2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

# Fast and Robust Point Cloud Registration with Tree-based Transformer

Guangyan Chen[1], Meiling Wang[1], Yi Yang[1], Li Yuan[2], Yufeng Yue[1*]

***Abstract*— Point cloud registration is essential in computer vision and robotics. Recently, transformer-based methods have achieved advanced point cloud registration performance. However, the standard attention mechanism utilized in these methods considers many low-relevance points, and it has difficulty focusing its attention weights on sparse and meaningful points, leading to limited local structure modeling capabilities and quadratic computational complexity. To address these limitations, we present the Tree-based Transformer (TrT), which is able to extract abundant local and global features with linear computational complexity. Specifically, the TrT builds coarse-to-dense feature trees, and a novel Tree-based Attention (TrA) is proposed to guide the progressive convergence of the attended regions toward meaningful points and to structurize point clouds following tree structures. In each layer, the top $\mathcal{S}$ key points with the highest attention scores are selected, such that in the next layer, attention is evaluated only within the specified high-relevance regions, corresponding to the child points of these selected $\mathcal{S}$ points. Additionally, coarse features containing high-level semantic information are incorporated into the child points to guide the feature extraction process, facilitating local structure modeling and multiscale information integration. Consequently, TrA enables the model to focus on critical local structures and extract rich local information with linear computational complexity. Experiments demonstrate that our method achieves state-of-the-art performance on 3DMatch and KITTI benchmarks. The code for our method is publicly available at https://github.com/CGuangyan-BIT/TrT.**

## I. Introduction

The aim of point cloud registration is to recover an optimal transformation for point cloud pair alignment, which is a fundamental problem in computer vision and robotics. Recent advances in 3D point representation learning are pushing point cloud registration beyond the traditional methods [5], [32], [43] to learning-based methods [3], [11], [35]. The most widely known traditional method is the iterative closest point (ICP) [5], which iterates between establishing correspondences and calculating a transformation. However, ICP and its variants [31], [32] are prone to becoming stuck in local minima when the initial errors are large. To achieve increased registration accuracy, learning-based methods [3], [11] integrate neural networks to extract pointwise features separately and establish correspondences based on feature similarity. However, the independence across point clouds produces obstacles when identifying common structures and extracting distinct features.

Recent learning-based methods [7], [8], [14], [35] have attempted to address these issues through the incorporation of transformer models, leveraging their inherent strengths in handling order-invariance and dependency modeling. These approaches enable one point cloud to perceive another point cloud and extract the contextual information between them, enhancing the discriminative power of the extracted features. However, the standard attention mechanism considers many low-relevance points, thereby encountering difficulties in concentrating its attention weights toward meaningful regions. Consequently, this results in limited capabilities for local feature extraction and imposes quadratic computational complexity.

Many recent studies [10], [20], [42] have explored local attention mechanisms for point cloud processing tasks, which prunes low-relevance points by constraining the attention fields to fixed, predefined patterns, such as local neighborhoods [42]. However, this hinders their ability to focus on high-relevance regions dynamically. Additionally, most of these methods are based on assumptions regarding location and correlations between points, e.g., spatially close points are correlated. Such assumptions are typically inaccurate in cross-point-cloud scenarios, making it difficult to employ predefined patterns for cross-attention. Therefore, there is still a crucial need for point cloud registration to design a transformer model that can encode important local structures and reduce computational complexity.

To achieve this goal, we propose the Tree-based Transformer (TrT), which is able to focus on critical local structures and achieve linear computational complexity without predefined attention sparsity. The TrT is built on the basis of the proposed Tree-based Attention (TrA) module, which drives the hierarchical convergence of the attended regions and incorporates the spatially coarse features into the child points to guide the feature extraction process. In the 1st layer, the attention computation for the query point encompasses all key points, from which the top $\mathcal{S}$ (here, $\mathcal{S} = 2$) points with the highest attention scores, highlighted in orange, are selected. In the 2nd layer, for the child points corresponding to the query point in the previous layer (1st), attention is calculated exclusively among the child points of the corresponding $\mathcal{S}$ keys selected in the previous layer, thereby skipping low-relevance points and reducing the computational complexity. Furthermore, the features derived from the previous layer are utilized to guide the feature extraction procedure for the child points. These processes are replicated in the 3rd layer, using the top $\mathcal{S}$ points selected

*Corresponding author: Yufeng Yue (yueyufeng@bit.edu.cn).

This work was supported by the National Natural Science Foundation of China under Grant No. NSFC 92370203, 62233002, the National Key RD Program of China (2022ZD0118).

[1]Guangyan Chen, Meiling Wang, Yi Yang, and Yufeng Yue are with the School of Automation, Beijing Institute of Technology, Beijing, 100081, China.

[2]Li Yuan is with the School of Electrical and Computer Engineering at Peking University & Pecheng Lab, Shenzhen, 518055, China.

in the 2nd layer. In this manner, TrA enables our method to adaptively specify high-relevance locations as attended areas and focus on critical local structures. The main contributions are three-fold:

- The TrT is proposed by integrating tree structures into the transformer model, allowing the model to extract rich local features and achieve linear computational complexity with the learned attended regions.
- The TrA is proposed to hierarchically and dynamically identify high-relevance key points and structurize point clouds along the tree, facilitating local structure modeling and multiscale information aggregation.
- Extensive experiments show that our method outperforms the baselines and achieves SOTA performance on the 3DMatch and KITTI benchmarks.

## II. Related Work

### A. Transformer-based Methods for Registration

Inspired by the success of transformers in natural language processing (NLP) [6], [19] and computer vision tasks [24], [38], [39], researchers have adapted them to point cloud registration. The deep closest point (DCP) [35] utilizes a dynamic graph CNN (DGCNN) [28] to separately extract features and introduces a transformer [34] to model relations across a pair of point clouds. The robust graph matching (RGM) method [14] adopts a transformer to improve the quality of the correspondences by aggregating information along graph edges. Predator [17] leverages self- and cross-attention mechanisms to perform information aggregation across a pair of point clouds and predict overlapping regions for feature sampling, which significantly improves the proportion of successful registrations in low-overlap scenarios. CoFiNet [37] alternately uses self-attention and cross-attention to extract features in a coarse-to-fine manner and achieves promising performance. The registration transformer (RegTR) [36] utilizes attention layers to directly generate correspondences. Geometric transformer [29] calculates pair-wise distances and triplet-wise angles, which are then combined with self-attention to capture geometric features, obtaining robust superpoint matching. DIT [8] introduces a full Transformer network that leverages the transformer architecture to extract local features and facilitate deep information interaction. RegFormer [23] presents an end-to-end transformer network designed for large-scale point cloud alignment. It achieves competitive performance in accuracy and efficiency, all without the need for additional post-processing steps. In general, most current transformer-based methods utilize attention mechanisms for contextual information learning. However, the standard attention mechanism struggles to focus its attention weights on meaningful points, leading to limited local structure modeling ability.

### B. Local Attention for Point Clouds

To assist transformers in focusing on meaningful points and enhance their local feature extraction capabilities, local attention mechanisms [10], [13], [20] introduce a local inductive bias by restricting the attention fields to local regions instead of the entire point cloud. The sparse voxel transformer [13] encodes short-range local relations based on voxels and learns long-range contextual relations based on clusters. The point transformer [42] applies an attention mechanism in the local neighborhood of the given points. PatchFormer [10] splits a raw point cloud into M patches, aggregates the local features in each patch, and then approximates the global attention map. The stratified transformer [20] captures the short-range dependencies within a voxel and models long-range relations with respect to the downsampled point cloud outside the voxel. In summary, most existing local attention mechanisms for point clouds limit the attention field by considering a fixed pattern. This makes it difficult to accurately attend to high-relevance points and achieve cross-attention, which is important for registration.

## III. Tree-based Transformer

### A. Preliminaries

Transformers have shown great potential in processing point clouds, benefiting from their ability to handle order-invariance and model dependencies through attention mechanisms. The multihead attention operation (MA) executes $H$ attention functions Att in parallel. Given a pair of point cloud embeddings $\boldsymbol{F}^{\tilde{X}}$ and $\boldsymbol{F}^{\tilde{Y}}$ as inputs to MA, each Att generates queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$, using projection matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$, respectively:

$$\boldsymbol{Q} = \boldsymbol{F}^{\tilde{X}}\boldsymbol{W}^Q, \quad \boldsymbol{K} = \boldsymbol{F}^{\tilde{Y}}\boldsymbol{W}^K, \quad \boldsymbol{V} = \boldsymbol{F}^{\tilde{Y}}\boldsymbol{W}^V. \tag{1}$$

Then, each Att obtains an attention map via a scaled dot-product operation and multiplies this map by $V$ for information aggregation. Subsequently, the results are concatenated and projected with $\boldsymbol{W}^O$ to obtain the final values:

$$\begin{aligned} \boldsymbol{A} &= \mathrm{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_K}})\boldsymbol{V}, \\ \mathrm{MA}(\boldsymbol{F}^{\tilde{X}}, \boldsymbol{F}^{\tilde{Y}}) &= \mathrm{Concat}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_H)\boldsymbol{W}^O, \end{aligned} \tag{2}$$

where $d_K$ is the dimensionality of the keys $\boldsymbol{K}$. The attention mechanism establishes associations across $\boldsymbol{F}^{\tilde{X}}$ and $\boldsymbol{F}^{\tilde{Y}}$, enabling $\boldsymbol{F}^{\tilde{X}}$ to receive information from $\boldsymbol{F}^{\tilde{Y}}$. Despite its versatile and powerful relation modeling capabilities, the standard attention suffers from a limited local feature extraction ability and quadratic computational complexity.

### B. Overall Architecture

The objective of point cloud registration is to estimate a rotation matrix $\hat{\boldsymbol{R}} \in SO(3)$ and a translation vector $\hat{\boldsymbol{t}} \in \mathbb{R}^3$ to align a source point cloud $\boldsymbol{X} = \{x_1, x_2, ..., x_M\} \subseteq \mathbb{R}^3$ with a target point cloud $\boldsymbol{Y} = \{y_1, y_2, ..., y_N\} \subseteq \mathbb{R}^3$.

The overall TrT pipeline is illustrated in Fig. 1. The TrT begins with a kernel point convolution (KPConv) [33] (Sec. III-C) that downsamples point clouds $\boldsymbol{X}$, $\boldsymbol{Y}$ into smaller sets of points $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{Y}}$ and extracts pointwise features $\boldsymbol{F}^{\tilde{X}}$, $\boldsymbol{F}^{\tilde{Y}}$. Subsequently, a tree transformer encoder learns contextual information and extracts features $\boldsymbol{\mathcal{F}}^{\tilde{X}}$, $\boldsymbol{\mathcal{F}}^{\tilde{Y}}$ with rich local information (Sec. III-D). These features are then utilized to generate the corresponding point clouds $\hat{\boldsymbol{Y}}$, $\hat{\boldsymbol{X}}$ and predict

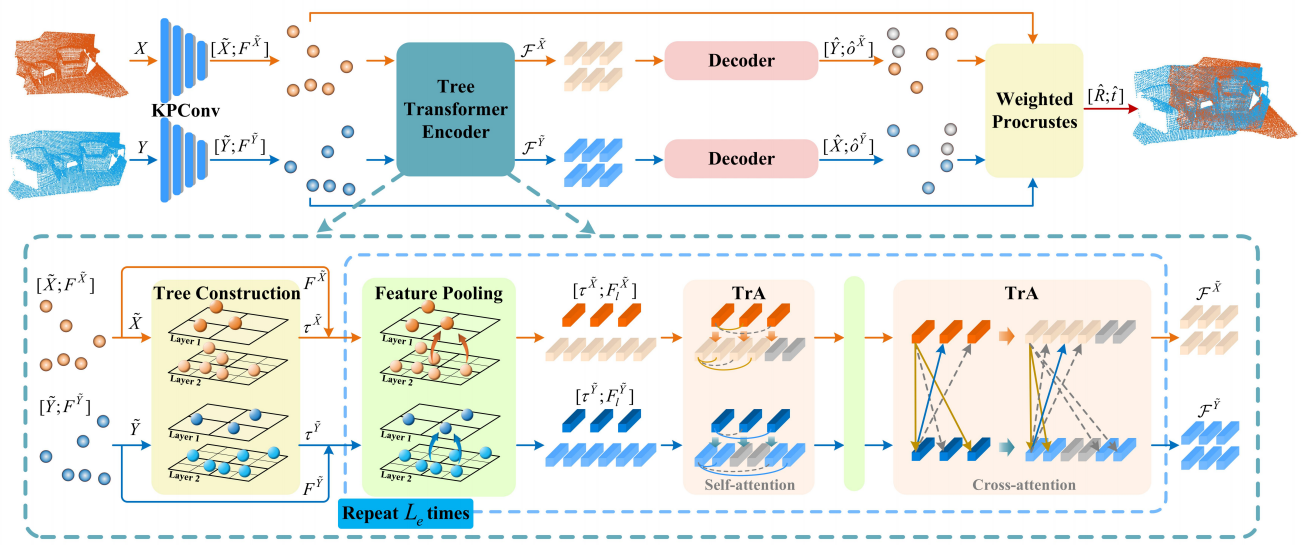


Fig. 1. Network architecture of the TrT. First, the KPConv extracts features for a sparse set of points. Subsequently, the tree transformer encoder builds feature trees and iteratively extracts features containing local and global information. Then, the decoder predicts the corresponding point clouds and overlap scores. Finally, a transformation is computed to align the point clouds.

overlap scores $\hat{\boldsymbol{o}}^{\tilde{X}}$, $\hat{\boldsymbol{o}}^{\tilde{Y}}$ in the decoder (Sec. III-E). Finally, a weighted Procrustes module estimates the optimal transformation $\{\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}}\}$ based on the predicted correspondences $\{\tilde{\boldsymbol{X}}, \hat{\boldsymbol{Y}}\}$, $\{\tilde{\boldsymbol{Y}}, \hat{\boldsymbol{X}}\}$ and the overlap scores $\hat{\boldsymbol{o}}^{\tilde{X}}$, $\hat{\boldsymbol{o}}^{\tilde{Y}}$.

### C. Downsampling and Feature Extraction

Following [17], a KPConv backbone, which consists of ResNet-like blocks and strided convolutions, is utilized for downsampling and feature extraction. Specifically, the KPConv backbone downsamples the point clouds $\boldsymbol{X} \in \mathbb{R}^{M\times 3}$, $\boldsymbol{Y} \in \mathbb{R}^{N\times 3}$ to $\tilde{\boldsymbol{X}} \in \mathbb{R}^{M'\times 3}$, $\tilde{\boldsymbol{Y}} \in \mathbb{R}^{N'\times 3}$ and performs feature extraction. The extracted features are then projected to obtain features $\boldsymbol{F}^{\tilde{X}} \in \mathbb{R}^{M'\times D}$, $\boldsymbol{F}^{\tilde{Y}} \in \mathbb{R}^{N'\times D}$.

### D. Tree Transformer Encoder

The downsampled point clouds $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{Y}}$ and the extracted features $\boldsymbol{F}^{\tilde{X}}$, $\boldsymbol{F}^{\tilde{Y}}$ are processed by the tree transformer encoder. The encoder is composed of a tree construction layer, as well as $L_e$ encoder layers for feature extraction. Each encoder layer consists of two feature pooling sub-layers and two TrA sub-layers.

**Tree construction**. To encourage the attention weights to converge toward meaningful points and progressively structurize point clouds, tree structures are employed to represent point clouds. Specifically, $L_\tau$-layer trees $\boldsymbol{\tau}^{\tilde{X}}$ and $\boldsymbol{\tau}^{\tilde{Y}}$ are built upon point clouds $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$, respectively, by first dividing the point clouds into voxels and then hierarchically grouping $\mathbb{N}$ adjacent voxels into one voxel. The constructed tree $\boldsymbol{\tau}^{\tilde{X}}$ is defined by (1) the respective sets $\boldsymbol{P}_l^{\tilde{X}}$, each containing $N_l^{\tilde{X}}$ points across different layers $l = 1, 2, ..., L_\tau$; (2) the coarse-to-dense indices $\boldsymbol{\rho}_{c\to d}^{\tilde{X}}$, which denote the indices of the child points corresponding to each coarse point; and (3) the dense-to-coarse indices $\boldsymbol{\rho}_{d\to c}^{\tilde{X}}$, indicating the parent point index for each dense point. Here, $c = 1, 2, ..., L_\tau - 1$ and $d = c+1$. The coordinates $\boldsymbol{C}_c^{\tilde{X}} \in \mathbb{R}^{N_c^{\tilde{X}}\times 3}$ of the coarse points are obtained by averaging the coordinates $\boldsymbol{C}_d^{\tilde{X}}$ of the child points. Specifically, the coordinates $\boldsymbol{C}_c^{\tilde{X}i}$ of the $i$th coarse point $\boldsymbol{P}_c^{\tilde{X}i}$ are obtained:

$$\boldsymbol{C}_c^{\tilde{X}i} = \frac{1}{|\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}|} \sum_{j\in\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}} \boldsymbol{C}_d^{\tilde{X}j}, \tag{3}$$

where $\boldsymbol{C}_{L_\tau}^{\tilde{X}} = \tilde{X}$ and $|\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}|$ is the cardinality of $\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}$.

**Feature pooling**. To aggregate information and build feature trees $\boldsymbol{F}_l^{\tilde{X}} \in \mathbb{R}^{N_l^{\tilde{X}}\times D}$ and $\boldsymbol{F}_l^{\tilde{Y}} \in \mathbb{R}^{N_l^{\tilde{Y}}\times D}$ ($l = 1, 2, ..., L_\tau$), the input features are initially utilized in the densest layer, and then the features are hierarchically aggregated to the corresponding parent points in a dense-to-coarse manner. We hypothesize that the contributions of child points to their parent points are related to their relative positions. To adaptively recalibrate the pointwise features based on their contributions, the dense features are concatenated along with their relative positions and then projected using a two-layer MLP, which comprises two fully connected layers and rectified linear unit (ReLU) activation. The features $\boldsymbol{F}_c^{\tilde{X}i}$ for the $i$th coarse point $\boldsymbol{P}_c^{\tilde{X}i}$ are given by

$$\boldsymbol{F}_c^{\tilde{X}i} = \frac{1}{|\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}|} \sum_{j\in\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}} \mathrm{MLP}(\mathrm{Concat}(\boldsymbol{F}_d^{\tilde{X}j}, \boldsymbol{C}_d^{\tilde{X}j} - \boldsymbol{C}_c^{\tilde{X}i})). \tag{4}$$

By aggregating information, the feature trees $\boldsymbol{F}_l^{\tilde{X}}$ and $\boldsymbol{F}_l^{\tilde{Y}}$ ($l = 1, 2, ..., L_\tau$) are built, where the level of semantic information increases from dense to coarse.

**Positional encoding**. To succinctly and efficiently integrate positional information into each layer of feature trees, we capitalize on the fact that both the coordinates and features of the coarse points are aggregated from the dense points through similar processes. Consequently, sinusoidal positional encoding [34] is directly integrated into the densest features before feature pooling.

**Tree-based Attention**. To capture important local structures and reduce computational complexity, TrA is introduced to progressively specify the attended regions and structurize the point clouds. Consider a general case in which feature trees $\boldsymbol{F}_l^{\tilde{X}}$ and $\boldsymbol{F}_l^{\tilde{Y}}$ of two different point clouds are given, $l = 1, 2, ..., L_\tau$. TrA starts from the coarsest layer and performs global attention $\mathrm{MA}(\boldsymbol{F}_1^{\tilde{X}}, \boldsymbol{F}_1^{\tilde{Y}})$, $\mathrm{MA}(\boldsymbol{F}_1^{\tilde{Y}}, \boldsymbol{F}_1^{\tilde{X}})$, which obtains the averaged attention maps $\mathbb{M}_1^{\tilde{X}} \in \mathbb{R}^{N_1^{\tilde{X}} \times N_1^{\tilde{Y}}}$, $\mathbb{M}_1^{\tilde{Y}} \in \mathbb{R}^{N_1^{\tilde{Y}} \times N_1^{\tilde{X}}}$, along with extracted features $\boldsymbol{\Phi}_1^{\tilde{X}} \in \mathbb{R}^{N_1^{\tilde{X}} \times D}$, $\boldsymbol{\Phi}_1^{\tilde{Y}} \in \mathbb{R}^{N_1^{\tilde{Y}} \times D}$. In the next layer, TrA incorporates the extracted features $\boldsymbol{\Phi}_1^{\tilde{X}}$, $\boldsymbol{\Phi}_1^{\tilde{Y}}$ to guide the feature extraction process, and specifies the attended regions based on the attention maps $\mathbb{M}_1^{\tilde{X}}$, $\mathbb{M}_1^{\tilde{Y}}$, then computes the attention within the specified regions. This process is iteratively repeated until the densest layer is reached, utilizing shared parameters. Ultimately, it yields features $\boldsymbol{\Phi}_{L_\tau}^{\tilde{X}}$ and $\boldsymbol{\Phi}_{L_\tau}^{\tilde{Y}}$ with mutual information.

To better illustrate TrA, we consider any two consecutive layers, denoted as the coarse layer $c$ and the dense layer $d$ ($d = c + 1$), and elaborate on the process within the dense layer. Given the averaged attention maps $\mathbb{M}_c^{\tilde{X}} \in \mathbb{R}^{N_c^{\tilde{X}} \times \mathcal{K}_c^{\tilde{Y}}}$, $\mathbb{M}_c^{\tilde{Y}} \in \mathbb{R}^{N_c^{\tilde{Y}} \times \mathcal{K}_c^{\tilde{X}}}$ and the extracted features $\boldsymbol{\Phi}_c^{\tilde{X}} \in \mathbb{R}^{N_c^{\tilde{X}} \times D}$, $\boldsymbol{\Phi}_c^{\tilde{Y}} \in \mathbb{R}^{N_c^{\tilde{Y}} \times D}$ from the coarse layer, $\mathcal{K}$ denotes the number of keys that each query attends to. The extracted features $\boldsymbol{\Phi}_c^{\tilde{X}}$ containing high-level semantics from the coarse layer are incorporated into the dense features $\boldsymbol{F}_d^{\tilde{X}}$. This facilitates local feature extraction and multiscale information aggregation. The incorporated features $\boldsymbol{\Psi}_d^{\tilde{X}i}$ for the $i$th dense point are obtained as

$$\boldsymbol{\Psi}_d^{\tilde{X}i} = \boldsymbol{F}_d^{\tilde{X}i} + \boldsymbol{\Phi}_c^{\tilde{X}j}, j = \boldsymbol{\rho}_{d\to c}^{\tilde{X}i}. \tag{5}$$

The features $\boldsymbol{\Psi}_d^{\tilde{Y}}$ are obtained via a similar procedure. Subsequently, the attention map $\mathbb{M}_c^{\tilde{X}}$ is utilized to specify the attended regions. For the $i$th query in the coarse layer, the $\mathcal{S}$ key points with the highest attention scores are selected, and their indices are denoted by $\boldsymbol{J}^{\tilde{X}i}$. Then, the child points of the $\mathcal{S}$ selected coarse keys constitute the attended regions of the query child points in the dense layer, forming the key points $\boldsymbol{K}_d^{\tilde{Y}} \in \mathbb{R}^{N_d^{\tilde{X}} \times \mathcal{K}_d^{\tilde{Y}} \times D}$ within the attended regions:

$$\begin{aligned} \boldsymbol{\mathcal{I}} &= \{(\boldsymbol{\rho}_{c\to d}^{\tilde{X}i}, \boldsymbol{\rho}_{c\to d}^{\tilde{Y}j}) | i \in [1, ..., N_c^{\tilde{X}}], j \in \boldsymbol{J}^{\tilde{X}i}\}, \\ \boldsymbol{K}_d^{\tilde{Y}} &= [\boldsymbol{K}_d^1, ..., \boldsymbol{K}_d^{|\mathcal{I}|}], \boldsymbol{K}_d^i = \boldsymbol{\Psi}_d^{\tilde{Y}j}, (i, j) \in \boldsymbol{\mathcal{I}}, \end{aligned} \tag{6}$$

where the first elements of $\boldsymbol{\mathcal{I}}$ represent the child point indices of the coarse queries, and the second elements indicate child point indices of the $\mathcal{S}$ selected coarse keys. $\boldsymbol{K}_d^i$ represents the key points within the attended regions of the $i$th dense query point $\boldsymbol{\Psi}_d^{\tilde{X}i}$. Such that the child points of the coarse queries only attend to the child points of the selected $\mathcal{S}$ coarse keys. Then, TrA performs attention to assemble the information of the key points located within the attended regions:

$$\boldsymbol{\Phi}_d^{\tilde{X}}, \mathbb{M}_d^{\tilde{X}} = \mathrm{MA}(\boldsymbol{\Psi}_d^{\tilde{X}}, \boldsymbol{K}_d^{\tilde{Y}}), \tag{7}$$

where MA is the attention operation described in Eq. 2 and $\boldsymbol{\Phi}_d^{\tilde{X}}$ denotes the extracted features. The attention map $\mathbb{M}_d^{\tilde{X}} \in \mathbb{R}^{N_d^{\tilde{X}} \times \mathcal{K}_d^{\tilde{Y}}}$ is generated by averaging the attention maps across all heads. By utilizing the formulated key points $\boldsymbol{K}_d^{\tilde{Y}}$, TrA establishes associations only within the specified areas, allowing it to focus on critical structures.

Similarly, the procedure for obtaining the features $\boldsymbol{\Phi}_d^{\tilde{Y}} \in \mathbb{R}^{N_d^{\tilde{Y}} \times D}$ and the averaged attention map $\mathbb{M}_d^{\tilde{Y}} \in \mathbb{R}^{N_d^{\tilde{Y}} \times \mathcal{K}_d^{\tilde{X}}}$ is performed in parallel. Finally, the output values $\boldsymbol{\Phi}_l^{\tilde{X}}$ and $\boldsymbol{\Phi}_l^{\tilde{Y}}$ from all layers are obtained. The features $\boldsymbol{\Phi}_{L_\tau}^{\tilde{X}}$ and $\boldsymbol{\Phi}_{L_\tau}^{\tilde{Y}}$ from the densest layer serve as the final outputs of TrA.

By utilizing the defined cross-attention formulation based on TrA, self-attention can be explicitly defined. Taking $\tilde{\boldsymbol{X}}$ as an example, TrA starts with the global attention operation $\mathrm{MA}(\boldsymbol{F}_1^{\tilde{X}}, \boldsymbol{F}_1^{\tilde{X}})$ in the first layer. In the subsequent layers, TrA incorporates the spatially coarse features $\boldsymbol{\Phi}_c^{\tilde{X}}$ to obtain $\boldsymbol{\Psi}_d^{\tilde{X}}$ and specifies the attended regions according to the attention maps $\mathbb{M}_c^{\tilde{X}}$ in the previous layer. Then, TrA generates the formulated key points $\boldsymbol{K}_d^{\tilde{X}}$ and performs the attention $\mathrm{MA}(\boldsymbol{\Psi}_d^{\tilde{X}}, \boldsymbol{K}_d^{\tilde{X}})$ to obtain $\boldsymbol{\Phi}_d^{\tilde{X}}$ and $\mathbb{M}_d^{\tilde{X}}$.

In general, TrA incorporates coarse features to facilitate local structure modeling and information aggregation. Furthermore, the dynamic attention sparsity of TrA enables each query to be evaluated using only highly relevant key points, reducing the computational complexity. For instance, in the case of self-attention, the computational complexity of TrA is derived as the number of floating point operations (FLOPs) required for the coarsest layer (${N_1}^2 D$) and the other layers ($\sum_{l=2}^{L_\tau} N_l \mathcal{K}_l D \leqslant \sum_{l=2}^{L_\tau} N_l \mathcal{K}_{max} D$, where $\mathcal{K}_{max}$ is a constant). The feature tree is constructed such that ${N_1}^2$ is less than a predefined constant that is independent of the number of points in the point cloud. Therefore, the complexity is linear with respect to the number of points. A direct exhibition is presented in Sec. IV-E.

The tree transformer encoder updates the features with contextual information in a series of self- and cross-attention, extracting the conditioned features $\boldsymbol{\mathcal{F}}^{\tilde{X}}$ and $\boldsymbol{\mathcal{F}}^{\tilde{Y}}$.

### *E. Decoder*

Inspired by [36], the conditioned features $\boldsymbol{\mathcal{F}}^{\tilde{X}}$ and $\boldsymbol{\mathcal{F}}^{\tilde{Y}}$ are employed to generate corresponding point clouds by means of a two-layer MLP. Specifically, the corresponding point clouds $\hat{\boldsymbol{Y}} \in \mathbb{R}^{M' \times 3}$ and $\hat{\boldsymbol{X}} \in \mathbb{R}^{N' \times 3}$ of point clouds $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{Y}}$, respectively, are predicted as

$$\begin{aligned} \hat{\boldsymbol{Y}} &= \mathrm{ReLU}(\boldsymbol{\mathcal{F}}^{\tilde{X}} \boldsymbol{W}_1 + \boldsymbol{b}_1) \boldsymbol{W}_2 + \boldsymbol{b}_2, \\ \hat{\boldsymbol{X}} &= \mathrm{ReLU}(\boldsymbol{\mathcal{F}}^{\tilde{Y}} \boldsymbol{W}_1 + \boldsymbol{b}_1) \boldsymbol{W}_2 + \boldsymbol{b}_2, \end{aligned} \tag{8}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{b}_1, \boldsymbol{b}_2$ are the learnable parameters in the MLP. Furthermore, to predict the probabilities that points lie in the overlap regions, the overlap scores $\hat{\boldsymbol{o}}^{\tilde{X}} \in \mathbb{R}^{M' \times 1}$ and $\hat{\boldsymbol{o}}^{\tilde{Y}} \in \mathbb{R}^{N' \times 1}$ are generated by a single fully connected layer (FC) and the sigmoid activation, as follows:

$$\hat{\boldsymbol{o}}^{\tilde{X}} = \mathrm{Sigmoid}(\mathrm{FC}(\boldsymbol{\mathcal{F}}^{\tilde{X}})), \hat{\boldsymbol{o}}^{\tilde{Y}} = \mathrm{Sigmoid}(\mathrm{FC}(\boldsymbol{\mathcal{F}}^{\tilde{Y}})). \tag{9}$$

### *F. Loss Functions*

Our method is trained with three loss functions: an overlap loss $\mathcal{L}_o$, a correspondence loss $\mathcal{L}_c$, and a feature loss $\mathcal{L}_f$. By introducing coefficients $\lambda_c$ and $\lambda_f$, the final loss function is

constructed and formulated as

$$\mathcal{L} = \mathcal{L}_o + \lambda_c \mathcal{L}_c + \lambda_f \mathcal{L}_f. \tag{10}$$

**Overlap loss**. $\mathcal{L}_o$ measures the consistency between the ground-truth overlap labels $\boldsymbol{o}^{\tilde{X}}$, $\boldsymbol{o}^{\tilde{Y}}$ and the predicted overlap scores $\hat{\boldsymbol{o}}^{\tilde{X}}$, $\hat{\boldsymbol{o}}^{\tilde{Y}}$. $\mathcal{L}_o = \mathcal{L}_o^X + \mathcal{L}_o^Y$, and $\mathcal{L}_o^X$ is defined as

$$\mathcal{L}_o^X = \frac{-1}{M'} \sum_{i=1}^{M'} [\boldsymbol{o}^{\tilde{X}_i} \times \log \hat{\boldsymbol{o}}^{\tilde{X}_i} + (1 - \boldsymbol{o}^{\tilde{X}_i}) \times \log(1 - \hat{\boldsymbol{o}}^{\tilde{X}_i})]. \tag{11}$$

The overlap labels $\boldsymbol{o}^{\tilde{X}}$, $\boldsymbol{o}^{\tilde{Y}}$ are obtained by downsampling the overlap labels $\boldsymbol{o}^X$, $\boldsymbol{o}^Y$ of point clouds $\boldsymbol{X}$, $\boldsymbol{Y}$, where $\boldsymbol{o}^X$, $\boldsymbol{o}^Y$ are computed by setting a threshold $r_o$ for the closest point distances between the aligned point clouds.

**Correspondence loss**. $\mathcal{L}_c$ measures the correctness of the predicted corresponding point clouds in the overlapping regions based on the $\ell^1$ loss. $\mathcal{L}_c = \mathcal{L}_c^X + \mathcal{L}_c^Y$, with the correspondence loss $\mathcal{L}_c^X$ defined as

$$\mathcal{L}_c^X = \frac{1}{\sum_{i=1}^{M'} \boldsymbol{o}^{\tilde{X}_i}} \sum_{i=1}^{M'} \boldsymbol{o}^{\tilde{X}_i} \left| \boldsymbol{T}_X^Y \left( \tilde{X}_i \right) - \hat{Y}_i \right|, \tag{12}$$

where $\boldsymbol{T}_X^Y$ is the ground-truth transformation from $\boldsymbol{X}$ to $\boldsymbol{Y}$.

**Feature loss**. $\mathcal{L}_f$ measures the discriminative power of the extracted features based on the InfoNCE loss [27]. $\mathcal{L}_f = \mathcal{L}_f^X + \mathcal{L}_f^Y$, with $\mathcal{L}_f^X$ defined as

$$\mathcal{L}_f^X = -\mathbb{E}_{x \in \mathcal{X}} \left[ \log \frac{f(x, p_x)}{f(x, p_x) + \sum_{n_x} f(x, n_x)} \right], \tag{13}$$
$$f(x, c) = \exp(\boldsymbol{\mathcal{F}}^{x T} W_f \boldsymbol{\mathcal{F}}^c),$$

where $\mathcal{X}$ denotes the set of points $\mathcal{X} \subseteq \tilde{\boldsymbol{X}}$ with a correspondence in $\tilde{\boldsymbol{Y}}$; $\boldsymbol{\mathcal{F}}^x$ indicates the extracted features for point $x$. $p_x$ and $n_x$ denote the positive and negative points in $\tilde{\boldsymbol{Y}}$, which are selected based on the positive and negative margins $(r_p, r_n)$; and $W_f$ is a learnable linear transformation.

## IV. Experimental Results

### A. Implementation Details

For the loss functions, $\lambda_c$ is set to 1, $\lambda_f$ is set to 0.1, and $(r_p, r_n)$ are set to $(m, 2m)$, where $m$ is the voxel distance used in the final downsampling layer of the KPConv backbone. The TrT is trained using AdamW [25] with an initial learning rate of $1e{-}4$ and a weight decay of $1e{-}4$; furthermore, a multistep learning rate schedule is utilized.

### B. Registration Performance on 3DMatch

**3DMatch**. The 3DMatch dataset is a real-world registration dataset, in which 46 scenes are designed for training, and the remaining 16 scenes are evenly allocated for validation and testing. The comparison methods are evaluated on both the 3DMatch ($> 30\%$ overlap) [40] and 3DLoMatch ($10-30\%$ overlap) [17] benchmarks.

**Evaluation metrics**. The methods are evaluated through various performance metrics, as per [36]. These include the *relative rotation error* RRE , *relative translation error* RTE , and *registration recall* RR (the percentage of successful alignments, where a correspondence with a root-mean-square-error below 0.2 m is considered successful).

TABLE I

Performance on the 3DMatch and 3DLoMatch benchmarks. The RRE is given in °, the RTE in $m$, and the RR in %. The three best results are highlighted in **red**, **green**, and **blue**.

| Method | Reference | 3DMatch | | | 3DLoMatch | | |
|---|---|---|---|---|---|---|---|
| | | RRE | RTE | RR | RRE | RTE | RR |
| 3DSN [16] | CVPR 2019 | 2.19 | 0.071 | 78.4 | 3.52 | 0.103 | 33.0 |
| FCGF [12] | CVPR 2019 | 2.14 | 0.070 | 85.1 | 3.74 | 0.100 | 40.1 |
| CG-SAC [30] | T-GE 2020 | 2.42 | 0.076 | 87.5 | 3.86 | 0.109 | 64.0 |
| D3Feat [4] | CVPR 2020 | 2.16 | 0.067 | 81.6 | 3.36 | 0.103 | 37.2 |
| DGR [11] | CVPR 2020 | 2.10 | 0.067 | 85.3 | 3.95 | 0.113 | 48.7 |
| DHVR [21] | ICCV 2021 | 2.25 | 0.078 | 91.9 | 4.97 | 0.123 | 65.4 |
| Predator [17] | CVPR 2021 | 2.02 | 0.064 | 89.0 | 3.04 | 0.093 | 62.5 |
| CoFiNet [37] | Neurips 2021 | 2.44 | 0.067 | 89.3 | 5.44 | 0.155 | 67.5 |
| RegTR [36] | CVPR 2022 | **1.57** | **0.049** | 92.0 | **2.83** | **0.077** | 64.8 |
| Lepard [22] | CVPR 2022 | 2.48 | 0.072 | **93.5** | 4.10 | 0.108 | 69.0 |
| SC$^2$PCR [9] | CVPR 2022 | 2.08 | 0.065 | 93.3 | 3.46 | 0.096 | 69.5 |
| GeoTR [29] | CVPR 2022 | **1.72** | 0.062 | 92.0 | **2.93** | **0.089** | **75.0** |
| VBReg [18] | CVPR 2023 | 2.04 | 0.065 | **93.5** | 3.48 | 0.096 | 69.9 |
| BUFFER [1] | CVPR 2023 | 1.85 | **0.059** | 93.2 | 3.09 | 0.101 | **71.8** |
| Ours | - | **1.49** | **0.043** | **95.4** | **2.26** | **0.067** | **76.3** |

The TrT is compared with the latest approaches: RegTR [36], Lepard [22], SC$^2$PCR [9], GeoTransformer (GeoTR) [29], VBReg [18], and BUFFER [1]. Furthermore, other comparison methods include 3DSN [16], FCGF [12], CG-SAC [30], D3Feat [4], DGR [11], DHVR [21], Predator [17], and CoFiNet [37]. The quantitative comparisons are summarized in Table I. The results show that our method precisely aligns a pair of real-world point clouds even at low overlap rates and outperforms the other methods on both 3DMatch and 3DLoMatch. These results verify that the learned attended regions and the guidance of the coarse features facilitate local feature extraction, enabling our method to precisely align point clouds.

### C. Registration Performance on KITTI

**KITTI**. To demonstrate the performance of our method on a large-scale point cloud dataset, the TrT and the baseline methods are evaluated on the KITTI [15] dataset. The KITTI dataset contains 11 sequences of LiDAR-scanned outdoor driving scenarios, of which scenarios 0–5 are used for training, 6–7 are used for validation, and 8–10 are used for testing. Following [17], only point cloud pairs that are at most $10m$ away from each other are utilized for evaluation.

**Evaluation metrics**. Following [36], the performance of each method is evaluated using the RRE, RTE, and RR (the percentage of successful alignments, whose RRE and RTE values are below $5^\circ$ and $2m$, respectively).

The TrT is compared with the latest approaches SC$^2$PCR [9], GeoTransformer (GeoTR) [29], and MAC [41]; the baseline methods also include DGR [11], D3Feat [4], HRegNet [26], SpinNet [2], Predator [17], and CoFiNet [37]. The quantitative comparisons are summarized in Table III. The results indicate that our method achieves the highest accuracy on the KITTI benchmark. The GeoTR approach exhibits the second-best performance, following our method; however, its inference time is more than ten times that of ours, as evidenced in Table II.

TABLE II

COMPUTATIONAL TIME IN SECONDS ON THE 3DMATCH BENCHMARK.

| 3DSN | FCGF | CG-SAC | D3Feat | DGR | DHVR | Predator | CoFiNet | RegTR | Lepard | SC$^2$PCR | GeoTR | VBReg | BUFFER | Ours |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30.234 | 1.562 | 0.263 | 0.916 | 1.741 | 3.43 | 1.572 | 1.134 | 0.103 | 0.522 | 0.380 | 1.523 | 0.223 | 0.196 | 0.123 |

TABLE III

PERFORMANCE ON THE KITTI BENCHMARK.THE THREE BEST RESULTS ARE HIGHLIGHTED IN **RED**, **GREEN**, AND **BLUE**.

| Method | Reference | RRE(°) | RTE(m) | RR(%) |
|---|---|---|---|---|
| DGR [11] | CVPR 2020 | 0.37 | 0.320 | 98.7 |
| D3Feat [4] | CVPR 2020 | 0.30 | 0.072 | **99.8** |
| HRegNet [26] | ICCV 2021 | 0.29 | 0.120 | 99.7 |
| SpinNet [2] | CVPR 2021 | 0.47 | 0.099 | 99.1 |
| Predator [17] | CVPR 2021 | 0.28 | **0.068** | **99.8** |
| CoFiNet [37] | Neurips 2021 | 0.41 | 0.082 | **99.8** |
| SC$^2$PCR [9] | CVPR 2022 | 0.32 | 0.072 | 99.6 |
| GeoTR [29] | CVPR 2022 | **0.24** | **0.068** | **99.8** |
| MAC [41] | CVPR 2023 | 0.40 | 0.084 | 99.5 |
| RegFormer [23] | ICCV 2023 | **0.24** | 0.084 | **99.8** |
| Ours | - | **0.23** | **0.063** | **99.8** |

TABLE IV

ABLATION RESULTS ON THE 3DMATCH AND 3DLOMATCH CONCERNING THE EFFECTS OF THE DIFFERENT MODEL COMPONENTS. THE RR IS GIVEN IN %, THE RRE IN °, AND THE RTE IN $m$.

| Method | 3DMatch | | | 3DLoMatch | | |
|---|---|---|---|---|---|---|
| | RR | RRE | RTE | RR | RRE | RTE |
| TrT$_{ST}$ | 92.3 | 1.53 | 0.048 | 66.9 | 2.67 | 0.078 |
| TrT$_{fixed}$ | 93.1 | 1.61 | 0.047 | 70.4 | 2.77 | 0.080 |
| TrT$_{fixed_shift}$ | 93.3 | 1.65 | 0.053 | 69.0 | 2.83 | 0.084 |
| TrT$_{w/o\ coarse_info}$ | 91.2 | 1.75 | 0.054 | 64.4 | 3.45 | 0.098 |
| TrT | **95.4** | **1.49** | **0.043** | **76.3** | **2.26** | **0.067** |

### D. Ablation Studies

To analyze the effectiveness of the proposed TrT, ablation studies are carried out on the 3DMatch and 3DLoMatch benchmarks. The results are shown in Table IV.

**Comparison with the standard transformer**. TrT$_{ST}$ is obtained by replacing the TrA module with the standard attention mechanism [34], resulting in lower RR values on both the 3DMatch and 3DLoMatch benchmarks. These results indicate that TrA enhances the local structure modeling capability, enabling the extraction of rich local information and yielding improved registration accuracy.

**Learned attention sparsity**. TrT$_{fixed}$ and TrT$_{fixed_shift}$ utilize a fixed pattern for attention mechanism. Concretely, TrT$_{fixed}$ specifies the attended regions as the child points of the $\mathcal{S}$ nearest coarse points. TrT$_{fixed_shift}$ additionally utilizes a window shifting operation [24]. Both variants exhibit significant performance degradation, demonstrating the enhancement in local structure modeling capability achieved through our learned attention sparsity.

**Coarse feature guidance**. TrT$_{w/o\ coarse_info}$ computes the attention without the guidance of the coarse features, resulting in decreased RR values on 3DMatch and 3DLoMatch. These results demonstrate that the guidance provided by the coarse features enhances the local feature extraction.

### E. Efficiency Evaluation

To facilitate a direct comparison with the standard attention mechanism (SA) [34], we provide a visualization of inference time and memory usage for both SA and our proposed TrA in Figure 2. As the number of points increases, both the inference time and memory usage of SA grow quadratically, whereas those of TrA increase linearly. Additionally, the inference time of the comparison methods is evaluated using the 3DMatch [40] benchmark, and the results are summarized in Table II. TrT emerges as the second fastest method, with an inference time under 130 ms.

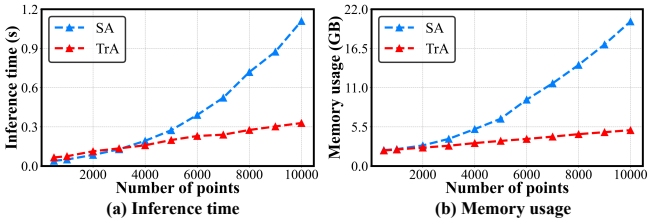


Fig. 2. Comparison of the inference time and memory usage between ST and our TrA, where ST utilizes the standard attention mechanism.

## V. LIMITATION

Since the TrT is only implemented with a naive CUDA kernel without many optimizations, and it is not as efficient as the well-optimized GPU matrix operation. In the future, the inference time of TrT can be further reduced by optimizing the CUDA kernel.

## VI. CONCLUSION

In this work, a novel transformer-based network named the TrT is proposed. This approach can extract abundant local and global information while maintaining linear computational complexity. Our approach builds coarse-to-dense feature trees, and the proposed TrA module follows tree structures to progressively narrow the attended regions and structurize point clouds. Specifically, the coarse layers adaptively specify the attended regions in the dense layers, and the extracted coarse features are incorporated into the dense layers to guide the feature extraction process, thereby enabling our method to focus on important local structures and facilitating local feature extraction. Experiments demonstrate that our method achieves state-of-the-art performance on 3DMatch and KITTI benchmarks.

## REFERENCES


[1] Sheng Ao, Qingyong Hu, Hanyun Wang, Kai Xu, and Yulan Guo. Buffer: Balancing accuracy, efficiency, and generalizability in point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1255–1264, 2023. 5

[2] Sheng Ao, Qingyong Hu, Bo Yang, Andrew Markham, and Yulan Guo. Spinnet: Learning a general surface descriptor for 3d point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11753–11762, 2021. 5, 6

[3] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019. 1

[4] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6359–6367, 2020. 5, 6

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 1

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 2

[7] Guangyan Chen, Meiling Wang, Qingxiang Zhang, Li Yuan, Tong Liu, and Yufeng Yue. Deep interactive full transformer framework for point cloud registration. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2825–2832. IEEE, 2023. 1

[8] Guangyan Chen, Meiling Wang, Qingxiang Zhang, Li Yuan, and Yufeng Yue. Full transformer framework for robust point cloud registration with deep information interaction. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 1, 2

[9] Zhi Chen, Kun Sun, Fan Yang, and Wenbing Tao. Sc2-pcr: A second order spatial compatibility for efficient and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13221–13231, 2022. 5, 6

[10] Zhang Cheng, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Patchformer: A versatile 3d transformer based on patch attention. *arXiv preprint arXiv:2111.00207*, 2021. 1, 2

[11] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2514–2523, 2020. 1, 5, 6

[12] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8958–8966, 2019. 5

[13] Zhaoxin Fan, Zhenbo Song, Hongyan Liu, Zhiwu Lu, Jun He, and Xiaoyong Du. Svt-net: Super light-weight sparse voxel transformer for large scale place recognition. AAAI, 2022. 2

[14] Kexue Fu, Shaolei Liu, Xiaoyuan Luo, and Manning Wang. Robust point cloud registration framework based on deep graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8893–8902, 2021. 1, 2

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 5

[16] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5545–5554, 2019. 5

[17] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4267–4276, 2021. 2, 3, 5, 6

[18] Haobo Jiang, Zheng Dang, Zhen Wei, Jin Xie, Jian Yang, and Mathieu Salzmann. Robust outlier rejection for 3d registration with variational bayes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1148–1157, 2023. 5

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 2

[20] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022. 1, 2

[21] Junha Lee, Seungwook Kim, Minsu Cho, and Jaesik Park. Deep hough voting for robust global registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15994–16003, 2021. 5

[22] Yang Li and Tatsuya Harada. Lepard: Learning partial point cloud matching in rigid and deformable scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5554–5564, 2022. 5

[23] Jiuming Liu, Guangming Wang, Zhe Liu, Chaokang Jiang, Marc Pollefeys, and Hesheng Wang. Regformer: An efficient projection-aware transformer network for large-scale point cloud registration. *arXiv preprint arXiv:2303.12384*, 2023. 2, 6

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 6

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[26] Fan Lu, Guang Chen, Yinlong Liu, Lijun Zhang, Sanqing Qu, Shu Liu, and Rongqi Gu. Hregnet: A hierarchical network for large-scale outdoor lidar point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16014–16023, 2021. 5, 6

[27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 5

[28] Anh Viet Phan, Minh Le Nguyen, Yen Lam Hoang Nguyen, and Lam Thu Bui. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108:533–543, 2018. 2

[29] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11143–11152, 2022. 2, 5, 6

[30] Siwen Quan and Jiaqi Yang. Compatibility-guided sampling consensus for 3-d point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, 58(10):7380–7392, 2020. 5

[31] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 1

[32] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. 1

[33] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 2

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 3, 6

[35] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019. 1, 2

[36] Zi Jian Yew and Gim Hee Lee. Regtr: End-to-end point cloud correspondences with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6677–6686, 2022. 2, 4, 5

[37] Hao Yu, Fu Li, Mahdi Saleh, Benjamin Busam, and Slobodan Ilic. Cofinet: Reliable coarse-to-fine correspondences for robust pointcloud registration. *Advances in Neural Information Processing Systems*, 34:23872–23884, 2021. 2, 5, 6

[38] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 2

[39] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition, 2021. 2

[40] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017. 5, 6

[41] Xiyu Zhang, Jiaqi Yang, Shikun Zhang, and Yanning Zhang. 3d

registration with maximal cliques. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17745–17754, 2023. 5, 6

[42] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 1, 2

[43] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European conference on computer vision*, pages 766–782. Springer, 2016. 1