A PLUG-AND-PLAY APPROACH FOR POINT CLOUD REGISTRATION

M. Bouzeid[1,2], P. Bruel[3], V. Poulain[3], J. Tachella[4], J.-Y. Tourneret[1,2] and D. Youssefi[5]

[1] TéSA laboratory, Toulouse, France, maurine.bouzeid@tesa.prd.fr
[2] IRIT-ENSEEIHT, Univ. Toulouse, Toulouse, France, [3] Thales Services Numériques, Toulouse, France.
[4] CNRS, ENS de Lyon, Lyon, France, [5] CNES, Toulouse, France.

ABSTRACT

Plug-and-play algorithms have shown impressive results on imaging inverse problems, such as registration, super-resolution, denoising and inpainting. These methods rely on a neural network denoiser to learn an implicit prior of the image to be estimated. This paper investigates a new plug-and-play approach for 3D point cloud registration, which is crucial for a wide range of applications such as urban planning, archaeology and autonomous vehicles. The 3D point cloud registration problem is formulated as an inverse problem whose unknowns are the image to be estimated and the transformation between the two point clouds. A plug-and-play approach using an alternating optimization strategy is proposed for solving the registration problem. Experiments conducted on synthetic data and LiDAR point clouds are presented showing the potential of the method.

***Index Terms*—** Point cloud registration, point cloud denoising, inverse problems, Plug-and-play, Chamfer distance

1. INTRODUCTION

Over the past decades, the development of technologies such as LiDAR (Light Detection and Ranging) [1], photogrammetry [2] [3], 3D reconstruction [4] has improved the three-dimensional representation of urban environments. When using different 3D sensors, one important problem is to register the point clouds acquired. The aim of registration is to find the optimal geometric transformation (translation, rotation, scaling) that minimizes the difference between different point clouds. This transformation can either be rigid or non-rigid, where non-rigid transformations allow the scene to be deformed from one point cloud to another. This paper studies a new rigid point cloud registration method assuming that the transformation between two point clouds is a combination of a rotation and a translation, as in many recent works [5] [6] [7] [8]. The Iterative Closest Point (ICP) algorithm [9] has become a benchmark for 3D point cloud registration. ICP operates by alternating between two main steps: finding correspondences between the two point clouds using nearest neighbors and computing the optimal transformation to align these points. Since ICP is sensitive to the initialization of this transformation and can converge to local minima, methods have been developed to initialize ICP with a coarse transformation [7] [8]. Different efficient ICP variants were also studied in [10] to improve the performance of ICP.

Point cloud registration methods can be mainly divided into two classes. The first class of methods consists in minimizing a cost function without explicitly finding a mapping between the points, where the cost function is expressed using an appropriate distance such as the Wasserstein distance [6], a likelihood [11], or a difference between vector descriptors [5]. The SPOT algorithm [6] leverages optimal transport theory to align point clouds by minimizing the Wasserstein distance. Gaussian Mixture Models (GMM) were considered in [11] for point cloud registration, where the Expectation-Maximization (EM) algorithm was used to maximize the likelihood (ensuring robust alignment even in the presence of outliers and deformations). This method obtains good performances, but its computational complexity makes it difficult to apply to large point clouds and also it assumes that the point sets to be registered are not significantly distant from each other. PointNetLK [5] is a registration framework based on deep learning that extends the Lucas Kanade (LK) algorithm [12] to point clouds. Features are extracted at each point with PointNet [13] and the source cloud is iteratively transformed. While this approach has demonstrated high precision and robustness on complete point clouds, it struggles with partial registration and point clouds with outliers. The second class of registration methods identifies corresponding points in the two point clouds and computes the optimal transformation using these correspondences. Detecting and describing keypoints in point clouds is a critical step in this second category, as the quality of the correspondences significantly impacts the registration accuracy [14] [15] [16].

Plug-and-play methods [17] have recently received much attention to solve inverse problems by including a regularization term based on a denoiser. They have been shown to be effective in various image restoration problems such as denoising, super-resolution or inpainting [18]. Recent advancements in deep learning have revolutionized 3D point cloud denoising [19] [20] [21]. Neural networks can learn complex structures and adapt to different noise types with sufficient training data. Thus an efficient denoiser based on neural networks can be considered as a tool to learn data priors.

This paper investigates a new plug-and-play method for 3D point cloud registration. Inspired by the success of plug-and-play methods in the field of image restoration [18], we formulate the 3D point cloud registration problem as an inverse problem and propose a new plug-and-play algorithm to solve this inverse problem. This paper is organized as follows: Section 2 introduces the proposed inverse problem for point cloud registration and the plug-and-play algorithm for computing its solution. Section 3 presents experimental results allowing the performance of the proposed method to be evaluated. Conclusions and future work are reported in Section 4.

2. POINT CLOUD REGISTRATION

The main idea of plug-and-play methods is to solve inverse problems using an implicit prior defined by a denoiser. This denoiser replaces more standard regularization terms that are for instance used for point cloud registration. After summarizing the principles of plug-and-play approaches, this section describes the proposed point cloud registration method.

2.1. Plug-and-play methods

There is a vast literature of plug-and-play methods for image restoration [22]. In this paper, we follow the approach of Kadkodhaie [18]. This section recalls the principle of this method to solve inverse problems and adapt it to 3D point cloud registration. Consider a vectorized image $\boldsymbol{x} \in \mathbb{R}^N$ and its transformed and noisy version $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2\mathbf{I})$. The unknown image $\boldsymbol{x}$ can be estimated by its Maximum a Posteriori (MAP) estimator:

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p_{X|Y}(\boldsymbol{x}|\boldsymbol{y}).$$

This problem can be rewritten as the following inverse problem:

$$\hat{\boldsymbol{x}}_{\text{MAP}} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2 - \sigma_y^2 \log p_X(\boldsymbol{x}), \tag{1}$$

which can be solved by a gradient descent method:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) + \alpha\sigma_y^2 \nabla_{\boldsymbol{x}} \log p_X(\boldsymbol{x}_k), \tag{2}$$

where the initial value $\boldsymbol{x}_0$ and the stepsize α have to be adjusted by the user. In order to solve (2), the gradient of the prior of $\boldsymbol{x}$ has to be computed. However, choosing the prior can be difficult in some applications and a badly specified prior can lead to inaccurate results. An interesting idea is to approximate the gradient of the prior using Tweedie's identity [23]. Tweedie's identity for a noisy image $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\mathbf{I})$ can be written as [23]:

$$\nabla_{\mathbf{z}} \log p_Z(\boldsymbol{z}) = \frac{D_\sigma(\boldsymbol{z}) - \boldsymbol{z}}{\sigma^2},$$

where D_σ is the minimum mean square estimator of $\boldsymbol{x}$ for Gaussian noise with a standard deviation σ. Since $p_Z = p_X * p_\epsilon$, if the level of noise is small $\sigma \approx 0$, the following approximation $p_Z \approx p_X$ is obtained, leading to:

$$\nabla_{\mathbf{x}} \log p_X(\boldsymbol{x}) \approx \frac{D_\sigma(\boldsymbol{x}) - \boldsymbol{x}}{\sigma^2}. \tag{3}$$

As a consequence, the gradient descent (2) can be replaced by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) + \alpha\sigma_y^2 \left(\frac{D_\sigma(\boldsymbol{x_k}) - \boldsymbol{x_k}}{\sigma^2}\right), \tag{4}$$

which is the basis of plug-and-play approaches. Note that σ and σ_y are two different values, since σ is the standard deviation of the least-squares denoiser D_σ and σ_y is the standard deviation of the measurements. Intuitively, (4) indicates that the regularization term has a large value when the denoised version of $\boldsymbol{x}_k$ is far from $\boldsymbol{x}_k$, i.e., the noisier $\boldsymbol{x}_k$, the more important the regularization. The practical implementation of the gradient descent (4) requires to define the initial value $\boldsymbol{x}_0$, the stepsize α and the hyperparameter σ_y^2/σ^2.

2.2. Point cloud registration

The 3D point cloud registration problem can be formulated as an inverse problem similar to (1), allowing the plug-and-play framework to be applied. Consider two point clouds:

$$\boldsymbol{y}_1 = \{\boldsymbol{y}_{11}, \dots, \boldsymbol{y}_{1N}\}, \boldsymbol{y}_{1i} \in \mathbb{R}^4,$$

$$\boldsymbol{y}_2 = \{\boldsymbol{y}_{21}, \dots, \boldsymbol{y}_{2M}\}, \boldsymbol{y}_{2i} \in \mathbb{R}^4,$$

where N and M are the numbers of points of the two point clouds. Using the homogeneous coordinates, each point $\boldsymbol{p} \in \mathbb{R}^3$ can be rewritten as $\begin{bmatrix}\boldsymbol{p}\\1\end{bmatrix} \in \mathbb{R}^4$ and the transformation matrix $\boldsymbol{T}$ between the point clouds $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ can be defined as:

$$\boldsymbol{T} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ \boldsymbol{0} & 1\end{bmatrix} = \begin{bmatrix}R_1 & R_2 & R_3 & t_1\\ R_4 & R_5 & R_6 & t_2\\ R_7 & R_8 & R_9 & t_3\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

The point cloud registration problem can then be formulated as:

$$\begin{cases}\boldsymbol{y}_{1i} &= \boldsymbol{T}\boldsymbol{x}_{m(i)} + \boldsymbol{\epsilon}_1,\\ \boldsymbol{y}_{2i} &= \boldsymbol{x}_i + \boldsymbol{\epsilon}_2,\end{cases} \tag{5}$$

where $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are the two point clouds to be registered, $m(i)$ ($\forall i \in \{1, \dots, M\}$) returns a number between 1 and N that represents the index of the point in $\boldsymbol{y}_1$ associated with $\boldsymbol{y}_{2i}$, $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{0}, \sigma_{y_1}^2\mathbf{I})$ and $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(\boldsymbol{0}, \sigma_{y_2}^2\mathbf{I})$. This formulation allows a joint denoising and registration of the two point clouds, by estimating the unknown transformation matrix $\boldsymbol{T}$ jointly with the point cloud $\boldsymbol{x}$:

$$\boldsymbol{x} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\}, \boldsymbol{x}_i \in \mathbb{R}^4.$$

The inverse problem (5) allows us to leverage the plug-and-play framework to jointly denoise and register the two point clouds, which is an important contribution of this work. The data fidelity term $\frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|^2$ appearing in (1) is due to the additive Gaussian noise ϵ and can be viewed as a distance between $\boldsymbol{y}$ and $\boldsymbol{A}\boldsymbol{x}$. In order to take into account the differences between 3D point clouds and images, this similarity measure needs to be modified to be better adapted to 3D point clouds. Indeed, it can only be used to compare vectors having the same dimensions, which requires to know the association between the vectors of the two point clouds. When these associations are unknown, the Chamfer distance [24] can be used, leading to the following problem:

$$\underset{\boldsymbol{x},\boldsymbol{T}}{\operatorname{argmin}}\, f(\boldsymbol{x}, \boldsymbol{T}) - \lambda_2 \log p_X(\boldsymbol{x}), \tag{6}$$

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

The Chamfer distance d is defined as:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i=1}^{N} \min_{k=1,\dots,M} \frac{\|\boldsymbol{p}_i - \boldsymbol{q}_k\|^2}{N} + \sum_{k=1}^{M} \min_{i=1,\dots,N} \frac{\|\boldsymbol{p}_i - \boldsymbol{q}_k\|^2}{M}, \tag{7}$$

where $\boldsymbol{p} = \{\boldsymbol{p}_i\}_{i=1,\dots,N}$ and $\boldsymbol{q} = \{\boldsymbol{q}_i\}_{i=1,\dots,M}$ are two point clouds. The Chamfer distance does not need any association between the two point clouds, guarantees the permutation invariance and can be used for two point clouds with different sizes.

2.3. Optimization algorithms

This section presents the optimization algorithm that we propose to solve (6). The proposed strategy follows an alternating update scheme, where the point cloud $\boldsymbol{x}$ and the transformation $\boldsymbol{T}$ are iteratively updated in separate steps. At each iteration, one variable is optimized while the other remains fixed. This strategy is detailed in Algorithm 1.

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

Algorithm 1 Plug-and-play algorithm

Input: $\boldsymbol{y}_1, \boldsymbol{y}_2, n_{\text{iter}}, D_\sigma, \alpha_x, \lambda_1, \lambda_2$
Output: $\boldsymbol{x}, \boldsymbol{T}$

$\boldsymbol{y}_1, \boldsymbol{y}_2 \leftarrow$ unit-sphere normalisation$(\boldsymbol{y}_1, \boldsymbol{y}_2)$
Initialization
$\boldsymbol{x}_1 \leftarrow \boldsymbol{y}_2$
$\boldsymbol{T} \leftarrow$ coarse registration$(\boldsymbol{y}_1, \boldsymbol{y}_2)$
for $k \in \{1, \dots, n_{\text{iter}}\}$ **do**
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \alpha_x \frac{\partial f(\boldsymbol{x}_k, \boldsymbol{T}_k)}{\partial \boldsymbol{x}_k} + \alpha_x \lambda_2 (D_\sigma(\boldsymbol{x_k}) - \boldsymbol{x_k})$
  $\boldsymbol{T}_{k+1} \leftarrow \boldsymbol{T}_k - \left(\boldsymbol{J}_{\boldsymbol{r}}^\top(\boldsymbol{T}_k)\boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{T}_k)\right)^{-1} \boldsymbol{J}_{\boldsymbol{r}}^\top(\boldsymbol{T}_k)\boldsymbol{r}(\boldsymbol{T}_k)$
  $k \leftarrow k + 1$
end for

The optimization of $\boldsymbol{x}$ in (6) for a fixed transformation $\boldsymbol{T}$ can be solved using gradient descent for the first term and inspired by Tweedie's identity as in (4) for the second term:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_x \frac{\partial f(\boldsymbol{x}_k, \boldsymbol{T}_k)}{\partial \boldsymbol{x}_k} + \alpha_x \lambda_2 [p(D_\sigma(\boldsymbol{x_k})) - \boldsymbol{x_k}],$$

where α_x is the stepsize for the update of $\boldsymbol{x}_k$[2] and that p permutes the order of points in the denoised cloud $D_\sigma(\boldsymbol{x}_k)$, finding the nearest neighbors to $\boldsymbol{x}_k$. This permutation is needed because the denoiser does not necessarily return the point cloud in the same order as the input. The plug-and-play method can be used with any point cloud denoiser D_σ. This paper concentrates on Score-Denoise [20], a 3D point cloud denoiser that uses a neural network to remove noise. To enforce the stability of the algorithm, the point cloud is randomly rotated before denoising, and the inverse rotation is applied to the denoised output [25].

In order to estimate $\boldsymbol{T}$ for a fixed $\boldsymbol{x}_k$, we propose to use the Gauss-Newton method. The Gauss-Newton algorithm uses a gradient descent where the optimal stepsize is chosen at each iteration [26]. The distance $d(\boldsymbol{y}_1, \boldsymbol{T}\boldsymbol{x})$ in (6) can be reformulated as $\sum_{i=1}^{N+M} \boldsymbol{r}_i^2(\boldsymbol{T})$ where:

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

where $k_i = \underset{j}{\text{argmin}} \|\boldsymbol{y}_{1i} - \boldsymbol{T}\boldsymbol{x}_j\|^2$ and $\ell_i = \underset{j}{\text{argmin}} \|\boldsymbol{y}_{1j} - \boldsymbol{T}\boldsymbol{x}_i\|^2$. Note that $\sum_{i=1}^{N} \boldsymbol{r}_i^2(\boldsymbol{T})$ is equivalent to the first term in the Chamfer distance in (7) and $\sum_{i=N+1}^{N+M} \boldsymbol{r}_i^2(\boldsymbol{T})$ to the second term. Since the only term in (6) that depends on $\boldsymbol{T}$ is $d(\boldsymbol{y}_1, \boldsymbol{T}\boldsymbol{x})$, the estimation of $\boldsymbol{T}$ for a fixed $\boldsymbol{x}_k$ can be determined with the Gauss-Newton algorithm as follows:

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

where $\boldsymbol{r}(\boldsymbol{T}) = \begin{bmatrix} \boldsymbol{r}_1^\top(\boldsymbol{T}) & \dots & \boldsymbol{r}_{N+M}^\top(\boldsymbol{T}) \end{bmatrix}^\top \in \mathbb{R}^{4(N+M)}$, $\boldsymbol{J}_{\boldsymbol{r}}$ is the Jacobian matrix of $\boldsymbol{r}$ and $\boldsymbol{T}$ is initialized with the identity matrix.

2.4. Hyperparameter tuning

The proposed method requires to adjust the following hyperparameters: λ_1 that balances the weight between the two point clouds, λ_2 for the regularization term, α_x the stepsize of the gradient descent for $\boldsymbol{x}$ and n_{iter} the number of iterations. The number of iterations is fixed to simplify hyperparameter tuning. Optuna [27] is an open-source framework for hyperparameter optimization that uses a define-by-run approach, allowing the search space to be dynamically constructed during execution rather than predefined to a fixed value. It is based on the tree-structured Parzen estimator, a Bayesian optimization algorithm that identifies the optimal hyperparameter spaces. However, this approach has shown some limitations since the hyperparameters are correlated with the level of noise and we still need to find a maximum and minimum value of each hyperparameter. The problem was simplified by estimating α_x and λ_2 while fixing $\lambda_1 = 1$ with a transformation $\boldsymbol{T}$ equal to the identity. The value of λ_2 obviously depends on the value of α_x. Thus, the first test was done without the data fidelity term, because in this case, the only remaining hyperparameter is $\alpha_x\lambda_2$. The best value for $\alpha_x\lambda_2$ was found by cross validation leading to $\alpha_x\lambda_2 = 0.1$ for $n_{\text{iter}} = 100$. Using this information, all the hyperparameters were searched in the following intervals (for $n_{\text{iter}} = 100$): $0 < \alpha_x \leq 500$, $0 < \lambda_1 \leq 100$ and $\lambda_2 = 0.1/\alpha_x$. The optimal values for the hyperparameters of the experiments in the following section, determined manually to provide the best results, are summarized in Table 1.

3. EXPERIMENTS

The proposed point cloud registration method is evaluated using LiDAR data available on the website of the National Institute of Geographic and Forest Information (IGN[3]). A rigid transformation, composed of a rotation and a translation, was applied to a reference LiDAR dataset associated with the stadium of Marseille in France, which was down-sampled to provide the unknown point cloud $\boldsymbol{x}$ of size 30000 points. Additive white Gaussian noises were added to the reference and transformed point clouds to provide the sets $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ (displayed in Fig. 2 (a)). Before running the algorithm, the point clouds $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ were normalized using a unit sphere normalization, which is commonly considered when using denoisers based on neural networks [20]. Note that for all the experiments, the two point clouds $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ have the same level of noise, i.e., $\sigma_{y_1} = \sigma_{y_2} \triangleq \sigma_y$.

Table 1. Hyperparameter values used in the experiments.

Hyperparameters	n_{iter}	$\alpha_{\boldsymbol{x}}$	λ_1	λ_2
Denoising with $\sigma_y = 0.02$	50	100	1	0.0008
Registration with $\sigma_y = 0.02$	100	0.1	46	0.3
Registration with $\sigma_y = 0.05$	200	0.1	46	0.1

3.1. Denoising

Even though the aim of the proposed method is to solve 3D registration problems, we can also use it to solve other inverse problems. It was first used for point cloud denoising to evaluate its performance for a simpler problem. The transformation matrix was set to the identity matrix and its update was frozen for this example. With this setup, the plug-and-play algorithm reduces to a denoiser with two noisy versions of the same point cloud. For this experiment, we added two Gaussian noises with the same standard deviation $\sigma_y = 0.02$. The Chamfer distance was used to evaluate the performance of denoising. This distance was computed between the

[2]Note that the minimum function and thus the Chamfer distance are not differentiable everywhere. However, the gradients can be replaced by subgradients (that can be computed analytically or with pytorch.autograd handling non-differentiability automatically)

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

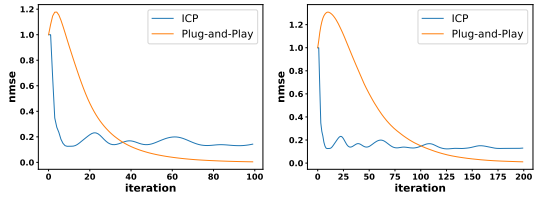
Table 2. PSNR comparison for the denoised point clouds

Initial data	Score-denoising	Plug-and-play
33.01 dB	38.23 dB	**39.35 dB**

ground-truth and one of the noisy point clouds in order to have a reference and then it was computed between the ground-truth and the estimated point cloud. After unit-sphere normalization, the value of Chamfer distance was on the order of 10^{-4} for the initial noisy data. The PSNR defined as $-10\log(\text{err})$ where err represents the Chamfer distance was then used for performance evaluation. The results obtained with a single application of the denoiser network [20] and the proposed denoiser were then compared. As shown in Table 2, a gain of 1dB is obtained with the plug-and-play approach.

3.2. Registration

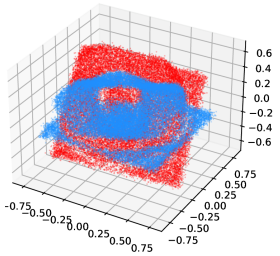
The registration method was also evaluated after applying a known transformation matrix to the LiDAR point cloud. In order to do so, a rotation was created with Euler angles, with angles for the x-axis, y-axis and z-axis equal to 40, 10 and 20, respectively. As for the translation vector, it was fixed to $\boldsymbol{t} = \begin{bmatrix}3 & 2 & 7\end{bmatrix}^\top$ without loss of generality. Gaussian noises with standard deviation $\sigma_y = 0.02$ were added to both point clouds. In this experiment with $n_{\text{iter}} = 100$, we found that promising results were obtained with $\alpha_x = 0.1$, $\lambda_1 = 46$ and $\lambda_2 = 0.3$. The denoised point cloud is displayed in blue in Fig. 2 (d). It can be compared to the ground-truth shown in green in Fig. 2 (e), and the noisy point clouds registered with the plug-and-play method (Fig. 2 (b)) and ICP (Fig. 2 (c)) with the same number of iterations. As shown in Fig. 2, the estimated point cloud seems to be well denoised and similar to the ground-truth. The PSNR between the noisy point cloud and the ground-truth is 33.01 dB and goes up to 39.87 dB after 100 iterations. Note that the point clouds are better registered with the plug-and-play method than with ICP, as shown in Table 3 with the PSNR values between $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ before and after registration. Note that even though ICP seems to work better than our method without noise, it is important to mention that 132 dB represents a distance of 5×10^{-14} which is very low. This shows that the proposed approach can register even with noisy data and denoising iteratively while computing the transformation matrix can improve the performance of the registration.



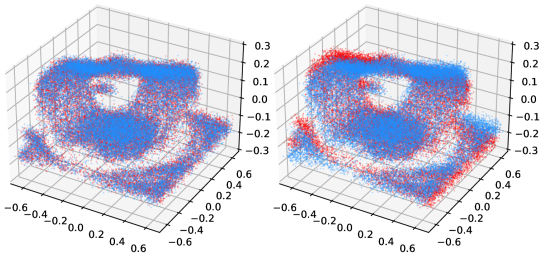
(a) $\sigma_y = 0.02$. (b) $\sigma_y = 0.05$.

Fig. 1. Evolutions of the NMSE between the ground-truth and the estimated transformation matrix.

To measure the difference between the estimated transformation matrix and the ground-truth, the normalized mean square error (NMSE) is computed between these two matrices. The evolution of the NMSE of the two methods through the iterations is shown in Fig. 1 for two noise levels ($\sigma_y = 0.02$ and $\sigma_y = 0.05$). The plug-and-play approach provides estimates with smaller NMSEs than with ICP.

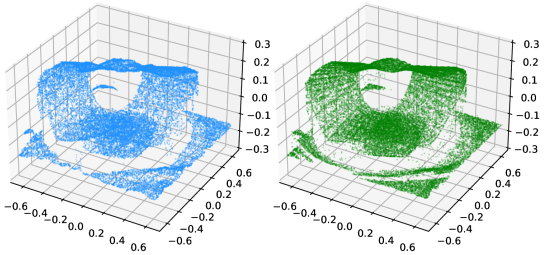


(a) Initial point clouds after normalization ($\boldsymbol{y}_1$ in red and $\boldsymbol{y}_2$ in blue).



(b) Registered point clouds with plug-and-play ($\boldsymbol{y}_1$ with the inverse transformation applied in red and $\boldsymbol{y}_2$ in blue).

(c) Registered point clouds with ICP ($\boldsymbol{y}_1$ with the inverse transformation applied in red and $\boldsymbol{y}_2$ in blue).



(d) Estimated point cloud ($\boldsymbol{x}$ in blue).

(e) Ground-truth ($\boldsymbol{x}_{gt}$ in green).

Fig. 2. Plots of Marseille's stadium.

Table 3. PSNR comparison between the initial point clouds y_1 and y_2 when a transformation $\boldsymbol{T}$ has been applied to the LiDAR data.

Method	Rot. + trans.	Rot. + trans. + noise
Initial data	11.09 dB	12.89 dB
Plug-and-play	132.00 dB	**35.36 dB**
ICP	**247.52 dB**	34.90 dB

4. CONCLUSION

This paper proposed a new plug-and-play algorithm for 3D point cloud registration. Experiments show competitive results when compared to the standard ICP algorithm for noisy point clouds. An interesting property of the plug-and-play method is that it can be used for joint registration and denoising of 3D point clouds. Future work includes extending the method to handle outliers and partially overlapping point clouds. Application of this approach to point clouds resulting from Pléaides-NEO, Pléiades and CO3D images or to point clouds with different resolutions is also an interesting prospect.

5. REFERENCES

[1] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 50–61, 2020.

[2] C. McGlone, *Manual of photogrammetry*. American Society of Photogrammetry and Remote Sensing, 1980.

[3] D. Youssefi, J. Michel, E. Sarrazin, F. Buffe, M. Cournet, J.-M. Delvit, C. L'Helguen, O. Melet, A. Emilien, and J. Bosman, "CARS: A photogrammetry pipeline using dask graphs to construct a global 3D model," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, pp. 453–456, 2020.

[4] Z. Ma and S. Liu, "A review of 3D reconstruction techniques in civil engineering and their applications," *Adv. Eng. Inform.*, vol. 37, pp. 163–174, 2018.

[5] 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.

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

[7] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2241–2254, 2015.

[8] A. Kolpakov and M. Werman, "An approach to robust ICP initialization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12685–12691, 2023.

[9] 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.

[10] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. IEEE Int. Conf. on 3D Imaging, Modeling, Process., Vis. & Transm. (3DIMPVT)*, (Quebec City, QC, Canada), pp. 145–152, 2001.

[11] 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.

[12] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, pp. 674–679, 1981.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, (Long Beach, California, United States), 2017.

[14] 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.

[15] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, (Miami, Florida, United States), pp. 373–380, 2009.

[16] A. Mian, M. Bennamoun, and R. Owens, "On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes," *Int. J. Comput. Vis.*, vol. 89, pp. 348–361, 2010.

[17] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imaging Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[18] 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.

[19] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "Pointcleannet: Learning to denoise and remove outliers from dense point clouds," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 185–203, 2020.

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

[21] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proc. IEEE Int. Conf. Comput. Vis.*, (Seoul, Korea), pp. 52–60, 2019.

[22] 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.

[23] B. Efron, "Tweedie's formula and selection bias," *J. Am. Stat. Assoc.*, vol. 106, no. 496, pp. 1602–1614, 2011.

[24] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Density-aware Chamfer Distance as a Comprehensive Metric for Point Cloud Completion," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 29088–29100, 2021.

[25] M. Terris, T. Moreau, N. Pustelnik, and J. Tachella, "Equivariant Plug-and-Play Image Reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 25255–25264, June 2024.

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

[27] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proc. ACM Int. Conf. Knowl. Discov. Data Min. (SIGKDD)*, (Anchorage, AK, USA), pp. 2623–2631, 2019.