PosDiffNet: Positional Neural Diffusion for Point Cloud Registration in a Large Field of View with Perturbations

Rui She*, Sijie Wang*, Qiyu Kang*†, Kai Zhao, Yang Song,
Wee Peng Tay, Tianyu Geng, Xingchao Jian

Nanyang Technological University, Singapore
{rui.she@, wang1679@e., qiyu.kang@, kai.zhao@, wptay@, tianyu.geng@, xingchao001@e.}ntu.edu.sg,
yang.song@connect.polyu.hk

Abstract

Point cloud registration is a crucial technique in 3D computer vision with a wide range of applications. However, this task can be challenging, particularly in large fields of view with dynamic objects, environmental noise, or other perturbations. To address this challenge, we propose a model called *PosDiffNet*. Our approach performs hierarchical registration based on window-level, patch-level, and point-level correspondence. We leverage a graph neural partial differential equation (PDE) based on Beltrami flow to obtain high-dimensional features and position embeddings for point clouds. We incorporate position embeddings into a Transformer module based on a neural ordinary differential equation (ODE) to efficiently represent patches within points. We employ the multi-level correspondence derived from the high feature similarity scores to facilitate alignment between point clouds. Subsequently, we use registration methods such as SVD-based algorithms to predict the transformation using corresponding point pairs. We evaluate PosDiffNet on several 3D point cloud datasets, verifying that it achieves state-of-the-art (SOTA) performance for point cloud registration in large fields of view with perturbations. The implementation code of experiments is available at https://github.com/AI-IT-AVs/PosDiffNet.

Introduction

Three-dimensional (3D) computer vision techniques recently have gained increasing popularity in various fields such as autonomous driving (Wang et al. 2023b), robotics (Li et al. 2021), and scene modeling (Kang et al. 2022). Point cloud registration, which estimates the transformation or relative pose between two given 3D point cloud frames (Wang and Solomon 2019), is a crucial task in many applications, such as object detection, odometry estimation, as well as simultaneous localization and mapping (SLAM) (Shan et al. 2020; Kang et al. 2022), owing to its robustness against seasonal changes and illumination variations.

Iterative methods, as demonstrated by the iterative closest point (ICP) algorithm (Besl and McKay 1992; Segal, Haehnel, and Thrun 2009), have become widely employed in point cloud registration. Despite their utility, these methods face obstacles. Specifically, the non-convexity of the optimization problem poses a significant challenge to the attainment of a globally optimal solution (Wang and Solomon 2019). When dealing with sparse and non-uniform data, traditional methods like nearest-neighbor search may not be effective, resulting in higher registration errors (Wang and Solomon 2019; Wei et al. 2020).

To address the aforementioned challenges in point cloud registration, deep learning-based methods have been investigated to predict transformation matrices or relative poses (Choy, Park, and Koltun 2019; Bai et al. 2020; Ao et al. 2021). However, achieving robust point cloud registration in large-scale scenarios remains a significant challenge due to LiDAR scan distortion and sparsity. For instance, real outdoor datasets often exhibit numerous perturbations among different frames, such as dynamic objects and environmental noise (Yu et al. 2019). Thus, an open question is how to efficiently estimate the transformation for large-scale scenarios with perturbations, especially in real outdoor datasets.

In this paper, we propose a model for point cloud registration based on neural diffusion. Considering the demonstrated capability of Beltrami flow in preserving non-smooth graph signals and its robustness in feature representation (Song et al. 2022), we utilize feature descriptors and position embeddings based on graph neural diffusion with Beltrami flow (Kimmel, Sochen, and Malladi 1997; Chamberlain et al. 2021a). We also present a transformation estimation method using a diffusion-based Transformer. Our approach mitigates the challenges from dynamic object non-correspondence and random perturbations in large fields of view, leading to robust and efficient point cloud registration. Our main contributions are as follows:

- We design a 3D point cloud representation module using graph neural diffusion based on Beltrami flow, from which point feature embedding and position embedding are both outputs.
- We propose a point cloud registration method based on the window-patch-point matching and a Transformer, which incorporates neural ODE modules and leverages point features and their positional information.
- We empirically evaluate our point cloud registration method to outperform other baselines in several real datasets in the large field of view with perturbations.

*These authors contributed equally.
†Corresponding author: Qiyu Kang.

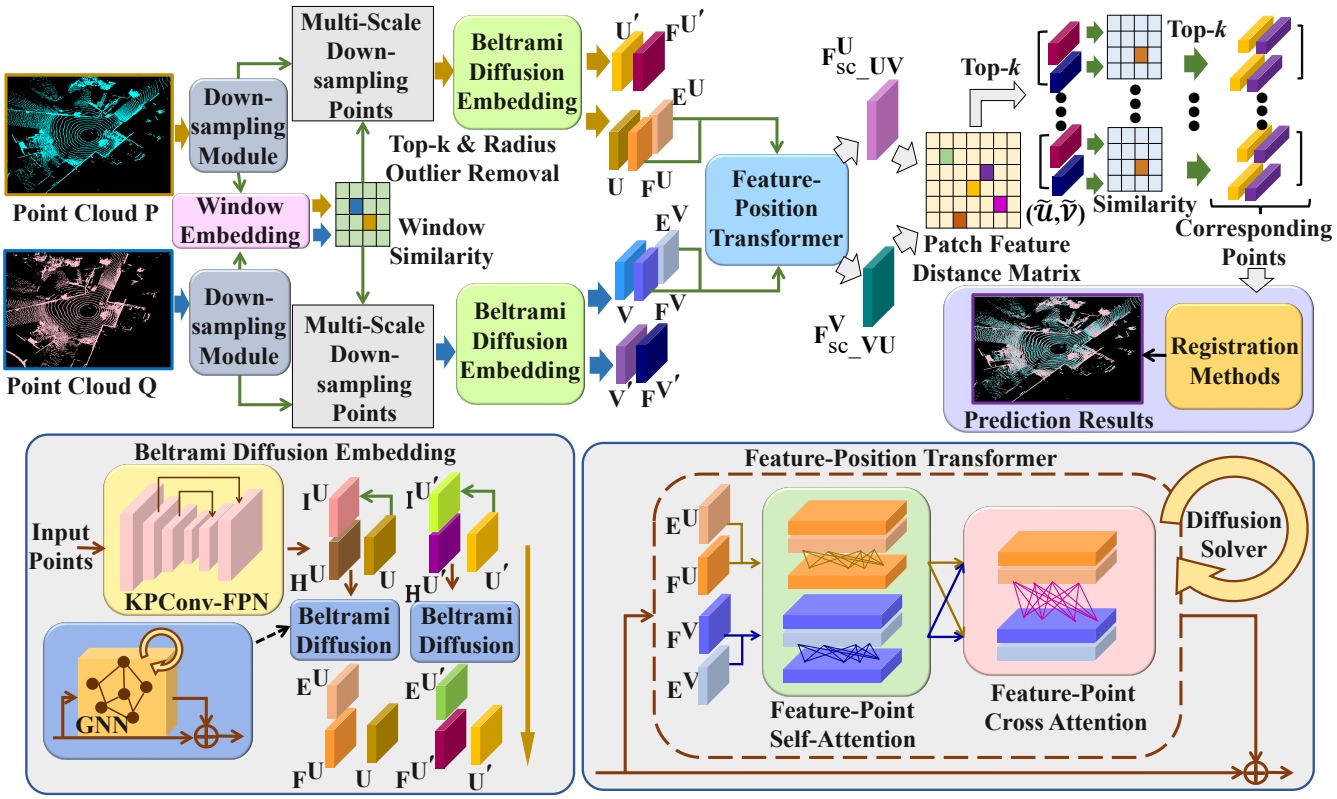

Figure 1: The architecture of our PosDiffNet for the registration task with respect to (w.r.t.) point cloud pairs. Detailed information about the modules can be found in the subsequent subsections of Methodology.

Related Work

Point Cloud Registration Methods. As classical approaches, the ICP (Besl and McKay 1992) and the random sample consensus (RANSAC) (Fischler and Bolles 1981) are widely used for point cloud registration. RANSAC requires more computing resources and higher running time complexity due to its low convergence. ICP's performance mainly depends on the selection of the initial value. A series of methods to refine ICP have been proposed (Guérout et al. 2017; Koide et al. 2021) to improve the acceleration and accuracy. Correspondence-based estimators are used for point cloud registration. One type of method performs repeatable keypoint detection (Bai et al. 2020; Huang et al. 2021) and then learns the keypoint descriptor for the correspondence acquisition (Choy, Park, and Koltun 2019; Ao et al. 2021) or similarity measures to obtain the correspondences (Quan and Yang 2020; Chen et al. 2022), such as D3Feat (Bai et al. 2020), SpinNet (Ao et al. 2021), PREDATOR (Huang et al. 2021) and SC^2-PCR (Chen et al. 2022). The other, such as deep closest point (DCP) (Wang and Solomon 2019), CoFiNet (Yu et al. 2021) and UDPReg (Mei et al. 2023), performs the correspondence retrieval for all possible matching point pairs without the keypoint detection. Additionally, auxiliary modules can be integrated into learning-based estimators, such as SuperLine3D (Zhao et al. 2022) and Maximal Cliques (MAC) (Zhang et al. 2023).

In order to achieve more robust non-handcrafted estimators, learning-based methods are introduced into the transformation prediction (Qin et al. 2022). Since conventional estimators like RANSAC have drawbacks in terms of convergence speed and are unstable in the presence of numerous outliers, learning-based estimators (Lu et al. 2021; Poiesi and Boscaini 2022; Pais et al. 2020), such as StickyPillars (Fischer et al. 2021), PointDSC (Bai et al. 2021), EDFNet (Zhang et al. 2022), GeoTransformer (GeoTrans) (Qin et al. 2022), Lepard (Li and Harada 2022), BUFFER (Ao et al. 2023), RoITr (Yu et al. 2023) and RoReg (Wang et al. 2023a), have attracted much interest.

Point Cloud Feature Representation. In general, there are three categories of 3D feature representation methods. In the first category, voxel alignment is initially performed on the points, followed by the extraction of corresponding features through a 3D convolutional neural network (CNN) (Sindagi, Zhou, and Tuzel 2019; Kopuklu et al. 2019; Kumawat and Raman 2019). However, it is worth noting that this approach exhibits a long running time. The second category focuses on the reduction of a 3D point cloud into a 2D map, subsequently leveraging classical 2D CNN techniques for feature extraction (Su et al. 2015). Nevertheless, this approach may introduce unforeseen noise artifacts, which can impact the quality of the extracted features. The third category is to extract features from the raw point clouds directly using specific neural networks, such as PointNet (Qi et al. 2017), dynamic graph convolutional neural networks (DGCNN) (Wang

et al. 2019), point cloud transformer (PCT) (Guo et al. 2021), GdDi (Poiesi and Boscaini 2022), PointMLP (Ma et al. 2022), and PointNeXt (Qian et al. 2022).

Beltrami Neural Diffusion. Beltrami flow is a partial differential equation widely used in signal processing (Kimmel, Sochen, and Malladi 1997; Chamberlain et al. 2021a; Zhao et al. 2023). A Beltrami diffusion on the graph is defined (Song et al. 2022) as

$$\frac{\partial \mathbf{Z}(\mu,t)}{\partial t} = \frac{1}{2}\frac{1}{\|\nabla \mathbf{Z}\|}\mathrm{div}\left(\frac{\nabla \mathbf{Z}}{\|\nabla \mathbf{Z}\|}\right)(\mu,t), \tag{1}$$

where div denotes the divergence, ∇ denotes the gradient operator, $\|\cdot\|$ is a norm operator, vertex feature $\mathbf{Z}(\cdot,t)$ satisfies $\mathbf{Z}(\mu,t) = (\mathbf{X}(\mu,t), \mathbf{Y}(\mu,t))$ and μ is the index of vertices, $(\mathbf{X}(\mu,t), \mathbf{Y}(\mu,t))$ denotes a pair of vertex features and positional features at the vertex with the index μ. To combine the Beltrami flow and graph neural diffusion, a Beltrami neural diffusion (Chamberlain et al. 2021a) is presented as

$$\left[\frac{\mathrm{d}\mathbf{X}(t)}{\mathrm{d}t}, \frac{\mathrm{d}\mathbf{Y}(t)}{\mathrm{d}t}\right] = (\mathbf{A}_{\mathbf{B}}(\mathbf{X}(t), \mathbf{Y}(t)) - \mathbf{I})[\mathbf{X}(t), \mathbf{Y}(t)], \tag{2}$$

$$\mathbf{X}(0) = \mathbf{X}; \mathbf{Y}(0) = \alpha\mathbf{Y}; t \geq 0, \tag{3}$$

where $\mathbf{X}$ and $\mathbf{Y}$ denote the vertex feature and positional feature in a graph, respectively. $\alpha > 0$ is a scaling factor and $\mathbf{A}_{\mathbf{B}}(\cdot,\cdot)$ is the learnable matrix-valued function. From (Chamberlain et al. 2021b; She et al. 2023), most graph neural networks (GNNs) can be regarded as partial differential diffusions using different discretization, which leads that (2) can be viewed as a neural partial differential equation (PDE).

As an advantage of neural diffusions with Beltrami flow, the robustness of feature representation for vertices is improved using both vertex features and positional features (Song et al. 2022; Chamberlain et al. 2021a). From (1), since there exists a term of $\frac{1}{\|\nabla \mathbf{Z}\|}$ when the gradient is large, the feature updates slowly. This benefits the shape description for the structure of vertices (Song et al. 2022). Due to the advantages of Beltrami diffusion, this process can smooth out the noise and enhance the shape features of the input.

Methodology

Problem Formulation

Consider two point clouds, $\mathbf{P} = \{P_i\}$ and $\mathbf{Q} = \{Q_j\}$, which are subsets of $\mathbb{R}^3$. We first employ a neural-diffusion-based mapping function f to embed each point (or a subset of points) into a d-dimensional feature space, $\mathbb{R}^d$. The intention behind this process is to leverage the similarities between the embeddings derived from the two point clouds for identifying matched or corresponding points. Subsequently, we anticipate predicting the rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{t}}$ that correspond to the ground-truth rotation $\mathbf{R}$ and translation $\mathbf{t}$. For the point cloud registration task, our objective function is naturally defined as: $\hat{\mathbf{R}}^*, \hat{\mathbf{t}}^* = \arg\min_{\hat{\mathbf{R}},\hat{\mathbf{t}}} \ell_{\mathrm{D}}((\hat{\mathbf{R}}, \hat{\mathbf{t}}), (\mathbf{R}, \mathbf{t}))$, where ℓ_{D} represents a metric. Alternatively, this can also be represented as

$$\hat{\mathbf{R}}^*, \hat{\mathbf{t}}^* = \arg\min_{\hat{\mathbf{R}},\hat{\mathbf{t}}} \ell_{\mathrm{loss}}(\pi(\mathbf{P}^{\mathrm{co}}, (\hat{\mathbf{R}}, \hat{\mathbf{t}})), \mathbf{Q}^{\mathrm{co}}). \tag{4}$$

In this context, the point-level matching $(\mathbf{P}^{\mathrm{co}}, \mathbf{Q}^{\mathrm{co}}) = (\{P_l\}_{l\in\mathcal{L}}, \{Q_l\}_{l\in\mathcal{L}})$ is established, where P_l corresponds to Q_l, and $\mathcal{L}$ is the index set of corresponding points. $\pi(\cdot,\cdot)$ denotes the transformation operation. ℓ_{loss} is a loss function such as mean squared error (MSE). To solve the proposed problem, we design a novel model, integrating neural diffusion. The architecture of this model is illustrated in Fig. 1.

In Fig. 1, the down-sampling module with multi-scale voxel sizes is the same as that in (Yu et al. 2021) for obtaining window-level and patch-level central points, denoted as $(\mathbf{U}_{\mathrm{win}}, \mathbf{V}_{\mathrm{win}})$ and $(\mathbf{U}, \mathbf{V})$ respectively. Every window encompasses patches whose central points are within it, with each patch encapsulating points within the same process. The window feature module consists of the DGCNN from (Wang et al. 2019) and the Transformer from (Wang and Solomon 2019). The Top-K and radius outlier removal methods are applied to filter out outlier windows within patches and points. Then, we use the remaining patches and points for further registration.

Point Cloud Representation with Beltrami Diffusion

To represent point clouds, we initially extract both point-level and patch-level features utilizing the KPConv-FPN method (Qin et al. 2022; Thomas et al. 2019). The two feature representations correspond to the downsampled points and patch central points. Then, we introduce the feature and position embeddings on the points and the patch central points.

Given a pair of original point clouds, *(i)* we represent these clouds by their patch central points $(\mathbf{U}, \mathbf{V})$, where $\mathbf{U} \in \mathbb{R}^{|\mathbf{U}|\times 3}$ and $\mathbf{V} \in \mathbb{R}^{|\mathbf{V}|\times 3}$. Each patch central point is denoted as $\mathbf{u}_i$ and $\mathbf{v}_j$, respectively. The learned patch-level features and position embeddings are represented by $([\mathbf{H}^{\mathbf{U}}, \mathbf{I}^{\mathbf{U}}], [\mathbf{H}^{\mathbf{V}}, \mathbf{I}^{\mathbf{V}}])$, where $\mathbf{H}^{\mathbf{U}}, \mathbf{I}^{\mathbf{U}} \in \mathbb{R}^{|\mathbf{U}|\times d}$ and $\mathbf{H}^{\mathbf{V}}, \mathbf{I}^{\mathbf{V}} \in \mathbb{R}^{|\mathbf{V}|\times d}$. Similarly, *(ii)* a pair of point clouds are denoted as $(\mathbf{U}', \mathbf{V}')$. The learned point-level features and position embeddings are represented by $([\mathbf{H}^{\mathbf{U}'}, \mathbf{I}^{\mathbf{U}'}], [\mathbf{H}^{\mathbf{V}'}, \mathbf{I}^{\mathbf{V}'}])$, where $\mathbf{H}^{\mathbf{U}'}, \mathbf{I}^{\mathbf{U}'} \in \mathbb{R}^{|\mathbf{U}'|\times d'}$ and $\mathbf{H}^{\mathbf{V}'}, \mathbf{I}^{\mathbf{V}'} \in \mathbb{R}^{|\mathbf{V}'|\times d'}$. *(iii)* Each patch central point can be associated with its patch consisting of points using a grouping strategy (Yu et al. 2021; Qin et al. 2022; Li, Chen, and Lee 2018). The corresponding patch sets based on $(\mathbf{U}, \mathbf{V})$ are denoted by $\mathcal{U} = \{\mathcal{U}_i | i = 1, \ldots, |\mathbf{U}|\}$ and $\mathcal{V} = \{\mathcal{V}_j | j = 1, \ldots, |\mathbf{V}|\}$, where $\mathcal{U}_i = \{\mathbf{u}'_\eta | \mathbf{u}'_\eta \in \mathbf{U}', \|\mathbf{u}'_\eta - \mathbf{u}_i\| < \Gamma, \eta = 1, \ldots, |\mathbf{U}'|\}$ and $\mathcal{V}_j = \{\mathbf{v}'_\xi | \mathbf{v}'_\xi \in \mathbf{V}', \|\mathbf{v}'_\xi - \mathbf{v}_j\| < \Gamma, \xi = 1, \ldots, |\mathbf{V}'|\}$, and Γ is a threshold parameter.

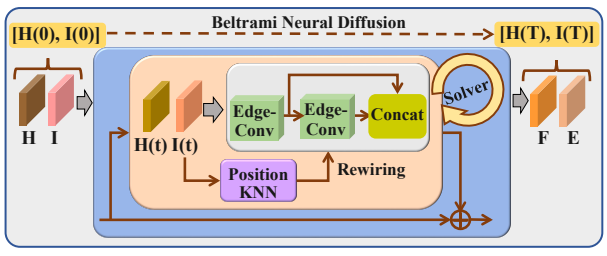

Figure 2: Architecture of the Beltrami neural diffusion module for feature and position embeddings.

To achieve enhanced robustness in the embeddings of both features and positions, we employ a neural diffusion mechanism rooted in Beltrami flow. This mechanism is specifically applied to each feature and position pair denoted as $[\mathbf{H}, \mathbf{I}]$. Taking $[\mathbf{H}^{\mathbf{U}}, \mathbf{I}^{\mathbf{U}}]$ as an instance, the Beltrami neural diffusion module is characterized by

$$\left[\frac{\mathrm{d}\mathbf{H}^{\mathbf{U}}(t)}{\mathrm{d}t}, \frac{\mathrm{d}\mathbf{I}^{\mathbf{U}}(t)}{\mathrm{d}t}\right] = f_{\mathrm{BND}}([\mathbf{H}^{\mathbf{U}}(t), \mathbf{I}^{\mathbf{U}}(t)]), \quad (5)$$

where $f_{\mathrm{BND}}([\mathbf{H}^{\mathbf{U}}(t), \mathbf{I}^{\mathbf{U}}(t)]) \in \mathbb{R}^{|\mathbf{U}|\times 2d}$, $t \in [0, T_f]$, and $f_{\mathrm{BND}}(\cdot)$ denotes a Graph Neural Network (e.g. DGCNN (Wang et al. 2019)). This mapping is employed to embed both the point features and positions. In the context of updating the diffusion state, the construction of the neighborhood graph is derived from the k nearest neighbors, based on $\mathbf{I}^{\mathbf{U}}(t)$ at time t. The metric used to search the nearest neighbors is the $\mathcal{L}_2$ distance in the Euclidean space of point positions. This graph construction facilitates the effective integration of information during the diffusion process.

By integrating the (5) from $t = 0$ to $t = T_f$, we obtain the embeddings for the $[\mathbf{H}^{\mathbf{U}}, \mathbf{I}^{\mathbf{U}}]$ given by

$$[\mathbf{F}^{\mathbf{U}}, \mathbf{E}^{\mathbf{U}}] = [\mathbf{H}^{\mathbf{U}}(T_f), \mathbf{I}^{\mathbf{U}}(T_f)] = \mathcal{F}_{\mathrm{BND}}([\mathbf{H}^{\mathbf{U}}, \mathbf{I}^{\mathbf{U}}]), \quad (6)$$

where $\mathcal{F}_{\mathrm{BND}}(\cdot)$ indicates the mapping for the Beltrami neural diffusion by solving (5), and $\mathbf{F}^{\mathbf{U}}, \mathbf{E}^{\mathbf{U}} \in \mathbb{R}^{|\mathbf{U}|\times d}$. Analogously, the embeddings corresponding to the point features and their respective positions, obtained through the Beltrami neural diffusion, are represented by $(\mathbf{F}^{\mathbf{U}'}, \mathbf{F}^{\mathbf{V}'})$. The architecture of the Beltrami neural diffusion module is shown in Fig. 2.

Feature-Position Transformer with Neural Diffusion

We propose a Transformer module based on neural ODE and the point and position embeddings derived from the Beltrami neural diffusion. For a pair of point clouds $(\mathbf{U}, \mathbf{V})$, the input point features and position embeddings utilized as inputs for the Transformer module are represented by $[\mathbf{F}^{\mathbf{U}}, \mathbf{E}^{\mathbf{U}}]$ and $[\mathbf{F}^{\mathbf{V}}, \mathbf{E}^{\mathbf{V}}]$, respectively. Leveraging these embeddings, we proceed to elaborate on the self-attention and cross-attention mechanisms within the Transformer module.

Feature-Position Self-Attention Mechanism. To emphasize the geometric position of each point and augment the richness of point representation, we integrate position embeddings into the self-attention module. For a given point cloud $\mathbf{U}$, we input the normalized versions of $\mathbf{F}^{\mathbf{U}}$ and $\mathbf{E}^{\mathbf{U}}$ into the self-attention module. As a result, we obtain the embedding generated by the feature-position self-attention module as follows

$$f_{\mathrm{s\_att}}(\mathbf{F}^{\mathbf{U}}) = \Big\|_{i=1}^{S_{\mathrm{head}}} \left(f_{\mathrm{sfx}}\left(\frac{(\mathbf{F}^{\mathbf{U}}\mathbf{W}_i^{\mathbf{sq}})(\mathbf{F}^{\mathbf{U}}\mathbf{W}_i^{\mathbf{sk}})^{\intercal}}{\sqrt{d_i^{\mathrm{s}}}} + \frac{(\mathbf{F}^{\mathbf{U}}\mathbf{W}_i^{\mathbf{seq}})(\mathbf{E}^{\mathbf{U}}\mathbf{W}_i^{\mathbf{sek}})^{\intercal}}{\sqrt{d_i^{\mathrm{e}}}} \right) (\mathbf{F}^{\mathbf{U}}\mathbf{W}_i^{\mathbf{sv}}) \right) \mathbf{W}^{\mathbf{s}}, \quad (7)$$

where $\mathbf{W}_i^{\mathbf{sq}}$, $\mathbf{W}_i^{\mathbf{sk}}$, $\mathbf{W}_i^{\mathbf{sv}}$, $\mathbf{W}_i^{\mathbf{seq}}$, $\mathbf{W}_i^{\mathbf{sek}}$, and $\mathbf{W}^{\mathbf{s}}$ are all learnable neural networks for feature embedding. d_i^{s} and d_i^{e} denote the number of dimensions for point cloud features and position embeddings in the i-th attention head. $(\cdot)^{\intercal}$ and $\|$ are the transpose operation and the concatenation operation respectively. S_{head} denotes the number of heads. $f_{\mathrm{sfx}}(\cdot)$ is the row-wise softmax normalization function.

Furthermore, we employ the neural network module mentioned in standard Transformer architecture, including linear layers, feed forward networks (FFN), and normalization layers (Vaswani et al. 2017), as an embedding for $f_{\mathrm{s\_att}}(\mathbf{F}^{\mathbf{U}})$ to obtain $f_{\mathrm{s\_ate}}(\mathbf{F}^{\mathbf{U}})$.

Feature-Position Cross-Attention Mechanism. Based on the embeddings from the aforementioned self-attention and position information of points, we design a cross-attention for $\mathbf{U}$ and $\mathbf{V}$. When inputting the normalized $f_{\mathrm{s\_att}}(\mathbf{F}^{\mathbf{U}})$ w.r.t. $\mathbf{U}$ and $f_{\mathrm{s\_ate}}(\mathbf{F}^{\mathbf{V}})$ w.r.t. $\mathbf{V}$ into the cross-attention module, we have the corresponding embedding given by

$$f_{\mathrm{c\_att}}^{\mathbf{UV}}(\mathbf{F}^{\mathbf{U}}) = \Big\|_{j=1}^{C_{\mathrm{head}}} \left(f_{\mathrm{sfx}}\left(\frac{(f_{\mathrm{s\_att}}(\mathbf{F}^{\mathbf{U}})\mathbf{W}_j^{\mathbf{cq}})(f_{\mathrm{s\_ate}}(\mathbf{F}^{\mathbf{V}})\mathbf{W}_j^{\mathbf{ck}})^{\intercal}}{\sqrt{d_j^{\mathrm{c}}}} + \frac{(\mathbf{E}^{\mathbf{U}}\mathbf{W}_j^{\mathbf{ceq}})(\mathbf{E}^{\mathbf{V}}\mathbf{W}_j^{\mathbf{cek}})^{\intercal}}{\sqrt{d_j^{\mathrm{e}}}} \right) (f_{\mathrm{s\_ate}}(\mathbf{F}^{\mathbf{V}})\mathbf{W}_j^{\mathbf{cv}}) \right) \mathbf{W}^{\mathbf{c}}, \quad (8)$$

where the notations are similar to those in (7).

Then, we combine $f_{\mathrm{s\_att}}(\mathbf{F}^{\mathbf{U}})$ and $f_{\mathrm{c\_att}}^{\mathbf{UV}}(\mathbf{F}^{\mathbf{U}})$ to obtain the point feature embedding for $\mathbf{U}$, which is denoted by $f_{\mathrm{sc}}^{\mathbf{UV}}(\mathbf{F}^{\mathbf{U}})$. Meantime, we use fully connected (FC) layers to obtain the embedding of $\mathbf{E}^{\mathbf{U}}$ denoted by $f_{\mathrm{fc}}(\mathbf{E}^{\mathbf{U}})$. Furthermore, we introduce $[f_{\mathrm{sc}}^{\mathbf{UV}}(\cdot), f_{\mathrm{fc}}(\cdot)]$ into the neural ODE to achieve the neural-diffusion-based Transformer given by

$$\left[\frac{\mathrm{d}\mathbf{F}^{\mathbf{U}}(t)}{\mathrm{d}t}, \frac{\mathrm{d}\mathbf{E}^{\mathbf{U}}(t)}{\mathrm{d}t}\right] = \left[f_{\mathrm{sc}}^{\mathbf{UV}}(\mathbf{F}^{\mathbf{U}}(t)), f_{\mathrm{fc}}(\mathbf{E}^{\mathbf{U}}(t))\right], \quad (9)$$

where $\left[\mathbf{F}^{\mathbf{U}}(0), \mathbf{E}^{\mathbf{U}}(0)\right] = \left[\mathbf{F}^{\mathbf{U}}, \mathbf{E}^{\mathbf{U}}\right]$ and $t \geq 0$. Finally, we use the output of the neural ODE, that is, the solution integrated from time 0 to the terminal time T, as the embeddings $\left[\mathbf{F}_{\mathrm{sc\_UV}}^{\mathbf{U}}, \mathbf{E}_{\mathrm{sc\_UV}}^{\mathbf{U}}\right]$ for $\mathbf{U}$. Similarly, we also obtain $\left[\mathbf{F}_{\mathrm{sc\_VU}}^{\mathbf{V}}, \mathbf{E}_{\mathrm{sc\_VU}}^{\mathbf{V}}\right]$ for $\mathbf{V}$. These embeddings reflect the integrated information and dynamics captured by the neural ODE. The architecture of the Transformer with neural diffusion is shown in Fig. 3.

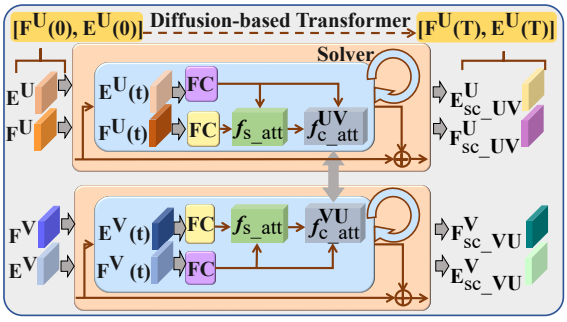

Figure 3: Architecture of the feature-position Transformer based on neural ODE.

Method	Testing on the Boreas (Sunny)				Testing on the Boreas (Night)				Testing on the KITTI			
	RTE (cm)		*RRE (°)*		*RTE (cm)*		*RRE (°)*		*RTE (cm)*		*RRE (°)*	
	MAE	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
ICP	11.97	33.99	0.14	0.35	10.83	18.28	0.11	0.21	9.86	19.48	0.17	0.27
DCP	14.59	25.39	0.16	0.34	11.63	17.36	0.12	0.21	19.96	31.44	0.25	0.45
HGNN++	13.81	23.63	0.16	0.34	14.41	23.16	0.14	0.25	10.38	19.69	0.19	0.30
VCR-Net	7.51	16.22	0.12	0.27	8.71	13.56	0.10	0.17	7.62	14.75	**0.16**	0.25
PCT++	9.92	19.19	0.14	0.30	9.81	15.77	0.10	0.19	9.40	17.60	0.17	0.27
GeoTrans	**3.11**	16.16	**0.08**	0.23	<u>4.58</u>	15.78	<u>0.08</u>	0.22	<u>6.19</u>	10.10	0.17	0.27
BUFFER	6.11	<u>7.49</u>	**0.08**	**0.12**	4.64	<u>6.22</u>	0.11	<u>0.13</u>	8.32	<u>9.71</u>	0.20	<u>0.26</u>
RoITr	7.66	13.05	0.10	0.18	9.37	13.68	0.09	0.14	7.50	11.94	0.20	0.31
PosDiffNet	<u>3.38</u>	**5.73**	**0.08**	<u>0.15</u>	**4.46**	**6.12**	**0.07**	**0.11**	**4.48**	**7.28**	**0.16**	**0.25**

Table 1: Point cloud registration performance using the Boreas dataset for training. The best and the second-best results are highlighted in bold and underlined, respectively.

Point Registration with Hierarchical Matching

Hierarchical matching. We conduct hierarchical matching for the corresponding windows, patches, and points. To match the corresponding windows and patches based on $(\mathbf{U}_{\text{win}}, \mathbf{V}_{\text{win}})$ and $(\mathbf{U}, \mathbf{V})$ respectively, we conduct the exponential feature distance matrices with dual normalization (Sun et al. 2021; Qin et al. 2022). For instance, we perform the patch-level matching on the patch central point pairs $(\mathbf{U}, \mathbf{V})$ corresponding to the point features $(\mathbf{F}^{\mathbf{U}}, \mathbf{F}^{\mathbf{V}})$. We have the dual-normalized feature distance correlation matrix $\mathbf{W}_{\mathbf{UV}} \in \mathbb{R}^{|\mathbf{U}|\times|\mathbf{V}|}$ where the element $w_{i,j}$ is given by

$$w_{i,j} = \frac{\exp(-2\|\mathbf{f}_i^{\mathbf{U}} - \mathbf{f}_j^{\mathbf{V}}\|_2^2)}{\sum_j \exp(-\|\mathbf{f}_i^{\mathbf{U}} - \mathbf{f}_j^{\mathbf{V}}\|_2^2) \sum_i \exp(-\|\mathbf{f}_i^{\mathbf{U}} - \mathbf{f}_j^{\mathbf{V}}\|_2^2)}, \tag{10}$$

where $\mathbf{f}_i^{\mathbf{U}}$ and $\mathbf{f}_j^{\mathbf{V}}$ are the elements of $\mathbf{F}^{\mathbf{U}}$ and $\mathbf{F}^{\mathbf{V}}$ respectively. Then, we use Top-K method to select N_{p} point pairs based on $w_{i,j}$, where the value of $w_{i,j}$ at the top N_{p}-th is denoted by $w_{N_{\text{p}}}$. We obtain the corresponding patch central points and their patches within points, respectively given by $(\tilde{\mathbf{U}}, \tilde{\mathbf{V}}) = \{(\mathbf{u}_i, \mathbf{v}_j) | (\mathbf{u}_i, \mathbf{v}_j) \in (\mathbf{U}, \mathbf{V}), w_{i,j} \geq w_{N_{\text{p}}}, w_{i,j} \in \mathbf{W}_{\mathbf{UV}}\}$ and $(\tilde{\mathcal{U}}, \tilde{\mathcal{V}}) = \{(\mathcal{U}_i, \mathcal{V}_j) | (\mathcal{U}_i, \mathcal{V}_j) \in (\mathcal{U}, \mathcal{V}), w_{i,j} \geq w_{N_{\text{p}}}, w_{i,j} \in \mathbf{W}_{\mathbf{UV}}\}$.

Furthermore, for each pair of corresponding patches within points, e.g. $(\tilde{\mathcal{U}}_l, \tilde{\mathcal{V}}_l) \in (\tilde{\mathcal{U}}, \tilde{\mathcal{V}})$, we compute cosine similarity with post-processing Sinkhorn algorithm (Sarlin et al. 2020) to obtain the similarity score matrix and use it to handle the point features. Then, using the Top-K method, we obtain the corresponding points in this pair of patches similar to the processing in (Qin et al. 2022; Sarlin et al. 2020).

Then, we use registration methods such as RANSAC (Fischler and Bolles 1981), weighted singular value decomposition (SVD) (Besl and McKay 1992) or local-to-global registration (LGR) (Qin et al. 2022) to predict the rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{t}}$ based on $(\tilde{\mathcal{U}}, \tilde{\mathcal{V}})$. In this paper, the LGR method is used to achieve the point-level registration.

Loss function. Due to advantages of learnable weights (Wang et al. 2022, 2020), we adopt a loss as follows

$$\mathcal{L} = \exp(-\varpi)\mathcal{L}_{\text{patch}} + \varpi + \exp(-\varrho)\mathcal{L}_{\text{point}} + \varrho, \tag{11}$$

where ϖ and ϱ are learnable parameters. $\mathcal{L}_{\text{patch}}$ and $\mathcal{L}_{\text{point}}$ are the overlap-aware circle loss (Qin et al. 2022) and negative log-likehood loss (Sarlin et al. 2020) respectively.

Experiments

Datasets. The Boreas dataset (Burnett et al. 2023), a publicly accessible street dataset comprising LiDAR and camera data, is used in our experiments. It encompasses diverse weather conditions, such as snow, rain, and nighttime scenarios. Notably, this dataset provides meticulously post-processed ground-truth poses. Leveraging these ground-truth poses, we can readily derive the transformation matrix for each adjacent pair of LiDAR point clouds. The KITTI dataset (Geiger et al. 2013) is also used which includes multi-sensor data. This dataset consists of 11 sequences capturing various street scenes, and it also offers global ground-truth poses. More details are provided in the supplementary material.

Implementation Details. We set the dimension d to 256 in (5). For handling the neighborhood graph of the k nearest neighbors, where $k = 15$, we employ the graph learning layer f_{BND} in (5) as a composition of EdgeConv layers (Wang et al. 2019). Specifically, we utilize two EdgeConv layers, with hidden input and output dimensions of $[1024, 512]$ and $[1536, 512]$ respectively. We also use the DGCNN and the Transformer based on self-cross attention whose architectures are identical to those in (Wang et al. 2019) and (Wang and Solomon 2019) to extract features of window central points. Regarding the Transformer modules, we employ four attention heads, each with 128 hidden features, resulting in a total of 512 hidden features. We adopt the Adam optimizer (Kingma and Ba 2015) with a learning rate of 0.0001. The number of training epochs is set to 50. The model is executed on an NVIDIA RTX A5000 GPU. More details are provided in the supplementary material.

Results and Analysis

Performance on datasets with dynamic object perturbations. We assess the point cloud registration performance of PosDiffNet and compare it against various baseline methods. *The training data is based on the subset of the Boreas dataset collected under sunny weather conditions.* During the testing phase, we evaluate PosDiffNet in three distinct categories.

Method	Testing on the Boreas (Sunny)				Testing on the Boreas (Night)				Testing on the KITTI			
	RTE (cm)		*RRE (°)*		*RTE (cm)*		*RRE (°)*		*RTE (cm)*		*RRE (°)*	
	MAE	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
ICP	11.97	33.99	0.14	0.35	10.83	18.28	0.11	0.21	9.86	19.48	0.17	0.27
HGNN++	14.31	24.41	0.15	0.34	16.06	25.86	0.15	0.27	8.86	17.20	0.20	0.31
VCR-Net	10.47	21.74	0.13	0.29	11.97	19.78	0.11	0.19	5.31	11.07	0.16	0.24
PCT++	13.22	24.41	0.15	0.34	11.61	19.57	0.13	0.31	6.16	13.96	0.18	0.28
GeoTrans	**3.98**	18.09	0.09	0.27	**5.97**	27.90	0.09	0.33	**3.93**	13.50	0.18	0.50
BUFFER	6.75	8.23	0.10	**0.12**	9.17	10.86	0.12	0.14	5.38	**5.76**	0.18	0.29
RoITr	7.73	13.34	0.10	0.18	9.61	14.00	0.09	0.14	6.97	11.48	0.19	0.29
PosDiffNet	4.30	**7.32**	**0.08**	0.16	6.65	**9.47**	**0.08**	**0.13**	3.97	6.44	**0.15**	**0.23**

Table 2: Point cloud registration performance using the KITTI dataset for training.

Weather	Method	RTE (cm)		RRE (°)	
		MAE	*RMSE*	*MAE*	*RMSE*
Rain	ICP	11.90	20.57	0.15	0.27
	DCP	10.60	16.00	0.14	0.22
	HGNN++	15.02	25.63	0.18	0.32
	VCR-Net	8.81	14.09	0.13	0.20
	PCT++	10.39	16.86	0.14	0.24
	GeoTrans	4.96	16.75	0.10	0.25
	BUFFER	8.00	8.36	0.12	0.18
	RoITr	8.01	11.53	0.11	0.16
	PosDiffNet	**4.56**	**6.26**	**0.09**	**0.14**
Snow	ICP	8.27	12.59	0.10	0.15
	DCP	7.82	11.51	0.12	0.19
	HGNN++	9.53	14.55	0.13	0.21
	VCR-Net	5.65	8.48	0.09	0.13
	PCT++	6.66	10.20	0.10	0.15
	GeoTrans	**3.90**	11.27	0.08	0.19
	BUFFER	7.00	7.58	0.09	**0.10**
	RoITr	8.67	12.82	0.10	0.15
	PosDiffNet	4.18	**5.89**	**0.07**	0.11

Table 3: Performance on the Boreas dataset under rainy and snowy weather conditions.

The first and second categories are subsets of the Boreas dataset, captured under different weather conditions: sunny and night, respectively. The third category consists of a subset of the KITTI dataset, where the point clouds are collected under sunny weather conditions. The experimental results, presented in Table 1, showcase the superior performance of PosDiffNet compared to the baseline methods. PosDiffNet achieves better results across most evaluation metrics, including root mean square error (RMSE) and mean absolute error (MAE), for the relative translation error (RTE) and relative rotation error (RRE).

Furthermore, we assess the performance and conduct a comparative analysis using the subset of KITTI dataset as the training dataset. During the testing phase, we employ the same three categories as mentioned in the previous subsection. From Table 2, we observe that PosDiffNet consistently outperforms the other baseline methods in most cases, demonstrating its superior registration performance.

Performance on datasets under bad weather conditions. To assess the robustness of PosDiffNet against natural noise, we conduct experiments on the Boreas dataset under adverse weather conditions such as rain and snow. From Table 3, it is evident that PosDiffNet outperforms the baseline methods across all evaluation criteria. This indicates the superior performance of PosDiffNet in challenging rainy conditions. Similarly, from Table 3, we observe that PosDiffNet has lower RMSE in RTE compared to the baseline methods. These results suggest that PosDiffNet produces fewer outliers among the predicted results, further verifying its robustness in handling snowy conditions compared to the baselines.

Method	RTE(cm)		RTE (°)	
	MAE	*RMSE*	*MAE*	*RMSE*
ICP	14.97	26.09	0.20	0.32
DCP	9.97	15.84	0.29	0.52
HGNN++	10.62	18.76	0.22	0.34
VCR-Net	6.40	12.40	**0.18**	**0.27**
PCT++	6.85	14.03	0.20	0.30
GeoTrans	5.37	14.43	0.25	0.50
BUFFER	6.12	7.04	0.23	0.36
RoITr	9.79	14.94	0.27	0.45
PosDiffNet	**4.84**	**6.93**	0.20	0.33

Table 4: Point cloud registration performance on the KITTI dataset with additive white Gaussian noise.

Performance on datasets with additive white Gaussian noise. We evaluate the robustness of PosDiffNet under the presence of additive white Gaussian noise $\mathcal{N}(\mu = 0, \sigma = 0.25)$ in the KITTI dataset during the testing phase. From Table 4, we observe that PosDiffNet outperforms the other benchmark methods in terms of relative translation prediction. Comparing Table 4 with Table 2, we note that PosDiffNet experiences a smaller degradation in relative rotation prediction compared to the baselines. These findings demonstrate the crucial role of PosDiffNet in handling additive white Gaussian noise, particularly in scenarios where accurate relative translation prediction is required.

Overlapping Discussion. We conduct the experiments under lower overlapping conditions using the KITTI dataset with the 10-m frame interval between each pair of frames. From Table 5, we observe that PosDiffNet outperforms or is on par with the SOTA baselines, which evaluates the efficiency of our method.

Method	TE(cm)	RE(°)	RR(%)
3DFeat-Net	25.9	0.25	96.0
D3Feat	7.2	0.30	**99.8**
SpinNet	9.9	0.47	99.1
Predator	6.8	0.27	**99.8**
CoFiNet	8.2	0.41	**99.8**
PointDSC	8.1	0.35	98.2
SC^2-PCR	7.2	0.32	99.6
GeoTrans	6.8	**0.24**	**99.8**
MAC	8.5	0.40	99.5
DGR	∼32	0.37	98.7
HRegNet	∼12	0.29	99.7
UDPReg	∼8.8	0.41	64.6
SuperLine3D	∼8.7	0.59	97.7
PosDiffNet	**6.6**	**0.24**	**99.8**

Table 5: Performance on the 10-m frame KITTI dataset (the same setting as that in (Qin et al. 2022; Zhang et al. 2023)). The results of baselines are borrowed from (Qin et al. 2022; Chen et al. 2022; Zhao et al. 2022; Mei et al. 2023; Zhang et al. 2023). The metrics are the same as those in (Zhang et al. 2023). "∼" indicates the lack of a dataset setting description or a setting similar to that in (Qin et al. 2022; Zhang et al. 2023). The PointDSC, SC^2-PCR, and MAC are based on the FPFH method (Rusu, Blodow, and Beetz 2009).

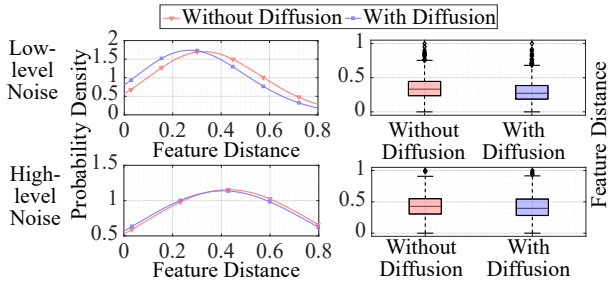

Figure 4: Kernel density estimate plots and box plots for the normalized feature distance between noisy and clean conditions for the modules with or without Beltrami diffusion. The additive noises include two Gaussian noises following $\mathcal{N}(0, \sigma = 0.25)$ and $\mathcal{N}(0, \sigma = 1.5)$, corresponding to the low-level and high-level noises.

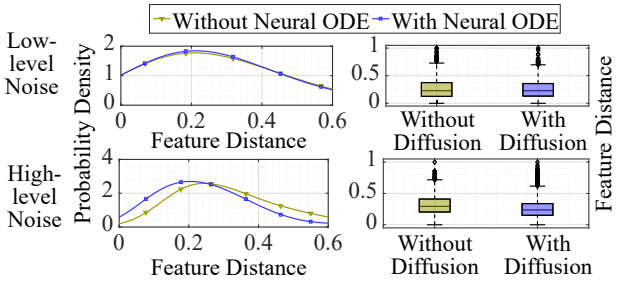

Figure 5: Kernel density estimate plots and box plots for the normalized feature distance between noisy and clean conditions for the Transformer with or without neural ODE.

Method	RTE (cm)		RRE (°)	
	MAE	*RMSE*	*MAE*	*RMSE*
full model (with all modules)	**3.97**	**6.44**	**0.15**	**0.23**
w/o window	4.34	6.99	**0.15**	**0.23**
w/o Beltrami diffusion	5.20	8.21	0.16	0.25
w/o diffusion-based Transformer	11.52	18.72	0.25	0.42
w/ Geometric Transformer	4.23	6.89	**0.15**	0.24

Table 6: Ablation for the modules.

Method	RTE (cm)		RRE (°)	
	MAE	*RMSE*	*MAE*	*RMSE*
full Transformer (with all components)	**3.97**	**6.44**	**0.15**	**0.23**
w/o Beltrami embedding	4.32	6.95	**0.15**	**0.23**
w/o neural ODE	4.81	7.79	0.16	0.24
w/o neural ODE & Beltrami	5.87	9.69	0.17	0.27
w/ Geometric (positional) embedding	4.07	6.55	**0.15**	**0.23**

Table 7: Ablation for the Transformer module.

Robustness of diffusion modules. We compare the feature distances based on $\mathcal{L}_2$ distance between different levels of noisy conditions and the clean condition for the output of the modules with and without diffusion. From Figs. 4 and 5, the statistics of feature distances with diffusion demonstrate superior performance and stability compared to those without diffusion. This indicates the robustness of diffusion.

Ablation Study. We evaluate the effectiveness of each module and the Transformer in our design. The performance improvements are observed through the evaluation of individual modules or components of the Transformer, as shown in Table 6 and Table 7.

Limitation Discussion. The resource-intensive nature of diffusion and attention computation presents challenges for resource-constrained devices. Future research will focus on exploring model miniaturization techniques to mitigate these constraints.

Conclusion

In this work, we introduce PosDiffNet, a model that combines a joint window-patch-point correspondence method with neural Beltrami flow and diffusion-based Transformer. PosDiffNet facilitates the simultaneous processing of point features and position information and achieves SOTA performance on datasets in large fields of view, demonstrating its effectiveness and robustness.

Acknowledgments

This research is supported by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE-T2EP20220-0002, and the National Research Foundation, Singapore and Infocomm Media Development Authority under its Future Communications Research and Development Programme. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Ao, S.; Hu, Q.; Wang, H.; Xu, K.; and Guo, Y. 2023. BUFFER: Balancing Accuracy, Efficiency, and Generalizability in Point Cloud Registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 1255–1264.

Ao, S.; Hu, Q.; Yang, B.; Markham, A.; and Guo, Y. 2021. SpinNet: Learning a general surface descriptor for 3d point cloud registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 11753–11762.

Bai, X.; Luo, Z.; Zhou, L.; Chen, H.; Li, L.; Hu, Z.; Fu, H.; and Tai, C.-L. 2021. Pointdsc: Robust point cloud registration using deep spatial consistency. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 15859–15869.

Bai, X.; Luo, Z.; Zhou, L.; Fu, H.; Quan, L.; and Tai, C.-L. 2020. D3feat: Joint learning of dense detection and description of 3d local features. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 6359–6367.

Besl, P. J.; and McKay, N. D. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, 586–606.

Burnett, K.; Yoon, D. J.; Wu, Y.; Li, A. Z.; Zhang, H.; Lu, S.; Qian, J.; Tseng, W.-K.; Lambert, A.; Leung, K. Y.; et al. 2023. Boreas: A multi-season autonomous driving dataset. *Int. J. Robot. Res.*, 42(1-2): 33–42.

Chamberlain, B.; Rowbottom, J.; Eynard, D.; Di Giovanni, F.; Dong, X.; and Bronstein, M. 2021a. Beltrami flow and neural diffusion on graphs. *Adv. Neural Inform. Process. Syst.*, 34: 1594–1609.

Chamberlain, B. P.; Rowbottom, J.; Goronova, M.; Webb, S.; Rossi, E.; and Bronstein, M. M. 2021b. GRAND: Graph Neural Diffusion. In *Proc. Int. Conf. Mach. Learn.*

Chen, Z.; Sun, K.; Yang, F.; and Tao, W. 2022. SC2-PCR: A second order spatial compatibility for efficient and robust point cloud registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 13221–13231.

Choy, C.; Park, J.; and Koltun, V. 2019. Fully convolutional geometric features. In *Proc. Int. Conf. Comput. Vis.*, 8958–8966.

Fischer, K.; Simon, M.; Olsner, F.; Milz, S.; Gross, H.-M.; and Mader, P. 2021. Stickypillars: Robust and efficient feature matching on point clouds using graph neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 313–323.

Fischler, M. A.; and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6): 381–395.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The KITTI dataset. *Int. J. Rob. Res.*, 32(11): 1231–1237.

Guérout, T.; Gaoua, Y.; Artigues, C.; Da Costa, G.; Lopez, P.; and Monteil, T. 2017. Mixed integer linear programming for quality of service optimization in Clouds. *Future Generation Computer Systems*, 71: 1–17.

Guo, M.-H.; Cai, J.-X.; Liu, Z.-N.; Mu, T.-J.; Martin, R. R.; and Hu, S.-M. 2021. PCT: Point cloud transformer. *Comput. Vis. Media.*, 7: 187–199.

Huang, S.; Gojcic, Z.; Usvyatsov, M.; Wieser, A.; and Schindler, K. 2021. Predator: Registration of 3d point clouds with low overlap. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 4267–4276.

Kang, Q.; She, R.; Wang, S.; Tay, W. P.; Navarro, D. N.; and Hartmannsgruber, A. 2022. Location learning for AVs: LiDAR and image landmarks fusion localization with graph neural networks. In *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 3032–3037.

Kimmel, R.; Sochen, N.; and Malladi, R. 1997. From high energy physics to low level vision. In *Proc. Int. Conf. Scale-Space Theory Comput. Vision*, 236–247.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Represent.*, 1–15.

Koide, K.; Yokozuka, M.; Oishi, S.; and Banno, A. 2021. Voxelized GICP for fast and accurate 3d point cloud registration. In *Proc. IEEE Int. Conf. Robot. Autom.*, 11054–11059.

Kopuklu, O.; Kose, N.; Gunduz, A.; and Rigoll, G. 2019. Resource efficient 3d convolutional neural networks. In *Proc. Int. Conf. Comput. Vis. Worksh.*, 0–0.

Kumawat, S.; and Raman, S. 2019. LP-3DCNN: Unveiling local phase in 3d convolutional neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 4903–4912.

Li, J.; Chen, B. M.; and Lee, G. H. 2018. So-net: Self-organizing network for point cloud analysis. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 9397–9406.

Li, J.; Qin, H.; Wang, J.; and Li, J. 2021. OpenStreetMap-based autonomous navigation for the four wheel-legged robot via 3D-LiDAR and CCD camera. *IEEE Trans. Ind. Electron.*, 69(3): 2708–2717.

Li, Y.; and Harada, T. 2022. Lepard: Learning partial point cloud matching in rigid and deformable scenes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 5554–5564.

Lu, F.; Chen, G.; Liu, Y.; Zhang, L.; Qu, S.; Liu, S.; and Gu, R. 2021. HRegNet: A hierarchical network for large-scale outdoor LiDAR point cloud registration. In *Proc. Int. Conf. Comput. Vis.*, 16014–16023.

Ma, X.; Qin, C.; You, H.; Ran, H.; and Fu, Y. 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *Proc. Int. Conf. Learn. Represent.*

Mei, G.; Tang, H.; Huang, X.; Wang, W.; Liu, J.; Zhang, J.; Van Gool, L.; and Wu, Q. 2023. Unsupervised Deep Probabilistic Approach for Partial Point Cloud Registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 13611–13620.

Pais, G. D.; Ramalingam, S.; Govindu, V. M.; Nascimento, J. C.; Chellappa, R.; and Miraldo, P. 2020. 3dregnet: A deep neural network for 3d point registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 7193–7203.

Poiesi, F.; and Boscaini, D. 2022. Learning general and distinctive 3D local deep descriptors for point cloud registration. *IEEE Trans. Pattern Anal. Mach. Intell.*

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 652–660.

Qian, G.; Li, Y.; Peng, H.; Mai, J.; Hammoud, H.; Elhoseiny, M.; and Ghanem, B. 2022. PointNeXt: Revisiting pointnet++ with improved training and scaling strategies. *Adv. Neural Inform. Process. Syst.*, 35: 23192–23204.

Qin, Z.; Yu, H.; Wang, C.; Guo, Y.; Peng, Y.; and Xu, K. 2022. Geometric transformer for fast and robust point cloud registration. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 11143–11152.

Quan, S.; and Yang, J. 2020. Compatibility-guided sampling consensus for 3-d point cloud registration. *IEEE Trans. Geosci. Remote. Sens.*, 58(10): 7380–7392.

Rusu, R. B.; Blodow, N.; and Beetz, M. 2009. Fast point feature histograms (FPFH) for 3D registration. In *Proc. Proc. IEEE Int. Conf. Robot. Automat.*, 3212–3217.

Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. Superglue: Learning feature matching with graph neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 4938–4947.

Segal, A.; Haehnel, D.; and Thrun, S. 2009. Generalized-ICP. In *Robotics: science and systems*, 435. Seattle, WA.

Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; and Rus, D. 2020. LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping. In *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 5135–5142.

She, R.; Kang, Q.; Wang, S.; Yang, Y.-R.; Zhao, K.; Song, Y.; and Tay, W. P. 2023. RobustMat: Neural diffusion for street landmark patch matching under challenging environments. *IEEE Trans. Image Process.*, 32: 5550–5563.

Sindagi, V. A.; Zhou, Y.; and Tuzel, O. 2019. Mvx-net: Multimodal voxelnet for 3d object detection. In *Proc. IEEE Int. Conf. Robot. Autom.*, 7276–7282.

Song, Y.; Kang, Q.; Wang, S.; Zhao, K.; and Tay, W. P. 2022. On the robustness of graph neural diffusion to topology perturbations. *Adv. Neural Inform. Process. Syst.*, 35: 6384–6396.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. Int. Conf. Comput. Vis.*, 945–953.

Sun, J.; Shen, Z.; Wang, Y.; Bao, H.; and Zhou, X. 2021. LoFTR: Detector-free local feature matching with transformers. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 8922–8931.

Thomas, H.; Qi, C. R.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proc. Int. Conf. Comput. Vis.*, 6411–6420.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Adv. Neural Inform. Process. Syst.*, 30.

Wang, B.; Chen, C.; Lu, C. X.; Zhao, P.; Trigoni, N.; and Markham, A. 2020. Atloc: Attention guided camera localization. In *Proc. AAAI Conf. Artificial Intell.*, 10393–10401.

Wang, H.; Liu, Y.; Hu, Q.; Wang, B.; Chen, J.; Dong, Z.; Guo, Y.; Wang, W.; and Yang, B. 2023a. RoReg: Pairwise Point Cloud Registration with Oriented Descriptors and Local Rotations. *IEEE Trans. Pattern Anal. Mach. Intell.*

Wang, S.; Kang, Q.; She, R.; Tay, W. P.; Hartmannsgruber, A.; and Navarro, D. N. 2022. RobustLoc: Robust Camera Pose Regression in Challenging Driving Environments. In *Proc. AAAI Conf. Artificial Intell.*

Wang, S.; Kang, Q.; She, R.; Wang, W.; Zhao, K.; Song, Y.; and Tay, W. P. 2023b. HypLiLoc: Towards Effective LiDAR Pose Regression with Hyperbolic Fusion. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*

Wang, Y.; and Solomon, J. M. 2019. Deep closest point: Learning representations for point cloud registration. In *Proc. Int. Conf. Comput. Vis.*, 3523–3532.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graphics*, 38(5): 1–12.

Wei, H.; Qiao, Z.; Liu, Z.; Suo, C.; Yin, P.; Shen, Y.; Li, H.; and Wang, H. 2020. End-to-end 3d point cloud learning for registration task using virtual correspondences. In *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2678–2683.

Yu, H.; Li, F.; Saleh, M.; Busam, B.; and Ilic, S. 2021. CoFiNet: Reliable coarse-to-fine correspondences for robust pointcloud registration. *Adv. Neural Inform. Process. Syst.*, 34: 23872–23884.

Yu, H.; Qin, Z.; Hou, J.; Saleh, M.; Li, D.; Busam, B.; and Ilic, S. 2023. Rotation-invariant transformer for point cloud matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 5384–5393.

Yu, J.; Lin, Y.; Wang, B.; Ye, Q.; and Cai, J. 2019. An advanced outlier detected total least-squares algorithm for 3-D point clouds registration. *IEEE Trans. Geosci. Remote. Sens.*, 57(7): 4789–4798.

Zhang, X.; Yang, J.; Zhang, S.; and Zhang, Y. 2023. 3D Registration with Maximal Cliques. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 17745–17754.

Zhang, Z.; Dai, Y.; Fan, B.; Sun, J.; and He, M. 2022. Learning a task-specific descriptor for robust matching of 3d point clouds. *IEEE Trans. Circuits Syst. Video Technol.*, 32(12): 8462–8475.

Zhao, K.; Kang, Q.; Song, Y.; She, R.; Wang, S.; and Tay, W. P. 2023. Adversarial robustness in graph neural networks: A Hamiltonian approach. *Adv. Neural Inform. Process. Syst.*

Zhao, X.; Yang, S.; Huang, T.; Chen, J.; Ma, T.; Li, M.; and Liu, Y. 2022. SuperLine3D: Self-supervised Line Segmentation and Description for LiDAR Point Cloud. In *Proc. Eur. Conf. Comput. Vis.*, 263–279.