HyperGCT: A Dynamic Hyper-GNN-Learned Geometric Constraint for 3D Registration

Xiyu Zhang[1], Jiayi Ma[2], Jianwei Guo[3], Wei Hu[4], Zhaoshuai Qi[1], Fei Hui[5], Jiaqi Yang[1]*, Yanning Zhang[1]
[1]Northwestern Polytechnical University [2]Wuhan University [3]Chinese Academy of Sciences
[4]Peking University [5]Chang'an University
{2426988253}@mail.nwpu.edu.cn; {ynzhang, jqyang}@nwpu.edu.cn

Abstract

*Geometric constraints between feature matches are critical in 3D point cloud registration problems. Existing approaches typically model unordered matches as a consistency graph and sample consistent matches to generate hypotheses. However, explicit graph construction introduces noise, posing great challenges for handcrafted geometric constraints to render consistency. To overcome this, we propose HyperGCT, a flexible dynamic **Hyper**-**G**NN-learned geometric **C**onstrain**T** that leverages high-order consistency among 3D correspondences. To our knowledge, HyperGCT is the first method that mines robust geometric constraints from dynamic hypergraphs for 3D registration. By dynamically optimizing the hypergraph through vertex and edge feature aggregation, HyperGCT effectively captures the correlations among correspondences, leading to accurate hypothesis generation. Extensive experiments on 3DMatch, 3DLoMatch, KITTI-LC, and ETH show that HyperGCT achieves state-of-the-art performance. Furthermore, HyperGCT is robust to graph noise, demonstrating a significant advantage in terms of generalization.*

1. Introduction

Point cloud registration (PCR) is essential in numerous 3D computer vision tasks including reconstruction [7], simultaneous localization and mapping (SLAM) [14], and robotics [27]. It aims to align 3D fragments from different views. The standard approach involves establishing correspondences using local features, sampling consistent matches that meet geometric constraints, and estimating six-degree-of-freedom (6-DoF) pose transformations. However, PCR becomes challenging when faced with noise, such as low-quality point clouds or incorrect matches.

Correspondence-based methods are the mainstream solution to PCR and can be categorized into deep-learned and traditional approaches. Deep-learned methods [2, 9, 19, 20, 25, 37, 38, 47] classify inliers and outliers by learning the consistency between correspondences and generating inlier probabilities for outlier removal. Most traditional methods [6, 21, 28, 33, 39–41, 43, 44, 46] follow the hypothesis generation-verification paradigm, leveraging geometric constraints to select consistent matches. Thus, designing a good geometric constraint is critical for both methods. Deep-learned methods integrate pairwise compatibility into networks, for example, PointDSC [2] using spatial consistency and 3DPCP-Net [37] incorporating deep geometric coherence. Traditional methods employ random sampling, triplet, or clique constraints. RANSAC [16] and its variants [3, 4, 10, 24] use random sampling to remove outliers, but their performance degrades with a high outlier ratio. Recently, researchers [21, 28, 33, 39–41, 43, 44, 46] have applied graph theory to identify geometrically consistent matches. These approaches typically check the geometric

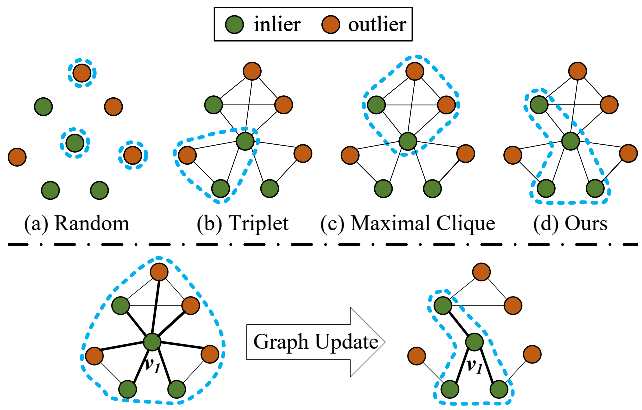

Figure 1. Key aspects of HyperGCT. It mines high-order consistency among inliers, offering greater flexibility than random sampling, triplet, or clique constraints. The implicit graph construction continuously refines the neighbor connections around the central node, effectively learning the high-order correlations.

*Corresponding Author.
Code link: https://github.com/zhangxy0517/HyperGCT.

consistency with empirical thresholds between correspondences before explicitly constructing a graph to represent the relationships. SAC-COT [41] leverages triplet sampling to accelerate the process and enhance efficiency and accuracy. Assuming that inliers form a maximum clique, TEASER [40] uses a parallel maximum clique finder algorithm, PMC [30], to identify this clique and filter out outliers. MAC [46] relaxes the maximum clique constraints by searching for maximal cliques, fully considering each local consensus. However, existing constraints have several key issues: **1)** pairwise constraint is ambiguous and struggles to associate multiple correspondences; **2)** static compatibility thresholds limit the flexibility to handle various input conditions; **3)** explicit graph construction is sensitive to graph noise introduced by matching and measurement errors.

In this paper, to address the above problems in PCR, we propose HyperGCT. This dynamic Hyper-GNN-learned geometric constraint leverages high-order consistency to capture the relationships among correspondences in a latent hypergraph feature space. The motivation is that measurement and matching errors can make geometric constraints in explicit space highly vulnerable to graph noise. By representing correspondences within a hypergraph, HyperGCT naturally captures complex, higher-order relationships that are more robust to noise than standard graph-based approaches. As shown in Fig. 1, HyperGCT differs from others in several key aspects. First, unlike other methods, HyperGCT uses a hypergraph to model multiple correspondences, offering more flexibility than random sampling, triplet, or clique constraints. With implicit graph construction and continuous topology updates via Hyper-GNN [15], our approach is parameter-insensitive and robust to graph noise. Second, unlike methods focus on inlier/outlier classification, HyperGCT emphasizes learning high-order correlations among correspondences, enabling it to handle matches across different modalities (RGB-D and LiDAR), and providing stronger generalization than existing approaches. Overall, our contributions are as follows:

- We introduce HyperGCT, a novel geometric constraint for 3D correspondences that offers greater flexibility than existing constraints, *e.g.*, random sampling, triplet, and maximal cliques.
- We further propose a HyperGCT-based registration network. Our method demonstrates a significant advantage in registration performance and generalization compared to other approaches.

2. Related Works

2.1. Point Cloud Registration

Traditional Methods. The RANSAC algorithm [16] is widely employed in PCR following the hypothesis generation-verification process. Over the years, many variants have been proposed to improve sampling [4, 10, 24, 36] and verification [3, 11, 12, 35, 42]. However, due to random sampling and the lack of inlier consistency constraints, these methods struggle with efficiency and accuracy when the outlier ratio is high. To address this, researchers model the unordered correspondences as a compatibility graph and use geometric constraints to guide hypothesis generation. Some approaches [33, 39, 41, 43, 44] utilize triplet constraints to reduce the ambiguity of pairwise consistency. Assuming that inliers are mutually compatible with each other, some approaches [21, 28, 40, 46] utilize clique constraints.

Traditional methods face two key issues: they require manual parameter tuning, limiting flexibility when input conditions change, and their reliance on explicit graph construction makes them sensitive to graph noise from matching and measurement errors. In contrast, we use a hypergraph to model relationships among multiple correspondences, offering greater flexibility than other constraints. With implicit graph construction and continuous topology updates via Hyper-GNN, our method is parameter-insensitive and robust to graph noise.

Deep-learned Methods. These methods [2, 9, 19, 20, 25, 37, 38, 47] design inlier/outlier classifiers by learning consistency between correspondences. DGR [9] and 3DRegNet [25] employ sparse convolution and point-by-point MLP to train end-to-end neural networks. DHVR [20] leverages deep Hough voting to establish consensus in the Hough space. PG-Net [38] proposes a grouped dense fusion attention feature embedding module to enhance inlier representations. VBReg [19] models long-range dependencies with a variational non-local network. However, they often neglect geometric consistency constraints. PointDSC [2] incorporates spatial consistency into a Non-Local module. 3DPCP-Net [37] introduces a DGC block to explore deep feature similarity, pairwise distances, and angles. Despite these advancements, they remain limited to refining higher-order relationships beyond pairwise vertex interactions. Though Hunter [47] addresses this limitation by introducing an HCR module to learn the high-order consistency of inliers, its hypergraph structure is static and lacks flexibility.

Deep-learned methods generally outperform traditional approaches but struggle with high outlier ratios due to weak inlier consistency. Furthermore, reliance on data-specific features hinders generalization in varying pattern matching of characteristics in different datasets. In contrast, we operate within a generalized hypergraph framework to optimize the relationships between vertices and hyperedges, enabling enhanced, precise information propagation and robust modeling of higher-order relationships. Our method effectively handles diverse matching patterns while improving generalization across various scenarios.

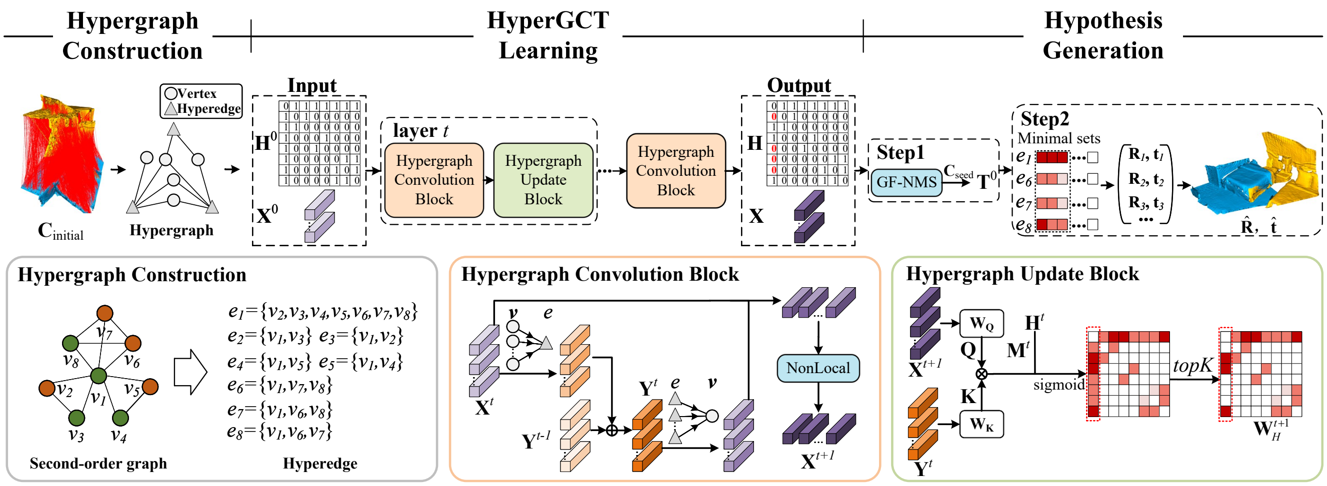

Figure 2. **The overall pipeline**. **1.** An initial hypergraph $\mathbf{H}^0$ encodes high-order consistency within $\mathbf{C}_{\text{initial}}$ **2.** Hypergraph is continuously updated through HyperGCT learning. **3.** Hypotheses generated from high-order subsets are evaluated to get the final transformation.

2.2. Hypergraph Learning for Point Clouds

A hypergraph extends graph theory by modeling complex relationships. Unlike regular graphs, where an edge connects two nodes, hypergraphs use hyperedges to connect two or more nodes, capturing higher-order relationships. This is particularly useful for high-dimensional data, such as point clouds, where higher-order correlations exist. Hypergraphs improve the model's expressiveness and accuracy, making them valuable for point cloud processing. In recent years, hypergraph learning has been applied to tasks like sampling [13, 50], denoising [50], classification [51], recognition [1], and registration [47].

3. Method

3.1. Problem Formulation

Given the source point cloud $\mathbf{P}^s = \{\mathbf{p}^s\}$ and target point cloud $\mathbf{P}^t = \{\mathbf{p}^t\}$ to be aligned, we first extract local features for them using geometric [31] or learned [8] descriptors. An initial correspondence set $\mathbf{C}_{\text{initial}} = \{\mathbf{c}\}$ is then established by nearest neighbor search in feature space, where $\mathbf{c} = (\mathbf{p}^s, \mathbf{p}^t)$. The final goal is to estimate an accurate pose transformation from $\mathbf{C}_{\text{initial}}$. To achieve this, we propose HyperGCT, designed to capture and represent high-order consistency among inliers for each correspondence. As illustrated in Fig. 2, our method comprises three stages: hypergraph construction, HyperGCT learning, and hypothesis generation. 1) *Hypergraph construction*: we build an initial hypergraph that encodes high-order consistency within $\mathbf{C}_{\text{initial}}$, allowing HyperGCT to optimize the structure. 2) *HyperGCT learning*: to avoid generalization issues from simply learning to distinguish inliers from outliers, we leverage inliers' high-order consistency to dynamically update the hypergraph structure and refine correspondence representation, which helps to improve the reliability of hypothesis generation. 3) *Hypothesis generation*: this final stage samples putative seed correspondences from the updated hypergraph, generating and evaluating multiple hypotheses. The best hypothesis is selected as the final transformation estimate.

3.2. Hypergraph Construction

A hypergraph is defined as $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{H})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of hyperedges. The incidence matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ represents hypergraph's structure, with $\mathbf{H}(v_i, e_j)$ indicating whether hyperedge e_j contains vertex v_i. The degree of hyperedge e_j and vertex v_i is defined as $\mathbf{D}(e_j) = \sum_{i=1}^{|\mathcal{V}|} \mathbf{H}(v_i, e_j)$ and $\mathbf{D}(v_i) = \sum_{j=1}^{|\mathcal{E}|} \mathbf{H}(v_i, e_j)$. Unlike an edge in a simple graph that connects only two vertices, a hyperedge connects multiple vertices, capturing high-order relationships. We model $\mathbf{C}_{\text{initial}}$ as an initial hypergraph $\mathcal{H}_0$, with each vertex v representing a correspondence $\mathbf{c}$. To construct hyperedges, we use the second-order measure [5, 46], which has a stronger constraint capability than the first-order measure (*i.e.*, pairwise measure). As illustrated in Fig. 2, we first model $\mathbf{C}_{\text{initial}}$ as a second-order graph (SOG) based on the pairwise rigid distance compatibility measure. For a correspondence pair $(\mathbf{c}_i, \mathbf{c}_j)$, the rigid distance is:

$$d_{ij} = \left| \left\| \mathbf{p}_i^s - \mathbf{p}_j^s \right\| - \left\| \mathbf{p}_i^t - \mathbf{p}_j^t \right\| \right|. \tag{1}$$

The compatibility score is:

$$\gamma_{ij} = \left[1 - \frac{{d_{ij}}^2}{{\sigma_{\mathrm{d}}}^2} \right]_+, \tag{2}$$

where σ_{d} controls the sensitivity to distance difference. $[\cdot]_+$ is the operation of $max(\cdot, 0)$. Usually, if γ_{ij} exceeds threshold θ_{cmp}, the two correspondences are considered compatible, and γ_{ij} is stored in matrix $\mathbf{W}_\gamma$. Different from methods that use static thresholds [2, 5, 41, 46], we compute θ_{cmp} in a dynamic way to handle $\mathbf{C}_{\text{initial}}$ with various inlier ratios:

$$\theta_{\text{cmp}} = \frac{1}{K_1 \left|\mathbf{C}_{\text{initial}}\right|} \sum_{i=0}^{\left|\mathbf{C}_{\text{initial}}\right|} \sum_{j=0}^{K_1} topK_1(\gamma_{ij}), \tag{3}$$

where $K_1 = 0.1 \left|\mathbf{C}_{\text{initial}}\right|$. The initial weight matrix $\mathbf{W}_H^0$ is:

$$\mathbf{W}_H^0 = \mathbf{W}_\gamma \odot (\mathbf{W}_\gamma \times \mathbf{W}_\gamma), \tag{4}$$

where $\odot$ denotes the element-wise product. Next, we define the vertices directly connected to each non-isolated vertex v_i, in the SOG as belonging to the hyperedge e_i. For vertices v_i and v_j, $\mathbf{H}(v_i, e_j)$ is set to 1 if $\mathbf{W}_H(i,j) > 0$, and the initial hypergraph $\mathbf{H}^0$ is constructed. Thus, e_i contains the high-order consistency information surrounding v_i, serving as the geometric constraint optimized in the HyperGCT learning step.

3.3. HyperGCT Learning

This step optimizes each hyperedge e_i based on initial hypergraph $\mathbf{H}^0$, ensuring it includes only the inliers. High-order consistency information is then leveraged to represent v_i accurately. Similar to traditional GNNs [22], HyperGCT employs a stacked structure of multiple convolution layers to learn feature representations progressively. Each convolution layer consists of a hypergraph convolution block, updating vertex and hyperedge features and enabling extensive information exchange. Between adjacent hypergraph convolution blocks, a hypergraph update block is introduced. This block refines the graph structure by assessing the similarity between the features of vertices and hyperedges, ensuring the structural relationships are continuously optimized. Combining the two kinds of blocks allows HyperGCT to capture high-order consistency among inliers effectively. The coordinates of correspondences are taken as the network's input, making our method applicable to different descriptors.

Hypergraph Convolution Block. The information passing in the t-th hypergraph convolution block is divided into two stages: the first stage aggregates the information of vertices within each hyperedge to update the hyperedge features from $\mathbf{Y}_{t-1}$ to $\mathbf{Y}_t$, while the second stage aggregates the hyperedge information to update the vertex features from $\mathbf{X}_t$ to $\mathbf{X}_{t+1}$. The first stage is formulated as follows:

$$Stage1 \begin{cases} \hat{\mathbf{Y}}^t = (\mathbf{D}_e^t)^{-1}(\mathbf{H}^t)^\top \mathbf{X}^t \\ \mathbf{Y}^t = \text{normalize}(MLP_1(concate(\mathbf{Y}^{t-1}, \hat{\mathbf{Y}}^t))) \end{cases}, \tag{5}$$

where $\mathbf{D}_e^t$ is the diagonal matrix of hyperedge degrees $\mathbf{D}(e)^t$. $\mathbf{Y}^t$ is obtained by merging the aggregated vertex information $\hat{\mathbf{Y}}^t$ with $\mathbf{Y}^{t-1}$ obtained from the previous block. The motivation is to enhance the feature representation by combining the local contextual information and historical feature representations, thereby more accurately capturing the high-order consistency. The second stage is formulated as follows:

$$Stage2 \begin{cases} \hat{\mathbf{X}}^{t+1} = (\mathbf{D}_v^t)^{-1} \mathbf{H}^t \mathbf{W}_{\text{e}}^t \mathbf{Y}^t \\ \mathbf{X}^{t+1} = \rho(\mathbf{X}^t + MLP_2(\hat{\mathbf{X}}^{t+1})) \\ \mathbf{X}^{t+1} = \text{normalize}(NonLocal(\mathbf{X}^{t+1}, \mathbf{W}_H^0)) \end{cases}, \tag{6}$$

where ρ represents the ReLU activation function, $\mathbf{D}_v^t$ is the diagonal matrix of vertex degrees $\mathbf{D}(v)^t$, and $\mathbf{W}_e^t$ is the diagonal matrix of hyperedge weights $\mathbf{W}(e)^t$, definded as:

$$\mathbf{W}(\text{e}_j)^t = \sum_{i=1}^{|\mathcal{V}|} \mathbf{W}_H^t(i,j). \tag{7}$$

When updating the vertex feature $\mathbf{X}^{t+1}$ using the aggregated hyperedge information $\hat{\mathbf{X}}^{t+1}$, we introduce a self-looping message passing path to prevent the vanishing of the vertex's own feature information [17]. Subsequently, the vertex feature $\mathbf{X}^{t+1}$ is passed through a NonLocal [2] block, where feature enhancement is performed using the matrix $\mathbf{W}_H^0$, which contains the global second-order compatibility information.

Hypergraph Update Block. The block first generates a mask matrix $\mathbf{M}^t$ based on $\mathbf{H}^t$, which is used to ignore vertices and hyperedges with no connections:

$$\mathbf{M}^t = (1 - \mathbf{H}^t) \cdot (-\infty). \tag{8}$$

Next, $\mathbf{H}^{t+1}$ and $\mathbf{W}_H^{t+1}$ are obtained through the graph pooling operation based on the similarity between vertex feature $\mathbf{X}^{t+1}$ and hyperedge feature $\mathbf{Y}^t$:

$$\mathbf{H}^{t+1}, \mathbf{W}_H^{t+1} = topK_2(sigmoid(\frac{(\mathbf{Q}^t)^\top \mathbf{K}^t}{\sqrt{N_{\text{channel}}}} + \mathbf{M}^t)), \tag{9}$$

where $K_2 = 0.1(5-t)\left|\mathbf{C}_{\text{initial}}\right|$, $t \in [0,4]$ and N_{channel} is the number of feature channels. $\mathbf{Q}^t$ and $\mathbf{K}^t$ are the projections of $\mathbf{X}^{t+1}$ and $\mathbf{Y}^t$. In this step, each vertex selects K_2 most similar hyperedges in feature space, ensuring that the selected hyperedges contain the vertex.

As K_2 decreases with increasing t, deeper layers allow each vertex to select fewer hyperedges. This encourages more comprehensive information exchange at shallower layers, while deeper layers focus on interactions among inliers. Because the similarity is computed between each vertex-hyperedge pair, the sigmoid activation function is applied here. The learned feature $\mathbf{X}_i$ for each correspondence and optimized incidence matrix $\mathbf{H}$ are used for predicting the inlier confidence $\hat{\mathbf{s}}$ and hypothesis generation.

3.4. Hypothesis Generation

The hypotheses are generated through guided sampling, which contains two steps. In the first step, initial hypotheses are derived from putative seeds. Leveraging both inlier confidence $\hat{\mathbf{s}}$ and the hypergraph structure $\mathbf{H}$, we propose a Graph-Filtering Non-Maximum Suppression (GF-NMS) method to retrieve more reliable correspondences as seeds $\mathbf{C}_{\text{seed}}$ compared to standard NMS [23] (refer to Supp. 6):

$$\mathbf{C}_{\text{seed}} = \text{GF-NMS}(\mathbf{H}, \hat{\mathbf{s}}, N_{\text{s}}), \tag{10}$$

where N_{s} is the number of seeds. A feature-space KNN search is then conducted to sample subsets for each correspondence in $\mathbf{C}_{\text{seed}}$. Subsequently, the SVD algorithm is applied to each subset to generate candidate hypotheses, with N_{initial} of the most promising ones retained as initial hypotheses $\mathbf{T}^0$.

Since $\mathbf{T}^0$ may not obtain local optima, the second step is introduced to explore the solution space further, guided by high-order consistency, to find the optimal hypothesis. Each $\mathbf{c}_i$ in $\mathbf{C}_{\text{seed}}$ corresponds to a hyperedge e_i in $\mathbf{H}$, which includes multiple correspondences exhibiting high-order consistency with $\mathbf{c}_i$. We compute the residuals for correspondences within e_i using the initial hypothesis $\mathbf{T}_i^0$ and sort them in ascending order based on the residual value: $r_{ij} = \left\|\mathbf{R}_i\mathbf{p}_{ij}^s + \mathbf{t}_i - \mathbf{p}_{ij}^t\right\|$, where $\mathbf{c}_{ij} = (\mathbf{p}_{ij}^s, \mathbf{p}_{ij}^t)$ represents a correspondence in e_i. For each hyperedge e, we iteratively sample minimal sets from front to back to generate hypotheses. Early-stage samples produce hypotheses similar to the initial ones, while later-stage samples generate hypotheses that diverge more significantly. This approach enables a thorough exploration of the local solution space initially, followed by the generation of more diverse hypotheses, thereby enhancing the robustness of the final pose estimation by covering a wider range of potential solutions. The final transformation is evaluated by:

$$\hat{\mathbf{R}}, \hat{\mathbf{t}} = \underset{\mathbf{R}_j, \mathbf{t}_j}{\operatorname{argmax}} \sum_{i=1}^{|\mathbf{C}_{\text{initial}}|} \phi(\left\|\mathbf{R}_j\mathbf{p}_i^s + \mathbf{t}_j - \mathbf{p}_i^t\right\|), \tag{11}$$

where ϕ is the Mean Absolute Error (MAE) function [42].

3.5. Loss Functions

To supervise HyperGCT learning, a joint loss function is proposed:

$$\mathcal{L} = \mathcal{L}_{class} + \mathcal{L}_{match} + \mathcal{L}_{graph}, \tag{12}$$

where $\mathcal{L}_{class}$ denotes the classification loss to supervise each correspondence, $\mathcal{L}_{match}$ denotes the matching loss to supervise pairs of correspondences, and $\mathcal{L}_{graph}$ denotes the graph loss to supervise the hypergraph.

For the classification loss, HyperGCT uses binary cross-entropy loss to learn the confidence scores $\hat{\mathbf{s}}$ for $\mathbf{C}_{\text{initial}}$:

$$\mathcal{L}_{class} = BCE(\hat{\mathbf{s}}, \mathbf{s}^*), \tag{13}$$

where $\mathbf{s}^*$ represents the ground truth labels of inlier/outlier: $\mathbf{s}^* = [\|\mathbf{R}^*\mathbf{p}_i^s + \mathbf{t}^* - \mathbf{p}_i^t\| < \theta_{\text{inlier}}]$.

For the match loss, a spectral matching loss [2] is used, which formulated as follows:

$$\mathcal{L}_{match} = \frac{1}{|\mathbf{C}_{\text{initial}}|^2} \sum_{ij} (\eta_{ij} - \eta_{ij}^*)^2, \tag{14}$$

where η_{ij} is the compatibility score of the pairwise correspondence features: $\eta_{ij} = \left[1 - \frac{1}{\sigma_{\text{f}}^2}\|\mathbf{X}_i - \mathbf{X}_j\|^2\right]_+$, where σ_{f} is a learnable parameter that controls the sensitivity to feature differences. The term η_{ij}^* is the ground-truth compatibility score between inliers.

The graph loss is formulated as follows:

$$\mathcal{L}_{graph} = \frac{1}{|\mathbf{C}_{\text{initial}}|} \sum_i BCE(\mathbf{W}_H(i,:), \mathbf{H}^*(i,:)), \tag{15}$$

$\mathbf{W}_H$ and $\mathbf{H}^*$ are the weight matrix output by HyperGCT and the ground truth incidence matrix formed by inliers.

4. Experiments

4.1. Experimental Setup

Dataset. We use indoor scene datasets 3DMatch/3DLoMatch [18, 49], as well as outdoor scene datasets KITTI-10m [28] and KITTI-LC [28], for training and testing. Additionally, we consider the outdoor scene dataset ETH [26] for generalization evaluation. For 3DMatch, following [2, 9], we use the provided train/validation/test split and evaluate on 1623 pairs from 8 scenes. For 3DLoMatch, following [18], 1781 low overlap pairs are used for evaluation. For KITTI-10m, we follow the split in [2, 9], with 0-5/6-7/8-10 sequences for train/val/test. For KITTI-LC, we conduct comparisons across 0-10m, 10-20m, and 20-30m translation ranges, comprising 914, 1151, and 1260 pairs, respectively. ETH contains 713 clutter and occlusion pairs from 4 scenes. Please refer to Supp. 8 for more details.

Evaluation Criteria. For all experiments, we report registration recall (RR), which measures the rate of successful pairs whose rotation error (RE) and translation error (TE) fall below specific thresholds. Following [2, 9, 45, 46], successful registration is considered when RE≤15° and TE≤30cm for 3DMatch/3DLoMatch, and ETH, and RE≤5° and TE≤60cm for KITTI-10m. For the KITTI-LC, the subsets "0-10m", "10-20m", and "20-30m" use TE thresholds of 60cm, 120cm, and 180cm, respectively, with the same RE threshold as KITTI-10m. Inlier precision (IP), inlier recall (IR), and F1-score (F1) are reported to assess the outlier removal performance.

Implementation Details. Our method is implemented in PyTorch. For 3DMatch/3DLoMatch and KITTI-10m, we generate FCGF [8] descriptors with the voxel size of 5cm

Table 1. Registration results on 3DMatch. The key metric is **RR**, as it reflects the number of successfully registered pairs and overall performance. RE and TE focus on local precision and are meaningful for comparison only when RR is similar.

	FPFH [31]							FCGF [8]						
	IP (%↑)	IR (%↑)	F1 (%↑)	RE (°↓)	TE (cm↓)	**RR** (%↑)	TIME (s)	IP (%↑)	IR (%↑)	F1 (%↑)	RE (°↓)	TE (cm↓)	**RR** (%↑)	TIME (s)
SC^2-PCR [5]	74.03	79.34	76.38	2.12	6.69	83.80	**0.03**	79.91	87.13	83.06	2.06	6.53	93.16	**0.03**
MAC [46]	74.56	78.96	76.55	2.11	6.79	83.92	1.77	79.93	86.64	82.84	2.03	6.56	93.78	1.80
PointDSC [2]	68.62	71.79	69.93	**2.04**	**6.42**	77.51	0.06	78.97	86.25	82.17	2.06	6.57	93.22	0.06
PG-Net [38]	71.57	76.60	73.82	2.30	6.73	82.32	0.19	79.25	86.53	82.43	2.15	6.51	93.10	0.19
VBReg [19]	70.10	72.61	71.13	2.30	7.00	79.73	0.19	79.79	86.12	82.57	2.03	6.47	92.98	0.19
Hunter [47]	71.79	76.96	74.12	2.12	6.63	82.38	0.12	79.19	85.29	81.90	2.06	6.54	93.47	0.11
3DPCP [37]	74.32	**80.83**	76.64	2.05	6.50	81.15	0.11	**89.65**	**95.94**	**91.25**	**1.95**	**6.31**	92.54	0.11
HyperGCT	**75.15**	80.10	**77.38**	2.26	6.74	**85.89**	0.20	80.72	87.56	83.75	2.04	6.34	**94.45**	0.21

Table 2. Registration results on 3DLoMatch.

	FPFH [31]							FCGF [8]						
	IP (%↑)	IR (%↑)	F1 (%↑)	RE (°↓)	TE (cm↓)	**RR** (%↑)	TIME (s)	IP (%↑)	IR (%↑)	F1 (%↑)	RE (°↓)	TE (cm↓)	**RR** (%↑)	TIME (s)
SC^2-PCR [5]	32.46	38.24	34.74	4.04	10.32	38.46	**0.03**	46.44	56.05	50.25	3.79	10.37	58.62	**0.03**
MAC [46]	**35.27**	**40.8**	**37.45**	4.04	10.61	41.38	1.24	44.97	54.05	48.51	**3.75**	10.61	60.08	1.26
PointDSC [2]	25.53	27.68	26.2	**3.81**	**9.80**	29.20	0.05	44.35	52.47	47.57	3.76	10.47	57.33	0.05
PG-Net [38]	30.59	35.5	32.52	5.05	10.30	37.11	0.16	44.97	54.07	48.58	4.42	10.53	58.39	0.17
VBReg [19]	28.46	31.2	29.41	4.13	10.61	32.57	0.20	45.24	53.44	48.48	3.81	10.68	57.50	0.20
Hunter [47]	31.37	36.85	33.54	3.96	10.15	38.24	0.09	46.48	55.11	49.97	3.79	10.51	60.02	0.09
3DPCP [37]	28.13	35.15	30.39	3.96	9.99	32.51	0.10	**49.35**	**65.29**	**53.47**	**3.75**	10.44	56.71	0.10
HyperGCT	34.08	39.57	36.27	4.81	10.8	**42.28**	0.18	48.99	58.00	52.59	4.18	**10.17**	**63.73**	0.20

and 30cm, respectively, and randomly sample 1000 points with computed FCGF features to build correspondences for training. We set parameter σ_d to $0.1m$ for 3DMatch and $1.2m$ for KITTI-10m, and initialize σ_f to 1.0. Using a batch size of 6, we train the network for 50 epochs using the ADAM optimizer, starting with an initial learning rate of 0.0001 and a decay factor of 0.99. The number of seeds N_s is set to $0.2|\mathbf{C}_\mathrm{initial}|$, and N_initial is set to $0.1N_\mathrm{s}$. Following [34], the size of the sampled minimal is set to 6, with a maximum of 30 iterations and a step size of 3. All experiments are conducted on a machine with an Intel Xeon E5-2690 CPU and NVIDIA RTX3090 GPUs. To fairly compare the actual standalone performance, we *DO NOT* apply ICP [32] to refine the results of all the methods.

4.2. Results on Indoor Scenes

We compare our method with two traditional methods SC^2-PCR [5], MAC [46], and five deep-learned outlier removal methods PointDSC [2], PG-Net [38], VBReg [19], Hunter [47], 3DPCP [37]. Note that we retrain all deep-learned methods for testing. We generate correspondences using FPFH [31] and FCGF [8] descriptors following [2, 5]. As shown in Table 1, HyperGCT achieves the highest RR of 85.89% with FPFH and 94.45% with FCGF, outperforming all the compared methods. Notably, when combined with FPFH, our method surpasses 3DPCP, a state-of-the-art correspondence pruning network, in terms of IP and F1. As shown in Table 2, HyperGCT also outperforms the others, achieving RR of 42.28% with FPFH and 63.73% with FCGF. Additionally, our method has the same time consumption as VBReg. HyperGCT is trained using only

Table 3. Registration results (RR) on KITTI-10m.

	FPFH [31]				FCGF [8]			
	8	9	10	**AVG.**	8	9	10	**AVG.**
SC^2-PCR [5]	98.37	**99.38**	**100**	98.92	99.02	97.52	98.85	98.56
MAC [46]	**99.35**	98.76	98.84	**99.10**	98.37	**98.14**	98.85	98.38
PointDSC [2]	97.71	98.14	**100**	98.19	99.02	97.52	97.70	98.38
PG-Net [38]	97.71	98.76	**100**	98.38	99.02	97.52	98.85	98.56
VBReg [19]	98.37	98.14	**100**	98.56	99.02	**98.14**	98.85	98.74
Hunter [47]	98.37	98.76	**100**	98.74	99.02	97.52	**100**	98.74
3DPCP [37]	98.69	**99.38**	**100**	**99.10**	99.02	95.65	97.70	97.83
HyperGCT	98.69	**99.38**	**100**	**99.10**	**99.35**	97.52	**100**	**98.92**

Table 4. Registration results (RR) on KITTI-LC with FPFH.

	0-10m	10-20m	20-30m
SC^2-PCR [5]	95.84	61.86	18.25
MAC [46]	**97.7**	69.24	23.73
PointDSC [2]	93.33	53.87	17.54
PG-Net [38]	93.00	42.75	10.48
VBReg [19]	96.28	56.91	17.46
Hunter [47]	92.34	50.22	13.89
3DPCP [37]	96.94	67.25	23.49
HyperGCT	97.05	**72.46**	**25.95**

FCGF correspondences, yet it still achieves the best performance when combined with other descriptors, demonstrating the strong generalization ability of our approach. Please refer to Supp. 7 for results with other methods.

4.3. Results on Outdoor Scenes

Following [28], we conduct comparisons on KITTI-10m and KITTI-LC for a comprehensive evaluation, using the same methods as in the indoor scene experiments. As shown in Table 3, our method achieves the highest RR of 99.10% with FPFH and 98.92% with FCGF on KITTI-10m,

Table 5. Generalize between 3DMatch and KITTI-10m.

	3DMatch→KITTI-10m		KITTI-10m→3DMatch	
	FPFH [31]	FCGF [8]	FPFH [31]	FCGF [8]
SC^2-PCR [5]	98.92	98.56	83.80	93.16
MAC [46]	**99.10**	98.38	83.92	93.78
PointDSC [2]	91.16	98.56	72.46	92.42
PG-Net [38]	50.36	94.22	76.22	91.99
VBReg [19]	97.11	98.56	76.16	92.17
Hunter [47]	98.01	**98.92**	82.13	93.47
3DPCP [37]	98.01	98.01	54.71	83.43
HyperGCT	**99.10**	**98.92**	**85.40**	**94.39**

Table 6. Generalize from 3DMatch to ETH.

	FPFH [31]				
	summer	winter	autmn	summer	**AVG.**
SC^2-PCR [5]	**57.07**	**34.60**	46.96	54.40	45.06
MAC [46]	46.74	27.68	33.04	43.20	36.12
PointDSC [2]	25.00	14.19	3.48	9.60	14.45
PG-Net [38]	41.85	21.80	24.35	32.80	29.31
VBReg [19]	38.59	22.15	16.52	24.80	25.95
Hunter [47]	47.28	25.95	42.61	44.00	37.31
3DPCP [37]	53.26	32.87	41.74	44.80	41.65
HyperGCT	**57.07**	31.83	**50.43**	**56.80**	**45.72**
	FCGF [8]				
	summer	winter	autmn	summer	**AVG.**
SC^2-PCR [5]	**79.89**	49.83	78.26	79.20	**67.32**
MAC [46]	75.54	42.91	71.30	73.60	61.29
PointDSC [2]	41.85	26.99	33.91	45.60	35.2
PG-Net [38]	71.20	39.10	64.35	68.00	56.52
VBReg [19]	66.30	35.29	66.09	59.20	52.45
Hunter [47]	61.96	27.34	61.74	61.60	50.77
3DPCP [37]	79.35	**51.56**	79.13	69.60	66.34
HyperGCT	79.35	47.06	**80.87**	**80.00**	66.62

with a more pronounced performance gap using FCGF. On KITTI-LC, as the translation distance increases, the outlier ratio rises, posing a challenge for registration. As shown in Table 4, HyperGCT outperforms other deep-learned methods on the "0-10m" subset and surpasses all others on the "10-20m" and "20-30m" subset, demonstrating its flexibility in handling correspondences of varying quality.

4.4. Generalization Results

We conduct generalization experiments across different data modalities (3DMatch, KITTI-10m, and ETH). Methods for comparison include SC^2-PCR [5], MAC [46], PointDSC [2], PG-Net [38], VBReg [19], Hunter [47], and 3DPCP [37]. No parameter tuning is performed.

Between 3DMatch and KITTI-10m. The direct generalization performance of the five comparison methods is lower than their results when trained on the corresponding datasets. In contrast, our method surpasses all comparison methods and maintains strong generalization capability. As shown in Table 5, our method's generalization performance on KITTI is consistent with the performance when trained on KITTI, and its generalization performance on 3DMatch is slightly lower ($<0.5\%$) than when trained on 3DMatch.

From 3DMatch, KITTI-10m to ETH. We further test the generalization of all methods on ETH, which represents more complex geometries and challenges such as clutter and occlusion. As shown in Tables 6 and 7, our method outperforms the others in all settings. Moreover, the results of our method, generalized from 3DMatch and KITTI-10m are relatively consistent, indicating strong generalization capability across different datasets.

Table 7. Generalize from KITTI-10m to ETH.

	FPFH [31]				
	summer	winter	autmn	summer	**AVG.**
SC^2-PCR [5]	57.07	34.60	46.96	**54.40**	45.06
MAC [46]	46.74	27.68	33.04	43.20	36.12
PointDSC [2]	17.93	8.30	6.96	5.60	10.10
PG-Net [38]	22.28	10.38	6.96	8.00	12.48
VBReg [19]	41.85	26.30	30.43	32.80	32.12
Hunter [47]	54.89	26.64	42.61	45.60	39.83
3DPCP [37]	53.26	**37.37**	36.52	40.80	41.94
HyperGCT	**57.61**	35.99	**49.57**	51.20	**46.42**
	FCGF [8]				
	summer	winter	autmn	summer	**AVG.**
SC^2-PCR [5]	79.89	**49.83**	78.26	79.20	67.32
MAC [46]	75.54	42.91	71.30	73.60	61.29
PointDSC [2]	41.85	23.18	36.52	36.80	32.54
PG-Net [38]	52.17	30.80	44.35	36.80	39.55
VBReg [19]	62.50	45.33	72.17	68.00	58.06
Hunter [47]	67.39	30.45	63.48	61.60	50.77
3DPCP [37]	66.30	47.06	70.43	60.80	58.20
HyperGCT	**81.52**	47.75	**80.00**	**80.80**	**67.46**

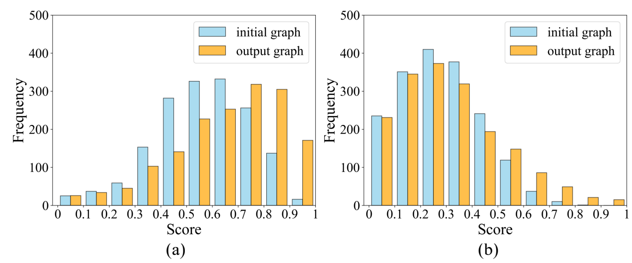

Figure 3. Improvement of graph quality HyperGCT brings when combined with FCGF on 3DMatch (a) and 3DLoMatch (b).

4.5. Analysis Experiments

Improvement of Graph Quality. To assess the graph quality improvement brought by HyperGCT, we introduce a metric called hyperedge precision $\mathcal{P}(\boldsymbol{\mathcal{E}})$, defined as:

$$\mathcal{P}(\boldsymbol{\mathcal{E}}) = \frac{1}{|\boldsymbol{\mathcal{E}}|}\sum_i \frac{|e_i^{\text{correct}}|}{|e_i|}, \tag{16}$$

where $e_i^{\text{correct}} = \{\mathrm{v}_i^j | \mathrm{v}_i^j \in e_i \cap e_i^{\text{inlier}}\}$ and e_i^{inlier} denotes the ground truth hyperedge formed by inlier labels. Scores of $\mathcal{P}(\boldsymbol{\mathcal{E}})$ lie in [0,1] and are divided into 10 equal bins for frequency analysis on 3DMatch and 3DLoMatch when combined with FCGF. As shown in Fig. 3, the frequency in the lower $\mathcal{P}(\boldsymbol{\mathcal{E}})$ range decreases in the hypergraph output by HyperGCT, while the frequency in the higher $\mathcal{P}(\boldsymbol{\mathcal{E}})$ range

Table 8. Robustness to threshold θ_{cmp}. '✗' denotes out of memory. ↓ / ↑ denote the decline / rise rate (%) compared to default.

			Default	0.8	0.85	0.9	0.95	0.99
3DMatch	FPFH	MAC	83.92	✗	✗	✗	✗	↓0.49
		HyperGCT	85.89	↓0.06	↓0.37	↓0.25	↓0.25	↓0.68
	FCGF	MAC	93.78	✗	✗	↓2.90	↓2.10	↓0.31
		HyperGCT	94.45	↓0.06	↓0.06	↓0.12	↓0.12	↓0.06
3DLoMatch	FPFH	MAC	41.38	✗	✗	↓0.05	↓0.45	0
		HyperGCT	42.28	↑0.50	↑0.79	↑0.73	↑0.67	↑0.17
	FCGF	MAC	60.08	↓5.84	↓5.50	↓3.82	↓0.73	0
		HyperGCT	63.73	↑0.05	↑0.34	0	↓0.11	↑0.05

Table 9. Robustness to threshold θ_{inlier}.

			0.1 (Default)	0.05	0.15	0.2
3DMatch	FPFH	MAC	83.92	↓3.27	0	↓0.06
		HyperGCT	85.89	↓0.68	↓0.49	↓0.92
	FCGF	MAC	93.78	↓1.05	↓0.43	↓0.93
		HyperGCT	94.45	↓0.18	↓0.12	↓0.12
3DLoMatch	FPFH	MAC	41.38	↓3.70	↓1.85	↓3.76
		HyperGCT	42.28	↑0.79	↓0.45	↓0.39
	FCGF	MAC	60.08	↓1.91	↓0.90	↓2.98
		HyperGCT	63.73	↓0.11	↑0.22	0

increases. This indicates that HyperGCT effectively captures the high-order consistency of inliers around the nodes, thereby improving the overall hyperedge precision.

Robustness to Noise in Graph Construction. We compare our method with an explicit graph construction method, MAC [46], to test robustness against graph noise from measurement and matching errors. Measurement error affects correspondence compatibility (*i.e.*, whether the compatibility score is less than threshold θ_{cmp}), while matching error identifies inliers (*i.e.*, whether the residual is below threshold θ_{inlier}). As shown in Table 8, reducing θ_{cmp} causes MAC to face out-of-memory issues due to graph density, with performance dropping by up to 5.84% on 3DLoMatch using FCGF. In contrast, our method remains stable, with performance variation under 1%. Similar trends with θ_{inlier} changes (Table 9) highlight the superior robustness of our method to graph noise.

Impact of Graph Update Frequency. The frequency significantly influences performance by affecting both computational efficiency and the quality of information propagation within the network. As shown in Table 10, excessive updates (more than 5 times) can increase model parameter sizes, thereby slowing convergence rates with the same training epochs. In contrast, insufficient updates (less than 5 times) allow more outliers to interfere with the information flow, which degrades overall performance. This highlights that there exists an optimal frequency for graph updates where computational resources are efficiently utilized and high-order relationships are accurately captured.

Combined with Different PCR Back-ends. We integrate HyperGCT with other registration back-ends, such as SC^2-PCR and Hunter's PLE. This integration incorporates the learned incidence matrix $\mathbf{H}$ into their compatibility calculations. As shown in Table 11, this integration significantly improves the performance of both registration back-ends. SC^2-PCR, in particular, demonstrates a more significant enhancement: for instance, a 2.34% increase on the 3DMatch with FPFH and a 4.55% increase on the 3DLoMatch with FCGF. The flexible constraint effectively enhances different solvers, demonstrating the adaptability and flexibility.

Table 10. Performance varying graph update times.

# update	# params (M)	3DMatch		3DLoMatch	
		FCGF (%)	FPFH (%)	FCGF (%)	FPFH (%)
1	0.30	93.96	84.47	63.05	40.93
3	0.64	94.15 (↑0.19)	85.15 (↑0.68)	63.28 (↑0.23)	41.66 (↑0.73)
5	0.98	**94.45** (↑0.49)	**85.89** (↑1.42)	**63.73** (↑0.68)	42.28 (↑1.35)
7	1.32	**94.45** (↑0.49)	84.72 (↑0.25)	63.39 (↑0.34)	**42.56** (↑1.63)
9	1.66	94.15 (↑0.19)	85.09 (↑0.62)	63.17 (↑0.12)	41.61 (↑0.68)

Table 11. Results of HyperGCT combined with other back-ends.

			SC^2-PCR	PLE
w/o HyperGCT	3DMatch	FPFH	83.80	82.38
		FCGF	93.16	93.47
	3DLoMatch	FPFH	38.46	38.24
		FCGF	58.62	60.02
w/ HyperGCT	3DMatch	FPFH	86.14 (↑2.34)	82.99 (↑0.61)
		FCGF	93.96 (↑0.80)	93.90 (↑0.43)
	3DLoMatch	FPFH	42.22 (↑3.76)	38.97 (↑0.73)
		FCGF	63.17 (↑4.55)	60.08 (↑0.06)

Table 12. Correctness (%) of generated hypotheses.

		RANSAC [16]	SAC-COT [41]	MAC [46]	HyperGCT
3DMatch	FPFH	0.74	0.46	6.64	**62.66**
	FCGF	10.17	9.35	23.62	**87.62**
3DLoMatch	FPFH	0.04	0.02	0.93	**14.61**
	FCGF	1.18	1.92	6.43	**40.51**

Correctness of Hypothesis Generation. To evaluate HyperGCT against random sampling, triplet, and maximal clique constraints, we assess its ability to generate valid hypotheses. As shown in Table 12, HyperGCT demonstrates superior capability in producing accurate high-order consistency hypotheses compared to these traditional approaches, which validates its robustness and effectiveness. Furthermore, the robustness of HyperGCT's solver enhances both the stability of the hypothesis generation process and the accuracy of pose estimation and verification.

5. Conclusion

This paper presents HyperGCT, a Hyper-GNN-learned geometric constraint for accurate hypothesis generation. HyperGCT is superior to competitors on several benchmarks, is robust to graph noise, and generalizes well across datasets. The GF-NMS module may not retrieve enough correct matches for accurate hypothesis generation. We will design a seeding module to select dynamically and reliably.

Acknowledgments. This work is partly supported by National Natural Science Foundation of China (No. 62372377 and 52172380) and China Postdoctoral Science Foundation (No. 2024M761014).

References

[1] Junjie Bai, Biao Gong, Yining Zhao, Fuqiang Lei, Chenggang Yan, and Yue Gao. Multi-scale representation learning on hypergraph for 3d shape retrieval and recognition. *IEEE Transactions on Image Processing*, 30:5327–5338, 2021. 3

[2] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15859–15869. IEEE, 2021. 1, 2, 4, 5, 6, 7

[3] Daniel Barath and Jiří Matas. Graph-cut ransac. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6733–6741, 2018. 1, 2

[4] Daniel Barath, Jana Noskova, Maksym Ivashechkin, and Jiri Matas. Magsac++, a fast, reliable and accurate robust estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1304–1312, 2020. 1, 2

[5] Zhi Chen, Kun Sun, Fan Yang, Lin Guo, and Wenbing Tao. Sc2-pcr++: Rethinking the generation and selection for efficient and robust point cloud registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10): 12358–12376, 2023. 3, 4, 6, 7, 2

[6] Yuxin Cheng, Zhiqiang Huang, Siwen Quan, Xinyue Cao, Shikun Zhang, and Jiaqi Yang. Sampling locally, hypothesis globally: accurate 3d point cloud registration with a ransac variant. *Visual Intelligence*, 1(1):20, 2023. 1

[7] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5556–5565, 2015. 1

[8] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proc. IEEE International Conference on Computer Vision*, pages 8958–8966, 2019. 3, 5, 6, 7

[9] 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. IEEE, 2020. 1, 2, 5

[10] Ondrej Chum and Jiri Matas. Matching with prosac-progressive sample consensus. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 220–226. IEEE, 2005. 1, 2

[11] Ondřej Chum and Jiří Matas. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008. 2

[12] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings 25*, pages 236–243. Springer, 2003. 2

[13] Qinwen Deng, Songyang Zhang, and Zhi Ding. An efficient hypergraph approach to robust point cloud resampling. *IEEE Transactions on Image Processing*, 31:1924–1937, 2022. 3

[14] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006. 1

[15] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3558–3565, 2019. 2

[16] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 2, 8

[17] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021. 4

[18] 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 Conference on Computer Vision and Pattern Recognition*, pages 4267–4276, 2021. 5, 2

[19] 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. 1, 2, 6, 7

[20] 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. 1, 2

[21] Raobo Li, Xiping Yuan, Shu Gan, Rui Bi, Sha Gao, Weidong Luo, and Cheng Chen. An effective point cloud registration method based on robust removal of outliers. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 1, 2

[22] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Learning of 3d graph convolution networks for point cloud analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4212–4224, 2021. 4

[23] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 5

[24] Kai Ni, Hailin Jin, and Frank Dellaert. Groupsac: Efficient consensus in the presence of groupings. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2193–2200. IEEE, 2009. 1, 2

[25] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7193–7203. IEEE, 2020. 1, 2

[26] François Pomerleau, Ming Liu, Francis Colas, and Roland Siegwart. Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14):1705–1711, 2012. 5, 2

[27] François Pomerleau, Francis Colas, Roland Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015. 1

[28] Zhijian Qiao, Zehuan Yu, Binqian Jiang, Huan Yin, and Shaojie Shen. G3reg: Pyramid graph-based global registration using gaussian ellipsoid model. *IEEE Transactions on Automation Science and Engineering*, 2024. 1, 2, 5, 6

[29] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, Slobodan Ilic, Dewen Hu, and Kai Xu. Geotransformer: Fast and robust point cloud registration with geometric transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9806–9821, 2023. 1, 2

[30] Ryan A Rossi, David F Gleich, and Assefaw H Gebremedhin. Parallel maximum clique algorithms with applications to network analysis. *SIAM Journal on Scientific Computing*, 37(5):C589–C616, 2015. 2

[31] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009. 3, 6, 7

[32] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, page 435. Seattle, WA, 2009. 6

[33] Lei Sun and Lu Deng. Trivoc: Efficient voting-based consensus maximization for robust point cloud registration with extreme outlier ratios. *IEEE Robotics and Automation Letters*, 7(2):4654–4661, 2022. 1, 2

[34] Ruwan B Tennakoon, Alireza Bab-Hadiashar, Zhenwei Cao, Reza Hoseinnezhad, and David Suter. Robust model fitting using higher than minimal subset sampling. *IEEE transactions on pattern analysis and machine intelligence*, 38(2): 350–362, 2015. 6

[35] Philip HS Torr and Andrew Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1):138–156, 2000. 2

[36] Philip Hilaire Torr, Slawomir J Nasuto, and John Mark Bishop. Napsac: High noise, high dimensional robust estimation-it's in the bag. In *British Machine Vision Conference (BMVC)*, page 3, 2002. 2

[37] Jingtao Wang and Zechao Li. 3dpcp-net: A lightweight progressive 3d correspondence pruning network for accurate and efficient point cloud registration. In *ACM Multimedia*, 2024. 1, 2, 6, 7

[38] Jingtao Wang, Xin Liu, Luanyuan Dai, Jiayi Ma, Lifang Wei, Changcai Yang, and Riqing Chen. Pg-net: Progressive guidance network via robust contextual embedding for efficient point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–12, 2023. 1, 2, 6, 7

[39] Lang Wu, Ruoyu Jia, Kai Zhong, Zhongwei Li, Junyun Shang, and Hua Luo. A robust and accurate post-validation voting scheme for ranking 3d correspondences. In *Proceedings of the International Conference on Computing and Artificial Intelligence*, pages 618–624, 2022. 1, 2

[40] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020. 2

[41] Jiaqi Yang, Zhiqiang Huang, Siwen Quan, Zhaoshuai Qi, and Yanning Zhang. Sac-cot: Sample consensus by sampling compatibility triangles in graphs for 3-d point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2021. 1, 2, 4, 8

[42] Jiaqi Yang, Zhiqiang Huang, Siwen Quan, Qian Zhang, Yanning Zhang, and Zhiguo Cao. Toward efficient and robust metrics for ransac hypotheses and 3d rigid registration. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(2):893–906, 2021. 2, 5

[43] Jiaqi Yang, Jiahao Chen, Siwen Quan, Wei Wang, and Yanning Zhang. Correspondence selection with loose–tight geometric voting for 3-d point cloud registration. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022. 1, 2

[44] Jiaqi Yang, Xiyu Zhang, Shichao Fan, Chunlin Ren, and Yanning Zhang. Mutual voting for ranking 3d correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1, 2

[45] Jiaqi Yang, Chu'ai Zhang, Zhengbao Wang, Xinyue Cao, Xuan Ouyang, Xiyu Zhang, Zhenxuan Zeng, Zhao Zeng, Borui Lu, Zhiyi Xia, et al. 3d registration in 30 years: A survey. *arXiv preprint arXiv:2412.13735*, 2024. 5

[46] Jiaqi Yang, Xiyu Zhang, Peng Wang, Yulan Guo, Kun Sun, Qiao Wu, Shikun Zhang, and Yanning Zhang. Mac: Maximal cliques for 3d registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 2, 3, 4, 5, 6, 7, 8

[47] Runzhao Yao, Shaoyi Du, Wenting Cui, Aixue Ye, Feng Wen, Hongbo Zhang, Zhiqiang Tian, and Yue Gao. Hunter: Exploring high-order consistency for point cloud registration with severe outliers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1, 2, 3, 6, 7

[48] 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, 2021. 1, 2

[49] 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, 2

[50] Songyang Zhang, Shuguang Cui, and Zhi Ding. Hypergraph spectral analysis and processing in 3d point cloud. *IEEE Transactions on Image Processing*, 30:1193–1206, 2020. 3

[51] Zizhao Zhang, Haojie Lin, Xibin Zhao, Rongrong Ji, and Yue Gao. Inductive multi-hypergraph learning and its application on view-based 3d object classification. *IEEE Transactions on Image Processing*, 27(12):5957–5968, 2018. 3