des hyperparamètres à régler par l'utilisateur. Afin d'obtenir une fonction adaptée aux nuages de points, nous proposons de remplacer l'erreur quadratique moyenne, habituellement utilisée comme mesure de dissimilarité entre deux images, par la distance de Chamfer définie par :

$$d(\boldsymbol{s}, \boldsymbol{q}) = \sum_{i=1}^{N} \min_k \frac{\|\boldsymbol{s}_i - \boldsymbol{q}_k\|^2}{N} + \sum_{k=1}^{M} \min_i \frac{\|\boldsymbol{s}_i - \boldsymbol{q}_k\|^2}{M}, \quad (4)$$

où $\boldsymbol{s} = \{\boldsymbol{s}_i\}_{i=1,\dots,N}$ et $\boldsymbol{q} = \{\boldsymbol{q}_i\}_{i=1,\dots,M}$ sont deux ensembles de points. La distance de Chamfer ne nécessite pas que les éléments comparés soient de même taille et ordonnés et n'exige pas de choisir une association entre les points en amont (la fonction d'association m n'est donc pas estimée). En effet, d'après (4), la distance de Chamfer utilise le plus proche voisin pour effectuer l'association de points.

2.2 Méthode de résolution

On peut appliquer un algorithme de minimisation alternée pour estimer $\boldsymbol{x}$ et $\boldsymbol{T}$. L'idée des méthodes PnP est de construire le gradient de $p_X(\boldsymbol{x})$ à l'aide d'un débruiteur et de l'approcher à l'aide de la formule de Tweedie, comme expliqué dans [7].

[1] Par définition, $\boldsymbol{T}\boldsymbol{x} = \{\boldsymbol{T}\boldsymbol{x}_1, \dots, \boldsymbol{T}\boldsymbol{x}_M\}$.

Ainsi, en appliquant une descente de gradient sur (3), on obtient la mise à jour sur $\boldsymbol{x}$ suivante :

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_x \frac{\partial f(\boldsymbol{x}_k, \boldsymbol{T}_k; \lambda_1)}{\partial \boldsymbol{x}_k} + \alpha_x \lambda_2 \left[g\left(\mathrm{D}(\boldsymbol{x_k})\right) - \boldsymbol{x_k}\right],$$

où α_x est le pas de gradient pour la mise à jour de $\boldsymbol{x}_k$[2], D est un débruiteur de nuage de points et g est une fonction qui permet de réordonner le nuage de points débruité en trouvant les plus proches voisins. Pour cet article, nous avons utilisé le débruiteur Score-Denoise étudié dans [11]. Dans un second temps, il est possible d'estimer chaque composante de $\boldsymbol{T}$ avec l'algorithme de Gauss-Newton [12]. D'après (4), la distance de Chamfer $d(\boldsymbol{y}_1, \boldsymbol{T}\boldsymbol{x})$ qui est l'unique terme de (3) dépendant de $\boldsymbol{T}$, peut s'écrire $d(\boldsymbol{y}_1, \boldsymbol{T}\boldsymbol{x}) = \sum_{i=1}^{N+M} ||\boldsymbol{E}_i(\boldsymbol{T})||^2$ où :

$$\boldsymbol{E}_i(\boldsymbol{T}) = \begin{cases} \frac{1}{\sqrt{N}}(\boldsymbol{y}_{1i} - \boldsymbol{T}\boldsymbol{x}_{p_i}) \text{ si } i = 1, \dots, N, \\ \frac{1}{\sqrt{M}}(\boldsymbol{y}_{1\ell_i} - \boldsymbol{T}\boldsymbol{x}_{i-N}) \text{ si } i = N+1, \dots, N+M, \end{cases}$$

avec $p_i = \underset{j}{\operatorname{argmin}} \|\boldsymbol{y}_{1i} - \boldsymbol{T}\boldsymbol{x}_j\|^2$ et $\ell_i = \underset{j}{\operatorname{argmin}} \|\boldsymbol{y}_{1j} - \boldsymbol{T}\boldsymbol{x}_i\|^2$. On remarquera que $\sum_{i=1}^{N} ||\boldsymbol{E}_i(\boldsymbol{T})||^2$ est équivalent au premier terme de la distance de Chamfer (4) et $\sum_{i=N+1}^{N+M} ||\boldsymbol{E}_i(\boldsymbol{T})||^2$ au second terme. À l'itération k, la mise à jour de la variable $\boldsymbol{T}$ est finalement définie par :

$$\boldsymbol{T}_{k+1} = \boldsymbol{T}_k - \left(\boldsymbol{J}^\top(\boldsymbol{T}_k)\boldsymbol{J}(\boldsymbol{T}_k)\right)^{-1} \boldsymbol{J}^\top(\boldsymbol{T}_k)\boldsymbol{E}(\boldsymbol{T}_k), \quad (5)$$

avec $\boldsymbol{E}(\boldsymbol{T}) = \begin{bmatrix} \boldsymbol{E}_1^\top(\boldsymbol{T}) & \dots & \boldsymbol{E}_{N+M}^\top(\boldsymbol{T}) \end{bmatrix}^\top \in \mathbb{R}^{4(N+M)\times 1}$ et $\boldsymbol{J}$ est la matrice jacobienne de $\boldsymbol{E}$. Les étapes de la version initiale de cette méthode PnP sont résumées dans l'algorithme 1 avec "lie = False".

2.3 Utilisation du groupe de Lie $SE(3)$

Au lieu de déterminer la matrice de transformation $\boldsymbol{T}$ à l'aide de (1) qui estime douze paramètres indépendamment les uns des autres (neuf paramètres pour la rotation et trois paramètres pour la translation), nous proposons une mise à jour exploitant le fait que la matrice $\boldsymbol{T}$ appartient au groupe de Lie $SE(3)$ (trois paramètres pour la rotation et trois paramètres pour la translation). Avant de s'intéresser à cette mise à jour, nous allons introduire la théorie des groupes de Lie.

Un groupe de Lie est une variété lisse munie d'une structure de groupe. Les groupes de Lie permettent de se déplacer de manière continue à l'intérieur d'un espace en respectant les propriétés géométriques des éléments de cet espace. Pour être une matrice de rotation, $\boldsymbol{R}$ doit vérifier les propriétés :

$$\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I} \text{ et } \det(\boldsymbol{R}) = 1. \quad (6)$$

Il faut également définir la loi de groupe afin de garantir un déplacement intrinsèque sur le groupe de Lie. La loi du groupe des matrices de la forme (1) est la multiplication matricielle.

On définit l'algèbre de Lie comme l'espace tangent à l'élément neutre du groupe de Lie, ici le groupe des transformations rigides qui admet la matrice identité comme élément neutre. L'algèbre de Lie correspond donc à une approximation locale plus simple à manipuler que le groupe de Lie lui-même. Les

[2] La fonction minimum étant différentiable presque partout, nous utilisons PyTorch Autograd, qui fournit un sous-gradient aux points non différentiables.

matrices de transformation rigide, telles que définies dans (1), constituent le groupe de Lie $SE(3)$ [8]. Ce groupe garantit que les propriétés (6) sont vérifiées pour la rotation de $\boldsymbol{T}$. L'algèbre de Lie et l'espace euclidien associés sont $\mathfrak{se}(3)$ et $\mathbb{R}^6$. On peut réécrire les éléments de ces ensembles comme suit :

$$\begin{bmatrix} 0 & -\omega_6 & \omega_5 & \omega_1 \\ \omega_6 & 0 & -\omega_4 & \omega_2 \\ -\omega_5 & \omega_4 & 0 & \omega_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(3) \text{ et } \boldsymbol{\omega} = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_6 \end{pmatrix} \in \mathbb{R}^6.$$

Il existe une bijection entre $SE(3)$, $\mathfrak{se}(3)$ et $\mathbb{R}^6$ au voisinage de l'identité. Afin de pouvoir appliquer l'algorithme de Gauss-Newton, nous avons besoin de la fonction qui permet de passer de l'espace euclidien au groupe de Lie. On définit cette fonction pour le groupe de Lie $SE(3)$ par :

$$\text{Exp}^{\wedge}_{SE(3)} : \mathbb{R}^6 \to SE(3)$$

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_6 \end{pmatrix} \mapsto \exp\left(\begin{bmatrix} 0 & -\omega_6 & \omega_5 & \omega_1 \\ \omega_6 & 0 & -\omega_4 & \omega_2 \\ -\omega_5 & \omega_4 & 0 & \omega_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right).$$

Puisque l'algorithme de Gauss-Newton (5) se fait itérativement avec un incrément euclidien, il est possible de l'adapter de manière intrinsèque sur $SE(3)$ à l'aide de l'opération de groupe. À travers cette loi, la notion de dérivée est redéfinie de la manière suivante : $\left.\frac{\partial \boldsymbol{E}_i\left(\boldsymbol{T}\text{Exp}^{\wedge}_{SE(3)}(\boldsymbol{\tau})\right)}{\partial \boldsymbol{\tau}}\right|_{\boldsymbol{\delta}=0} =$

$$\begin{cases} -\frac{1}{\sqrt{N}} \left.\frac{\partial \boldsymbol{T}\text{Exp}^{\wedge}_{SE(3)}(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}}\right|_{\boldsymbol{\delta}=0} \boldsymbol{x}_{k_i} \text{ si } i \leq N, \\ -\frac{1}{\sqrt{M}} \left.\frac{\partial \boldsymbol{T}\text{Exp}^{\wedge}_{SE(3)}(\boldsymbol{\tau})}{\partial \boldsymbol{\tau}}\right|_{\boldsymbol{\delta}=0} \boldsymbol{x}_{i-N} \text{ si } i \geq N+1, \end{cases}$$

où $\boldsymbol{\tau}$ représente un petit déplacement de $\mathbb{R}^6$. Notons que la dérivée de $\boldsymbol{T}$ sur $SE(3)$ est connue et peut s'exprimer en fonction des vecteurs de base de l'algèbre de Lie $\mathfrak{se}(3)$. De plus, la multiplication matricielle étant l'opération de groupe, nous obtenons la mise à jour suivante [9] :

$$\boldsymbol{T}_{k+1} = \boldsymbol{T}_k \text{Exp}\left([\boldsymbol{\delta}_k]^{\wedge}\right), \tag{7}$$

où $\boldsymbol{\delta}_k = -\left(\boldsymbol{J}^{\top}_{SE(3)}(\boldsymbol{T}_k)\boldsymbol{J}_{SE(3)}(\boldsymbol{T}_k)\right)^{-1} \boldsymbol{J}^{\top}_{SE(3)}(\boldsymbol{T}_k)\boldsymbol{E}(\boldsymbol{T}_k)$, avec $\boldsymbol{J}_{SE(3)}$ la matrice jacobienne de $\boldsymbol{E}$ sur $SE(3)$. L'adaptation de la mise à jour de $\boldsymbol{T}$ est détaillée dans l'algorithme 1 lorsque "lie=True".

3 Résultats

Afin de tester et comparer les méthodes de recalage basées sur (5) et (7), nous considérons des données LiDAR disponibles sur le site de l'Institut National de l'Information Géographique et Forestière (IGN)[3]. Nous nous sommes concentrés sur un nuage de points enregistré au niveau du stadium de Marseille en France. Afin d'éviter des problèmes de mémoire sur la carte graphique, nous avons sous-échantillonné les nuages de points de manière à obtenir un maximum de 30000 points. Une transformation rigide, composée d'une rotation et d'une translation a été appliquée à ce nuage de points qui a ensuite été

[3]https ://geoservices.ign.fr/lidarhd

Algorithm 1 Algorithme PnP

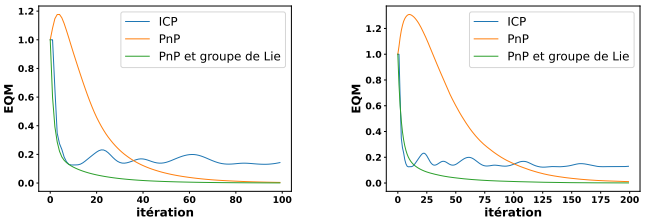
Entrées : $\boldsymbol{y}_1, \boldsymbol{y}_2, n_{\text{iter}}, D, \alpha_x, \lambda_1, \lambda_2$, lie
Sorties : $\boldsymbol{x}_{n_{\text{iter}}}, \boldsymbol{T}_{n_{\text{iter}}}$

$\boldsymbol{y}_1, \boldsymbol{y}_2 \leftarrow$ normalisation$(\boldsymbol{y}_1, \boldsymbol{y}_2)$
$\boldsymbol{x}_0 \leftarrow \boldsymbol{y}_2$
$\boldsymbol{T}_0 \leftarrow \boldsymbol{I}_4$
for $k \in \{1, \ldots, n_{\text{iter}}\}$ **do**
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \alpha_x \frac{\partial f(\boldsymbol{x}_k, \boldsymbol{T}_k; \lambda_1)}{\partial \boldsymbol{x}_k} + \alpha_x \lambda_2 (D(\boldsymbol{x_k}) - \boldsymbol{x_k})$
  if lie **then**
    $\boldsymbol{T}_{k+1} \leftarrow \boldsymbol{T}_k \text{Exp}\left([\boldsymbol{\delta}_k]^{\wedge}\right)$
  else
    $\boldsymbol{T}_{k+1} \leftarrow \boldsymbol{T}_k - \left(\boldsymbol{J}^{\top}(\boldsymbol{T}_k)\boldsymbol{J}(\boldsymbol{T}_k)\right)^{-1} \boldsymbol{J}^{\top}(\boldsymbol{T}_k)\boldsymbol{E}(\boldsymbol{T}_k)$
  end if
  $k \leftarrow k + 1$
end for

normalisé puis bruité avec un bruit additif gaussien (des bruits d'écart-types σ_1 et σ_2 ont été utilisés pour les nuages de points transformés $\boldsymbol{y}_1$ et $\boldsymbol{y}_2$). Pour les expériences présentées dans cet article, on considère le même niveau de bruit pour les deux nuages de points, i.e., $\sigma_1 = \sigma_2 \triangleq \sigma$. Nous avons quatre hyperparamètres à fixer pour nos expériences : α_x le pas de gradient de la mise à jour de $\boldsymbol{x}$, λ_1 qui permet d'ajuster l'importance respective des deux nuages de points, λ_2 qui définit le poids du terme de régularisation et n_{iter} le nombre d'itérations de la méthode. Notons que le choix de ces hyperparamètres a été effectué comme dans [7] et que leurs valeurs sont indiquées dans la Table 1. L'objectif de ces simulations est d'étudier l'intérêt de la mise à jour (7) par rapport à (5) pour résoudre le problème inverse lié au recalage de nuages de points 3D.

TABLE 1 : Valeurs des hyperparamètres pour les expériences.

Hyperparamètres	n_{iter}	$\alpha_{\boldsymbol{x}}$	λ_1	λ_2
Recalage avec $\sigma = 0.02$	100	0.1	46	0.3
Recalage avec $\sigma = 0.05$	200	0.1	46	0.1



(a) $\sigma = 0.02$. (b) $\sigma = 0.05$.

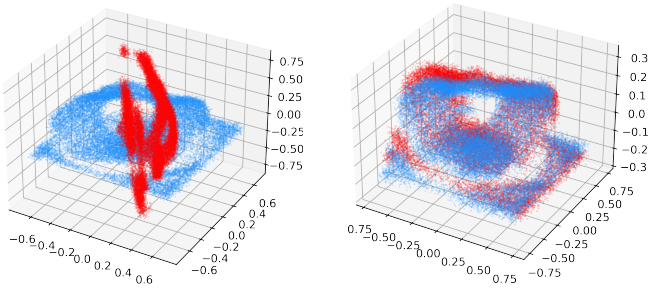
FIGURE 1 : Évolution de l'erreur quadratique moyenne normalisée entre la vérité-terrain et la transformation estimée.

Afin de comparer la différence entre la matrice de transformation $\boldsymbol{T}$ estimée et sa vérité-terrain, on utilise l'erreur quadratique moyenne normalisée définie par $\text{EQM} = \|\hat{\boldsymbol{T}} - \boldsymbol{T}_{\text{gt}}\|^2 / \|\boldsymbol{I}_4 - \boldsymbol{T}_{\text{gt}}\|^2$, où $\boldsymbol{T}_{\text{gt}}$ est la vérité-terrain, $\hat{\boldsymbol{T}}$ la transformation estimée et $\boldsymbol{I}_4$ la matrice identité (initialisation de notre estimation). Pour les courbes de Fig. 1, les tests ont été réalisés avec une transformation composée d'une rotation paramétrée par les angles d'Euler de valeurs $40°$, $10°$

et $20°$ sur les axes x, y et z ainsi qu'un vecteur de translation $\boldsymbol{t} = \begin{bmatrix}3 & 2 & 7\end{bmatrix}^\top$. Nous constatons que la méthode PnP converge vers une meilleure solution que ICP. De plus, nous remarquons également que l'utilisation des groupes de Lie permet de converger beaucoup plus rapidement vers la solution.

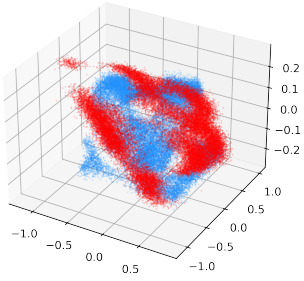
Nous nous sommes également intéressés au temps de calcul de la méthode PnP avec et sans utilisation des groupes de Lie. Le temps de calcul d'une seule itération de l'algorithme 1 est de 14 secondes lors de l'utilisation des groupes de Lie ("lie = True") et de 18 secondes sans ("lie = False"). L'une des opérations les plus coûteuses de cet algorithme est l'inversion matricielle utilisée dans Gauss-Newton, un coût qui est d'autant plus élevé que la taille de la matrice est importante. L'algorithme de Gauss-Newton sans groupe de Lie (5) nécessite l'inversion d'une matrice de taille 12×12, tandis qu'avec groupe de Lie (7) une matrice de taille 6×6 doit être inversée. L'utilisation des groupes de Lie pour le recalage engendre donc une diminution du temps d'exécution de l'algorithme.

D'autres expériences ont été réalisées avec une transformation composée d'une rotation avec les angles d'Euler de valeurs $40°$, $80°$ and $70°$ pour les axes x, y et z, ainsi qu'un vecteur de translation $\boldsymbol{t} = \begin{bmatrix}3 & 2 & 7\end{bmatrix}^\top$ et $\sigma = 0.02$. Comme on peut le constater sur Fig. 2, l'utilisation des groupes de Lie améliore les performances de l'estimation de $\boldsymbol{T}$, pour un même nombre d'itérations égal à 100.

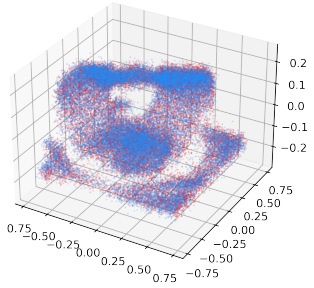


(a) Nuages de points initiaux normalisés à recaler ($\boldsymbol{y}_1$ en rouge et $\boldsymbol{y}_2$ en bleu).

(b) Nuages de points recalés avec ICP ($\hat{\boldsymbol{T}}^{-1}\boldsymbol{y}_1$ en rouge et $\boldsymbol{y}_2$ en bleu).



(c) Nuages de points recalés avec PnP ($\hat{\boldsymbol{T}}^{-1}\boldsymbol{y}_1$ en rouge et $\boldsymbol{y}_2$ en bleu).



(d) Nuages de points recalés avec PnP et groupe de Lie ($\hat{\boldsymbol{T}}^{-1}\boldsymbol{y}_1$ en rouge et $\boldsymbol{y}_2$ en bleu).

FIGURE 2 : Résultats de différentes méthodes de recalage de nuages de points 3D avec $n_{\text{iter}} = 100$.

4 Conclusion

Cet article a étudié une amélioration d'une méthode plug-and-play pour le recalage de nuages de points 3D en exploitant les propriétés géométriques des matrices de transformation rigide. L'utilisation des groupes de Lie permet de gagner en rapidité de convergence mais aussi de converger pour des exemples plus complexes lorsque d'autres méthodes échouent, avec un coût calculatoire réduit. La suite de ces travaux consistera à gérer le recalage de nuages de points ayant un recouvrement partiel et d'ajouter de la robustesse aux données aberrantes. Nous testerons également cette méthode sur des nuages de points 3D de sources différentes, notamment obtenus par photogrammétrie avec des images satellites comme Pléiades ou Pléiades Neo.

Références

[1] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK : Robust & Efficient Point Cloud Registration Using PointNet," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, (Long Beach, California, United States), June 2019.

[2] N. Bonneel and D. Coeurjolly, "Spot : sliced partial optimal transport," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–13, 2019.

[3] J. Ma, J. Zhao, and A. L. Yuille, "Non-rigid point set registration by preserving global and local structures," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 53–64, 2015.

[4] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Proc. SPIE*, vol. 1611, (Boston, Massachusetts, United States), pp. 586–606, Spie, 1992.

[5] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. Int. Conf. Robot. Autom.*, (Kobe, Japan), pp. 3212–3217, 2009.

[6] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, "Plug-and-Play Methods for Integrating Physical and Learned Models in Computational Imaging : Theory, algorithms, and applications," *IEEE Signal Processing Magazine*, vol. 40, no. 1, pp. 85–97, 2023.

[7] M. Bouzeid, P. Bruel, V. Poulain, J. Tachella, J.-Y. Tourneret, and D. Youssefi, "A plug-and-play approach for point cloud registration," in *Proc. IEEE Stat. Signal Process. (SSP)*, June 2025. Submitted.

[8] J. Sola, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv preprint arXiv :1812.01537*, 2018.

[9] S. Labsir, *Méthodes statistiques fondées sur les groupes de Lie pour le suivi d'un amas de débris spatiaux.* PhD thesis, Université de Bordeaux, 2020.

[10] Z. Kadkhodaie and E. Simoncelli, "Stochastic Solutions for Linear Inverse Problems using the Prior Implicit in a Denoiser," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, pp. 13242–13254, 2021.

[11] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proc. IEEE Int. Conf. Comput. Vis.*, (Montreal, Canada), pp. 4583–4592, 2021.

[12] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.