Wasserstein Hypergraph Learning for General 3D Point Cloud Alignment

Chanyoung Park chp026@ucsd.edu
Minghan Wu miw039@ucsd.edu
Gal Mishne gmishne@ucsd.edu
Yusu Wang yuw122@ucsd.edu

Abstract

We study whether Wasserstein-based aggregation can improve hypergraph neural networks for 3D point cloud alignment. Building on HyperGCT, we replace mean pooling with a sliced-Wasserstein aggregation operator that treats local neighborhood features as empirical distributions rather than summarizing them by a simple average. We evaluate several layer-placement strategies on FAUST and PartNet across piecewise-rigid, rigid, and affine settings, and measure both correspondence performance and downstream alignment quality. Our results show that at least one Wasserstein aggregation configuration improves over the learned HyperGCT baseline in every evaluated regime, while the best placement depends on the dataset and transformation setting. In particular, all-layer aggregation performs best on FAUST, whereas the strongest configuration on PartNet varies across rigid and affine regimes. These findings show that distribution-aware aggregation can strengthen hypergraph-based geometric learning, and that its effectiveness depends on architectural placement.

Website: https://youngpark1516.github.io/Hypergraph-Alignment/
Code: https://github.com/youngpark1516/Hypergraph-Alignment

1 Introduction 3
2 Background and Related Work 4
3 Methods 5
4 Results 12
5 Discussion 14
6 Limitations 15
7 Conclusion 16
Appendices A1

1 Introduction

Accurate 3D point cloud alignment is a core problem in geometric learning, with applications in shape correspondence, reconstruction, and tracking. The goal is to estimate a consistent mapping between two point sets despite common challenges such as non-rigid deformation, partial observations, and noise. Many modern pipelines rely on learned local features and neighborhood reasoning, since alignment quality often depends on whether local geometric structure is preserved well enough to disambiguate correspondences (Qi et al. 2017; Wang et al. 2019).

Hypergraph neural networks provide a natural framework for point cloud alignment because local geometry is not always well modeled by pairwise edges alone (Feng et al. 2019). In particular, HyperGCT models higher-order relationships by grouping points into hyperedges that represent local patches or neighborhoods, and then propagates information through hypergraph message passing (Zhang et al. 2025). However, a key step in hypergraph learning is the aggregation of features within a neighborhood or hyperedge. In standard HyperGCT implementations, this aggregation is typically done using mean pooling. While mean pooling is permutation-invariant and efficient (Zaheer et al. 2017), it can discard informative local structure by collapsing a neighborhood into a single average. This becomes especially problematic when neighborhoods are multi-modal or when perturbations alter point sampling and local density.

In this work, we modify HyperGCT by replacing mean pooling with a Wasserstein-based aggregation operator that treats node features within a hyperedge as an empirical distribution rather than collapsing them to a simple average. Our primary goal is to test whether this distribution-aware aggregation improves correspondence prediction and downstream alignment, and whether its effect depends on where it is inserted in the network. Concretely, we evaluate several layer-placement strategies, ranging from early-layer replacement to all-layer replacement, while keeping the rest of the learned architecture as consistent as possible (Villani 2009; Peyré and Cuturi 2019).

We evaluate the modified model on FAUST and PartNet across piecewise-rigid, rigid, and affine settings (Bogo et al. 2014; Mo et al. 2019). In our setup, FAUST serves as a piecewise-rigid regime, while PartNet is evaluated under one rigid and two affine perturbation settings. We report both correspondence metrics and downstream alignment quality, together with structural diagnostics that help explain how different aggregation choices affect the learned hypergraph organization. In addition, for downstream alignment evaluation, we use a matching-based hypothesis-generation stage in place of HyperGCT's original post-processing pipeline so that all learned variants are compared under the same decoding procedure. Overall, our experiments show that Wasserstein aggregation improves over the learned HyperGCT baseline in every evaluated regime for at least one placement strategy, while also showing that the best placement is regime-dependent.

2 Background and Related Work

2.1 Classical Point Cloud Registration

A standard baseline for point cloud alignment is Iterative Closest Point (ICP), which alternates between assigning correspondences, typically by nearest neighbors, and estimating the rigid transformation that best fits those correspondences (Besl and McKay 1992). ICP is simple and widely used, but its performance depends strongly on initialization and correspondence quality. It can degrade under partial overlap, significant noise, non-uniform sampling, and non-rigid deformation, where correspondence ambiguity violates its assumptions (Chen and Medioni 1992).

2.2 Learning-Based Registration

Learning-based registration methods aim to produce features and correspondence scores that are more robust to sampling variation, noise, and deformation. Many approaches learn local descriptors and then use matching or attention-style reasoning to infer correspondences before estimating the final transformation. A common theme is that local neighborhoods must be represented well for correspondence confidence to remain reliable under perturbation. Representative methods include PointNetLK (Aoki et al. 2019), Deep Closest Point (DCP) (Wang and Solomon 2019), and RPM-Net (Yew and Lee 2020).

2.3 Graph and Hypergraph Models for 3D Geometry

Graph neural networks model neighborhood interactions explicitly (Kipf and Welling 2017), while hypergraph neural networks extend this idea to higher-order relationships among groups of points (Feng et al. 2019). This makes hypergraph models especially relevant for alignment tasks, where local consistency often depends on interactions among multiple candidates rather than only pairwise relations. HyperGCT is an example of this direction, combining dynamic hypergraph construction with higher-order consistency reasoning for 3D registration (Zhang et al. 2025).

2.4 Optimal Transport and Wasserstein Geometry for Aggregation

Optimal transport provides a principled way to compare and aggregate distributions by accounting for the geometry of how mass moves between them (Villani 2009; Peyré and Cuturi 2019). In modern machine learning, scalable OT often relies on regularized formulations and Sinkhorn-style solvers (Cuturi 2013). This perspective is relevant for neighborhood aggregation: instead of summarizing a set of features only by their mean, one can treat the neighborhood as an empirical distribution and aggregate it using transport-aware operators. Recent work has argued for Wasserstein-style pooling in hypergraph settings (Duta

and Liò 2025), and sliced/generalized sliced Wasserstein formulations provide practical tools for doing so efficiently (Kolouri et al. 2019).

3 Methods

3.1 Problem Setup

Given a source point cloud $\mathscr{P}^{\text{src}} = \{p_i^{\text{src}}\}_{i=1}^{N_s}$ and a target point cloud $\mathscr{P}^{\text{tgt}} = \{p_j^{\text{tgt}}\}_{j=1}^{N_t}$, the goal is to estimate reliable correspondences between points across the two shapes. In our setting, the model operates on a candidate set of source–target pairs rather than all possible pairings, so the task is semi-dense correspondence prediction over a filtered set of plausible matches.

Each point is associated with geometric and descriptor features, such as coordinates, normals, and local descriptors, denoted by

$$x_i^{\text{src}} \in \mathbb{R}^d, \qquad x_j^{\text{tgt}} \in \mathbb{R}^d. \tag{1}$$

We formulate correspondence prediction as pair classification over candidate matches:

$$v_{ij} \equiv (p_i^{\text{src}}, p_j^{\text{tgt}}), \quad v_{ij} \in \mathscr{V}, \tag{2}$$

where $\mathscr{V}$ is the candidate set. During training, $\mathscr{V}$ is formed from ground-truth positive pairs together with randomly sampled negatives. During evaluation, it is formed from label-blind k-nearest-neighbor retrieval in feature space. A candidate pair therefore represents a plausible source–target match that must be classified as correct or incorrect.

The objective is to learn a classifier $\mathscr{M} : \mathscr{V} \to \{0, 1\}$ that predicts whether a candidate pair is a true correspondence. We evaluate $\mathscr{M}$ using precision, recall, accuracy, and a distance-aware loss

$$\mathscr{L}_{dist} = \frac{1}{|\mathscr{V}_1|} \sum_{v_{ij} \in \mathscr{V}_1} \|p_j^{tgt} - q_i^{tgt}\|, \tag{3}$$

where $\mathscr{V}_1 \subseteq \mathscr{V}$ is the subset mapped to 1 by $\mathscr{M}$ and q_i^{tgt} is the ground-truth correspondence of p_i^{src} in the target.

We evaluate the method under piecewise-rigid, rigid, and affine settings. In our setup, FAUST represents the piecewise-rigid regime, while PartNet is used for one rigid setting and two affine perturbation settings of increasing strength.

3.2 Hypergraph Construction

Nodes. Each node corresponds to a candidate source–target pair v_{ij}. Candidate generation is performed either by (i) ground-truth positives with negative sampling during training, or (ii) k-NN retrieval in feature space during evaluation.

Hyperedges. Unlike pairwise graphs, hyperedges connect groups of candidate pairs that are likely to share a common transformation hypothesis. Let $\mathscr{E} = \{e_m\}_{m=1}^{M}$ denote the set of hyperedges with fixed budget M. Following the HyperGCT construction, hyperedges are initialized from geometric consistency between candidate pairs. For two candidate correspondences, we first measure the discrepancy between source and target pairwise distances:

$$d_{ij} = \|\mathbf{p}_i^{\text{src}} - \mathbf{p}_j^{\text{src}}\|_2 - \|\mathbf{p}_i^{\text{tgt}} - \mathbf{p}_j^{\text{tgt}}\|_2. \tag{4}$$

This is converted into a feature-consistency score

$$\text{FCG}_{ij} = \max\left(1 - \frac{d_{ij}^2}{\sigma_d^2}, 0\right), \qquad \text{FCG}_{ii} = 0. \tag{5}$$

The resulting matrix is sparsified by row-wise thresholding, then converted into an initial weighted hypergraph matrix $\mathbf{W}$ and binary incidence matrix $\mathbf{H}_0$. Intuitively, this procedure groups together candidate correspondences that are mutually geometrically compatible, providing the initial higher-order structure that is later refined during message passing.

3.3 HyperGCT Baseline

The baseline network stacks $L = 6$ hypergraph message-passing layers. Each layer has two stages: (a) hypergraph propagation, consisting of node-to-hyperedge aggregation followed by hyperedge-to-node update, and (b) dynamic hypergraph refinement.

Let $h_v^{(\ell)}$ be the embedding of node v at layer ℓ. Hyperedge embeddings are computed by mean pooling:

$$h_e^{(\ell)} = \frac{1}{|e|} \sum_{v \in e} \phi^{(\ell)}\left(h_v^{(\ell)}\right), \tag{6}$$

where $\phi^{(\ell)}$ is an MLP. Node embeddings are then updated by aggregating messages from incident hyperedges:

$$\tilde{h}_v^{(\ell)} = \psi^{(\ell)}\left(h_v^{(\ell)}, \underset{e \ni v}{\text{AGG}}\, h_e^{(\ell)}\right), \qquad h_v^{(\ell+1)} = \text{LN}\left(h_v^{(\ell)} + \tilde{h}_v^{(\ell)}\right). \tag{7}$$

After each layer, pairwise consistency is recomputed from the updated node embeddings and the hypergraph is refreshed through re-ranking and regrouping. This enables iterative refinement of both the features and the underlying higher-order structure.

After the final layer, the model outputs correspondence confidence scores together with compatibility structures used for downstream hypothesis generation.

Our implementation follows the overall HyperGCT pipeline of Zhang et al. (2025), but the description above emphasizes only the components most relevant to our modification: hypergraph message passing, dynamic refinement, and the aggregation operator.

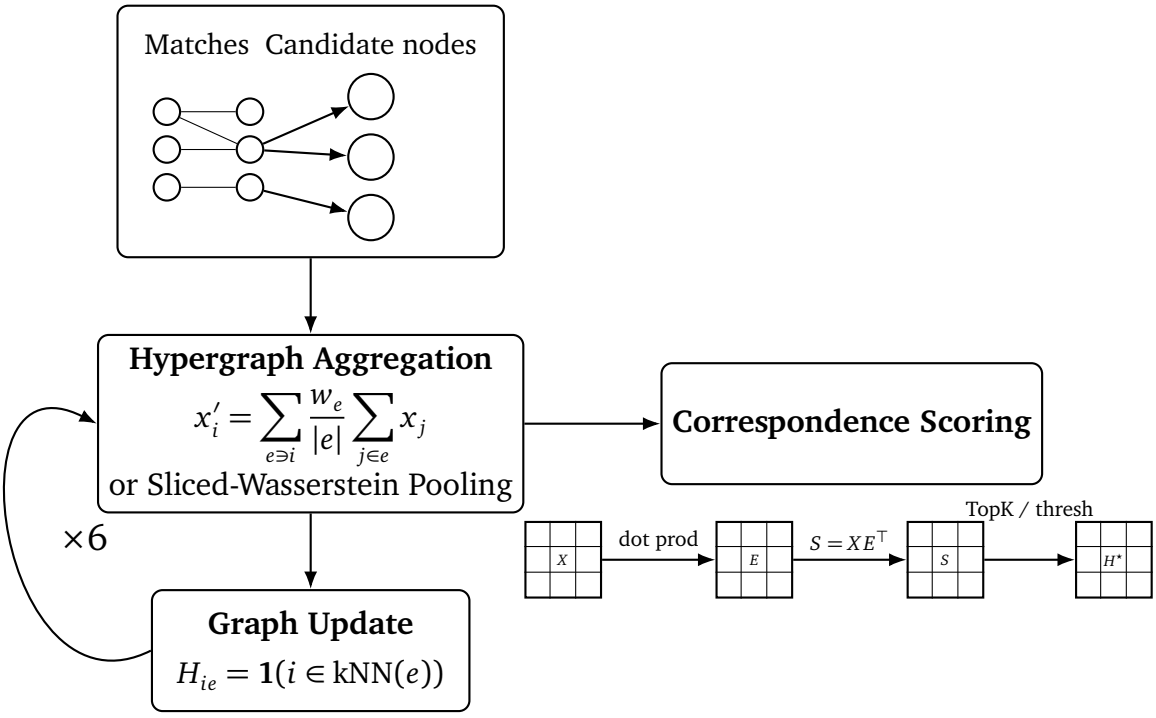

Figure 1: Illustration of HyperGCT architecture.

3.4 Wasserstein Aggregation

Mean pooling can over-smooth multi-modal node features inside a hyperedge. To preserve richer local structure, we replace mean aggregation in selected HyperGCT layers with a sliced-Wasserstein-based pooling operator. In our implementation, this acts as a drop-in replacement for the pooling step, while the rest of the HyperGCT architecture remains unchanged.

Sliced Wasserstein Distance. Given two probability distributions p_i, p_j on $\mathbb{R}^d$, the 2-Wasserstein distance is defined as

$$W_2(p_i, p_j) = \left(\inf_{\gamma \in \Gamma(p_i, p_j)} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\|^2 \, d\gamma(x, y) \right)^{1/2}, \tag{8}$$

where $\Gamma(p_i, p_j)$ denotes the set of transport plans with marginals p_i and p_j. Since computing this quantity directly is expensive, we use the *sliced Wasserstein distance*, which approximates it by projecting the distributions onto one-dimensional directions:

$$SW_2(p_i, p_j) = \left(\int_{S^{d-1}} W_2^2(P_\theta p_i, P_\theta p_j) \, d\theta \right)^{1/2} \approx \left(\frac{1}{L} \sum_{l=1}^{L} W_2^2(P_{\theta_l} p_i, P_{\theta_l} p_j) \right)^{1/2}, \tag{9}$$

where S^{d-1} is the unit sphere in $\mathbb{R}^d$, and P_θ denotes projection onto direction $\theta \in S^{d-1}$. This reduces the computation to multiple 1D Wasserstein distances, which can be computed in closed form. For two one-dimensional empirical distributions with equal weights:

$$W_2^2(P_\theta p_i, P_\theta p_j) = \frac{1}{n}\sum_{k=1}^{n}\left(x_{(k)} - y_{(k)}\right)^2. \tag{10}$$

where $\{x_{(k)}\}_{k=1}^n$ and $\{y_{(k)}\}_{k=1}^n$ denote the sorted projected samples.

Node → Hyperedge Pooling. In the hypergraph setting, each hyperedge e is treated as an empirical distribution whose samples are the node embeddings $\{\tilde{x}_i^e\}_{i\in e}$. Sliced Wasserstein pooling maps this set to a hyperedge representation

$$h_e = \mathrm{SWP}(\{\tilde{x}_i^e\}_{i\in e}), \tag{11}$$

which captures distributional properties such as spread and shape rather than only the mean. In practice, the operator projects node features onto multiple one-dimensional directions, computes transport-inspired slice-wise summaries from the sorted projected values, and combines those summaries through a learnable transformation to produce the hyperedge embedding.

Hyperedge → Node Pooling. The same operator is used in the reverse direction. Each node v_i receives messages from the hyperedges it belongs to:

$$\tilde{x}_i = \mathrm{SWP}(\{h_e \mid v_i \in e\}), \tag{12}$$

producing updated node embeddings. A Wasserstein hypergraph layer therefore follows the same two-stage message-passing pattern as the baseline, but replaces mean pooling with transport-aware aggregation.

We evaluate several replacement schemes, including the first two layers, the last two layers, the first and last layers, and all layers, to analyze where distribution-aware aggregation contributes most.

3.5 Hypothesis Generation

For downstream alignment evaluation, we use a matching-based post-processing stage in place of HyperGCT's original hypothesis-generation pipeline. This change is not the primary architectural focus of the report; rather, it provides a consistent decoding procedure for converting model outputs into final correspondence sets across all learned variants. Instead of generating transformation hypotheses through GF-NMS, subset sampling, and SVD, we construct correspondence sets directly from the predicted compatibility structure using greedy and spectral matching procedures.

Let $\mathcal{V}$ denote the set of candidate correspondences and let $A \in \{0,1\}^{N\times N}$ be the adjacency matrix derived from the learned compatibility output of the model. We convert model predictions into a consistent correspondence set in two stages: seed selection and matching-based expansion.

Stage 1: Seed selection. We first use the graph structure to define local neighborhoods for filtering. From A, we form the graph Laplacian

$$L = D - A. \tag{13}$$

Given confidence scores γ_i over candidate pairs, we apply graph-aware non-maximum suppression and retain a candidate v_i if

$$\gamma_i = \max_{j\in\mathcal{N}_A(i)} \gamma_j, \tag{14}$$

where $\mathcal{N}_A(i)$ denotes the neighborhood of i under A. The retained candidates form the seed set $\mathcal{S}$.

Stage 2: Consistent correspondence generation. Starting from the model outputs, we construct a final one-to-one matching using a compatibility matrix $M \in \mathbb{R}^{N\times N}$. We evaluate three strategies:

- **Greedy compatibility expansion:** starting from the highest-confidence seed, we iteratively add the candidate with the largest compatibility with the current set,

$$i^* = \arg\max_{i\notin\mathcal{V}^*} \sum_{j\in\mathcal{V}^*} M_{ij}, \tag{15}$$

  subject to one-to-one constraints.
- **Spectral matching:** we apply spectral relaxation to M, compute the leading eigenvector by power iteration,

$$\mathbf{v} \leftarrow \frac{M\mathbf{v}}{\|M\mathbf{v}\|}, \tag{16}$$

  and greedily decode matches in descending order of v_i under one-to-one constraints.
- **Two-stage spectral matching:** we first run spectral matching on the restricted seed matrix

$$M_{\mathcal{S}} = M[\mathcal{S},\mathcal{S}], \tag{17}$$

  obtain a consistent seed subset $\mathcal{S}^*$, then expand to a larger neighborhood and run spectral matching again on the expanded subgraph.

This post-processing module is applied after correspondence scoring for all learned variants in every evaluated setting, so differences in final alignment quality primarily reflect differences in the learned correspondence model rather than differences in decoding.

3.6 Baselines

ICP. A classical non-learning registration baseline. ICP is sensitive to initialization, overlap, and local minima, but serves as a geometric reference.

HyperGCT (Mean-Pooling). The original dynamic hypergraph architecture with standard mean aggregation is used as the direct architectural baseline.

3.7 Training Setup

The goal of this setup is to isolate the effect of the aggregation operator while keeping the rest of the training and evaluation pipeline as consistent as possible across experiments. Unless otherwise noted, models share the same optimization settings, feature pipeline, and post-processing procedure, and differ primarily in the aggregation operator and its placement.

Table 1: Shared training hyperparameters for learned models.

Setting	**Value**
Optimizer	Adam ($\beta_1 = 0.9$)
Learning rate	1×10^{-4}
Weight decay	1×10^{-6}
Epochs	100
LR scheduler	Exponential decay, $\gamma = 0.99$ per epoch
Training iterations per epoch	3500
Validation iterations per eval	1000
Feature dimension	33
Number of nodes	512
Descriptor	FPFH
Downsample voxel size	0.02
Generation method	spectral-2
Seed ratio	0.2
Registration thresholds	`re_thre` $= 15$, `te_thre` $= 30$
Loss weights	$\lambda_{\text{cls}} = 1,\ \lambda_{\text{graph}} = 1,\ \lambda_{\text{match}} = 1,\ \lambda_{\text{trans}} = 0$

Additional dataset-specific training details are provided in Appendix A.1.

Table 2: Aggregation configurations evaluated in the learned models.

Model	Aggregation	Wasserstein layer placement
HyperGCT	mean pooling	N/A (baseline)
WHNN (first)	Wasserstein	(0)
WHNN (first 2)	Wasserstein	(0, 1)
WHNN (last)	Wasserstein	(5)
WHNN (last 2)	Wasserstein	(4, 5)
WHNN (first + last)	Wasserstein	(0, 5)
WHNN (all layers)	Wasserstein	(0, 1, 2, 3, 4, 5)

3.8 Evaluation Metrics

Classification metrics. We report precision, recall, F1, classification loss, graph loss, and matching mean squared error (matching MSE):

$$\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN}, \quad F1 = \frac{2PR}{P+R}. \tag{18}$$

Distance metrics. After hypothesis generation, we evaluate the produced matching plan $\mathscr{M}$ with the distance-aware metric in (3). Among the matching-based post-processing variants considered, greedy expansion and two-stage spectral matching gave the strongest alignment results overall. We therefore report, for each dataset and model, the better L1 score produced by these two predefined decoding procedures, applying the same selection rule uniformly across all learned variants. For the alignment-improvement results, evaluation was repeated 10 times with different random seeds, and we report the mean and standard deviation across those runs. These repeated-seed statistics apply only to the L1-based alignment results, not to the validation classification metrics in Table 3.

Structural diagnostics. Beyond pair accuracy, we evaluate hypergraph quality via:

- **Aggregatedness**: degree to which true-compatible pairs are grouped within hyperedges.
- **Disjointness**: degree to which conflicting/incorrect pairs are separated across hyperedges.
- **In-edge Distance Consistency**: the variance of point distances of pairs in the same hyperedge

Higher aggregatedness and disjointness indicate cleaner hypothesis partitioning and typically correspond to better downstream correspondence quality. In-edge Distance Consistency measures the geometric consistency of pairs grouped within the same hyperedge.

These structural metrics are not direct task objectives, but they help diagnose how well the learned hypergraph organizes compatible and incompatible candidate matches across layers and aggregation schemes.

4 Results

Table 3: Validation Performance

Dataset	Model	Wass. Layers	Agg	Disj	Prec	Rec	F1
FAUST	HyperGCT	N/A	0.430	0.044	0.972	0.985	0.978
	WHNN	0,1	0.415	0.049	0.973	**0.993**	0.983
	WHNN	4,5	0.436	**0.072**	0.980	0.983	0.981
	WHNN	0,5	0.431	0.046	**0.992**	0.975	0.983
	WHNN	0,1,2,3,4,5	**0.478**	0.036	0.991	0.991	**0.991**
PartNet (rigid)	HyperGCT	N/A	0.444	0.034	0.875	0.979	0.917
	WHNN	0,1	**0.557**	0.066	0.930	0.992	0.959
	WHNN	4,5	0.548	0.064	**0.988**	**0.999**	**0.993**
	WHNN	0,5	0.551	**0.113**	0.986	0.998	0.992
PartNet (1.5×)	HyperGCT	N/A	0.445	0.028	0.762	0.914	0.823
	WHNN	0,1	**0.514**	**0.077**	0.885	**0.993**	0.932
	WHNN	4,5	0.486	0.046	**0.984**	0.944	**0.959**
	WHNN	0,5	0.510	0.074	0.883	0.992	0.932
PartNet (2.0×)	HyperGCT	N/A	0.445	0.024	0.735	0.859	0.782
	WHNN	0,1	**0.522**	0.055	0.880	0.938	0.900
	WHNN	4,5	0.438	0.031	**0.991**	0.666	0.776
	WHNN	0,5	0.493	**0.073**	0.976	**0.940**	**0.956**

Table 4: Alignment L1 errors for aggregation configurations. For each dataset and model, we report the better L1 result obtained from greedy expansion or two-stage spectral matching. Where uncertainty is shown, values are reported as mean ± standard deviation over 10 evaluation runs with different random seeds.

Dataset	Model	Wass. Layers	L1 (before)	L1 (after)	Improvement
FAUST	WHNN	0,1,2,3,4,5	0.181	$\mathbf{4.1 \times 10^{-5}}$	**1.000 ± 0.0004**
	WHNN	0,5	0.180	5.7×10^{-4}	0.997 ± 0.0004
	HyperGCT	N/A	0.180	0.016	0.914 ± 0.0004
	ICP	N/A	0.180	0.059	0.676 ± 0.0004
PartNet (rigid)	WHNN	0,5	0.237	3.8×10^{-4}	0.998 ± 0.0004
	WHNN	4,5	0.237	6.3×10^{-4}	0.997 ± 0.0005
	HyperGCT	N/A	0.237	0.002	0.992 ± 0.0020
	ICP	N/A	0.237	**0.000**	**1.000 ± 0.0020**
PartNet (1.5×)	WHNN	0,1	0.248	**0.003**	**0.989 ± 0.0010**
	WHNN	0,5	0.248	0.004	0.984 ± 0.0003
	HyperGCT	N/A	0.248	0.010	0.958 ± 0.0012
	ICP	N/A	0.248	0.029	0.883 ± 0.0012
PartNet (2.0×)	WHNN	0,5	0.261	**0.006**	**0.975 ± 0.0002**
	WHNN	0,1	0.261	0.010	0.960 ± 0.0008
	HyperGCT	N/A	0.261	0.035	0.867 ± 0.0015
	ICP	N/A	0.261	0.059	0.774 ± 0.0015

Table 3 reports representative aggregation configurations in the main text, with the full ablation deferred to the appendix. Table 4 reports final alignment quality using L1 error, where each entry reflects the stronger result from greedy expansion or two-stage spectral matching under the common evaluation rule described in the Evaluation Metrics subsection.

Overall, the comparison against HyperGCT shows that Wasserstein aggregation improves the learned model in every evaluated regime for at least one layer-placement strategy. On **FAUST**, treated here as a piecewise-rigid setting, the strongest result is achieved by **All layers**. This configuration gives the best learned alignment performance and the strongest overall correspondence quality among the learned variants. On **PartNet (rigid)**, the strongest learned configuration is **Last 2 layers**, which improves on the mean-pooling HyperGCT baseline, although ICP remains nearly perfect in this easiest regime.

Under affine perturbations, the most effective placement depends more strongly on the setting. On **PartNet (1.5× affine)**, **Last 2 layers** gives the strongest correspondence-level result in the main table, while **First 2 layers** yields the best final L1 alignment among the learned variants. On **PartNet (2.0× affine)**, **First + Last** provides the strongest overall learned performance relative to HyperGCT. Taken together, these comparisons show that

Wasserstein aggregation is beneficial relative to the HyperGCT baseline, but that its gain depends on where it is introduced in the network.

The structural diagnostics broadly support the same interpretation. In several settings, the configurations that improve most over HyperGCT in downstream alignment also show stronger aggregatedness and disjointness, suggesting that Wasserstein aggregation helps organize compatible correspondences more cleanly within the learned hypergraph. However, the relationship is not strictly monotonic: the configuration with the strongest structural diagnostics is not always the one with the best final L1 result. These diagnostics are therefore useful for interpretation, but they should not be treated as complete proxies for end-task alignment quality.

5 Discussion

Our results show that replacing mean pooling with Wasserstein aggregation can improve both correspondence quality and final alignment accuracy in hypergraph-based point cloud registration. Across the evaluated datasets, at least one Wasserstein-based configuration outperforms the mean-pooling HyperGCT baseline in each regime, supporting the central hypothesis of the project: treating a neighborhood as a distribution rather than collapsing it to a simple average can preserve more useful local geometric information for matching and alignment.

A central pattern in the experiments is that the effect of Wasserstein aggregation depends strongly on where it is inserted in the network. On FAUST, applying Wasserstein aggregation at all layers gives the strongest result, suggesting that in this piecewise-rigid regime, preserving distributional structure throughout the feature hierarchy is beneficial. On PartNet, however, the best-performing configuration changes across rigid and affine perturbation regimes. In the rigid setting, the strongest learned variant is the late-layer configuration, while under stronger affine perturbations the optimal placement becomes more regime-dependent. This suggests that Wasserstein aggregation is not simply a universally better replacement for mean pooling at every depth, but rather a mechanism whose effect depends on the type and severity of geometric variation.

One possible explanation is that earlier and later layers capture different kinds of information. Earlier layers likely preserve finer local geometric detail, so Wasserstein aggregation there may help avoid blurring neighborhood structure too early. Later layers operate on more abstract relational features and may benefit from Wasserstein aggregation when the task depends more on consolidating higher-level consistency before correspondence prediction. The fact that all-layer aggregation performs best on FAUST but not uniformly on PartNet suggests that the best aggregation strategy depends on the transformation regime rather than following a single fixed rule.

The structural metrics provide additional evidence that the gains are meaningful at the representation level. In several settings, stronger downstream performance is accompanied by improved diagnostics such as aggregatedness and disjointness, indicating that Wasserstein aggregation can produce cleaner hyperedge structure. At the same time, these relation-

ships are not perfectly monotonic. The best structural scores do not always yield the best final alignment result, which suggests that these diagnostics are informative indicators of representation quality but not complete proxies for end-task success.

Another important observation is that the learned models do not outperform ICP in the easiest rigid setting. This does not weaken the core contribution of the project. ICP is highly effective when the alignment problem is nearly ideal and geometric correspondences are relatively easy to recover. The more meaningful comparison is against the learned HyperGCT baseline, where Wasserstein aggregation consistently improves performance. In this sense, the results show that the proposed modification strengthens the learning-based model even if it does not replace classical registration methods in every regime.

Taken together, the experiments show that Wasserstein aggregation can strengthen hypergraph-based alignment, but that its effect depends meaningfully on architectural placement and dataset regime. More broadly, the results suggest that the choice of aggregation operator should be treated as an important modeling decision rather than a fixed implementation detail.

6 Limitations

This study has several limitations. First, we evaluate only a limited number of layer-placement strategies, so the current results should be interpreted as evidence that placement matters rather than as a complete exploration of the design space. Additional experiments with more systematic placement patterns would be needed to determine whether the observed trends generalize beyond the specific configurations tested here.

Second, Wasserstein-based aggregation is computationally more expensive than mean pooling, but this report does not quantify the runtime or memory overhead introduced by the proposed modification. Since practical adoption depends not only on accuracy but also on efficiency, future work should compare the performance gains against the added computational cost.

Third, the empirical evaluation is limited to FAUST and several PartNet transformation regimes. Although these datasets provide useful contrast across piecewise-rigid, rigid, and affine settings, they do not fully capture the diversity of real-world registration challenges such as heavy noise, partial overlap, outliers, or large-scale scenes. As a result, the conclusions of this report should be viewed as promising but still limited in scope.

Finally, while the structural diagnostics help explain how the representations change under Wasserstein aggregation, they do not fully explain why one configuration performs better than another in final alignment. This means the report identifies useful empirical patterns, but the mechanism linking structural quality to downstream registration accuracy is still only partially understood.

7 Conclusion

This project examined whether replacing mean pooling with Wasserstein aggregation strengthens hypergraph-based point cloud alignment. Across FAUST and multiple PartNet regimes, the results show that Wasserstein aggregation can improve the learned HyperGCT pipeline, but that its benefit is not uniform across network depth or transformation setting. The strongest configuration varies by regime: FAUST benefits most from all-layer replacement, while PartNet favors different placements depending on whether the perturbation is rigid or affine.

Taken together, these results support two main conclusions. First, neighborhood aggregation matters: preserving distributional structure can improve both correspondence prediction and final alignment relative to standard mean pooling. Second, the placement of that aggregation matters as well: the same operator can behave differently depending on where it is introduced in the network. This makes aggregation design an architectural variable worth tuning rather than treating as a fixed implementation choice.

Future work should expand the placement search, measure runtime and memory costs directly, and test the method in more challenging registration settings such as partial overlap, heavier noise, and larger scenes.

References

Aoki, Yasuhiro, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. 2019. "PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF. [Link]

Besl, Paul J., and Neil D. McKay. 1992. "A Method for Registration of 3-D Shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14(2): 239–256. [Link]

Bogo, Federica, Javier Romero, Matthew Loper, and Michael J. Black. 2014. "FAUST: Dataset and Evaluation for 3D Mesh Registration." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. [Link]

Chen, Yang, and Gérard Medioni. 1992. "Object Modelling by Registration of Multiple Range Images." *Image and Vision Computing* 10(3): 145–155. [Link]

Cuturi, Marco. 2013. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport." In *Advances in Neural Information Processing Systems*. [Link]

Duta, Iulia, and Pietro Liò. 2025. "Wasserstein Hypergraph Neural Network." *arXiv preprint*. [Link]

Feng, Yifan, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. "Hypergraph Neural Networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*. [Link]

Kipf, Thomas N., and Max Welling. 2017. "Semi-Supervised Classification with Graph Convolutional Networks." *International Conference on Learning Representations (ICLR)*. [Link]

Kolouri, Soheil, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. 2019. "Generalized Sliced Wasserstein Distances." In *Advances in Neural Information Processing Systems*. [Link]

Mo, Kaichun, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. 2019. "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. [Link]

Peyré, Gabriel, and Marco Cuturi. 2019. "Computational Optimal Transport." *Foundations and Trends® in Machine Learning*. [Link]

Qi, Charles R., Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. [Link]

Villani, Cédric. 2009. *Optimal Transport: Old and New*. Springer. [Link]

Wang, Yue, and Justin M. Solomon. 2019. "Deep Closest Point: Learning Representations for Point Cloud Registration." In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE/CVF. [Link]

Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. "Dynamic Graph CNN for Learning on Point Clouds." *ACM Transactions on Graphics* 38(5): 146:1–146:12. [Link]

Yew, Zi Jian, and Gim Hee Lee. 2020. "RPM-Net: Robust Point Matching Using Learned Features." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE/CVF. [Link]

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. 2017. "Deep Sets." In *Advances in Neural Information Processing Systems*. [Link]

Zhang, X. et al. 2025. "HyperGCT: A Dynamic Hyper-GNN-Learned Geometric ConstrainT for 3D Registration." In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Computer Vision Foundation. [Link]

Appendices

A.1 Training Details A1
A.2 Additional Tables A2

A.1 Training Details

Table A 1: Dataset-specific training settings.

Dataset	Data root	Batch	Workers	Neg. ratio	Inlier thr.	Val. ratio
FAUST	`data/faust/fpfh`	8	8	2	0.05	0.15
PartNet (rigid)	`data/partnet/fpfh_rigid`	8 or 16[†]	12	2	0.10	0.10
PartNet (1.5×)	`data/partnet/fpfh_affine_1-5`	8 or 16[†]	12	2	0.10	0.10
PartNet (2.0×)	`data/partnet/fpfh_affine_2-0`	8 or 16[†]	12	2	0.10	0.10

[†] For Wasserstein models, the majority of the single-layer replacement runs use batch size 16, while the multiple-layer replacement runs use batch size 8 due to memory differences.

Table A 1 reports dataset-specific settings used in training and validation. These details supplement the shared hyperparameters listed in the main text.

A.2 Additional Tables

Table A 2: Validation performance — full table

Dataset	Model	Wass. Layers	Agg	Disj	Class	Graph	SM	Prec	Rec	F1
FAUST	HyperGCT	N/A	0.430	0.044	0.050	0.066	0.011	0.972	0.985	0.978
	WHNN	0	0.425	0.044	0.038	0.053	0.013	0.983	0.986	0.985
	WHNN	0,1	0.415	0.049	0.041	0.055	0.017	0.973	**0.993**	0.983
	WHNN	5	0.435	0.024	0.046	0.119	0.022	0.984	0.977	0.980
	WHNN	4,5	0.436	**0.072**	0.041	0.073	0.014	0.980	0.983	0.981
	WHNN	0,5	0.431	0.046	0.037	0.058	0.011	**0.992**	0.975	0.983
	WHNN	0,1,2,3,4,5	**0.478**	0.036	**0.021**	**0.048**	**0.008**	0.991	0.991	**0.991**
PartNet (rigid)	HyperGCT	N/A	0.444	0.034	0.124	0.200	0.024	0.875	0.979	0.917
	WHNN	0	0.507	0.044	0.153	0.252	0.030	0.900	0.928	0.901
	WHNN	0,1	**0.557**	0.066	0.085	0.114	0.018	0.930	0.992	0.959
	WHNN	5	0.457	0.021	0.036	0.166	0.008	0.976	0.992	0.983
	WHNN	4,5	0.548	0.064	**0.018**	**0.054**	**0.003**	**0.988**	**0.999**	**0.993**
	WHNN	0,5	0.551	**0.113**	0.020	0.100	0.004	0.986	0.998	0.992
	WHNN	0,1,2,3,4,5	0.539	0.021	0.056	0.111	0.006	0.973	0.980	0.976
PartNet (1.5×)	HyperGCT	N/A	0.445	0.028	0.236	0.256	0.049	0.762	0.914	0.823
	WHNN	0	0.496	0.041	0.284	0.385	0.093	0.807	0.843	0.806
	WHNN	0,1	**0.514**	**0.077**	0.136	0.188	0.027	0.885	**0.993**	0.932
	WHNN	5	0.459	0.029	0.124	0.247	0.026	0.911	0.967	0.937
	WHNN	4,5	0.486	0.046	**0.077**	**0.168**	**0.014**	**0.984**	0.944	**0.959**
	WHNN	0,5	0.510	0.074	0.134	0.186	0.026	0.883	0.992	0.932
	WHNN	0,1,2,3,4,5	0.446	0.038	0.162	0.222	0.030	0.879	0.993	0.926
PartNet (2.0×)	HyperGCT	N/A	0.445	0.024	0.282	0.254	0.069	0.735	0.859	0.782
	WHNN	0	0.483	0.034	0.326	0.395	0.084	0.792	0.900	0.827
	WHNN	0,1	**0.522**	**0.055**	0.216	0.315	0.045	0.880	**0.938**	0.900
	WHNN	5	0.451	0.026	0.213	0.280	0.043	0.956	0.804	0.868
	WHNN	4,5	0.438	0.031	0.403	**0.197**	0.058	**0.991**	0.666	0.776
	WHNN	0,5	0.493	0.073	**0.088**	0.215	**0.016**	0.976	0.940	**0.956**
	WHNN	0,1,2,3,4,5	0.492	0.045	0.205	0.283	0.030	0.980	0.857	0.904

Table A 3: Alignment L1 errors for aggregation configurations — full table.

Dataset	Model	Wass. Layers	L1 (before)	L1 (after)	Improvement
FAUST	WHNN	0,1,2,3,4,5	0.181	$\mathbf{4.1 \times 10^{-5}}$	$\mathbf{1.000 \pm 0.0004}$
	WHNN	0,1	0.180	0.001	0.992 ± 0.0008
	WHNN	4,5	0.180	0.016	0.910 ± 0.0003
	WHNN	0,5	0.180	5.7×10^{-4}	0.997 ± 0.0004
	WHNN	0	0.180	0.002	0.990 ± 0.0003
	WHNN	5	0.180	0.001	0.994 ± 0.0003
	HyperGCT	N/A	0.180	0.016	0.914 ± 0.0004
	ICP	N/A	0.180	0.059	0.676 ± 0.0004
PartNet (rigid)	WHNN	0,1,2,3,4,5	0.237	0.009	0.962 ± 0.0025
	WHNN	0,1	0.237	0.001	0.995 ± 0.0022
	WHNN	4,5	0.237	6.3×10^{-4}	0.997 ± 0.0005
	WHNN	0,5	0.237	$\mathbf{3.8 \times 10^{-4}}$	$\mathbf{0.998 \pm 0.0004}$
	WHNN	0	0.237	0.002	0.993 ± 0.0014
	WHNN	5	0.237	8.2×10^{-4}	0.997 ± 0.0036
	HyperGCT	N/A	0.237	0.002	0.992 ± 0.0020
	ICP	N/A	0.237	0.000	1.000 ± 0.0020
PartNet (1.5×)	WHNN	0,1,2,3,4,5	0.248	0.004	0.983 ± 0.0008
	WHNN	0,1	0.248	0.003	$\mathbf{0.989 \pm 0.0010}$
	WHNN	4,5	0.248	0.005	0.980 ± 0.0006
	WHNN	0,5	0.248	0.004	0.984 ± 0.0003
	WHNN	0	0.248	0.007	0.973 ± 0.0014
	WHNN	5	0.248	0.005	0.980 ± 0.0006
	HyperGCT	N/A	0.248	0.010	0.958 ± 0.0012
	ICP	N/A	0.248	0.029	0.883 ± 0.0012
PartNet (2.0×)	WHNN	0,1,2,3,4,5	0.261	0.018	0.930 ± 0.0010
	WHNN	0,1	0.261	0.010	0.960 ± 0.0008
	WHNN	4,5	0.261	0.013	0.951 ± 0.0009
	WHNN	0,5	0.261	0.006	$\mathbf{0.975 \pm 0.0002}$
	WHNN	0	0.261	0.015	0.943 ± 0.0009
	WHNN	5	0.261	0.012	0.954 ± 0.0009
	HyperGCT	N/A	0.261	0.035	0.867 ± 0.0015
	ICP	N/A	0.261	0.059	0.774 ± 0.0015