Diffusion-assisted Matching for 6D Pose Estimation and Registration

Anonymous authors
Paper under double-blind review

Abstract

This paper presents DiffMatcher, a new diffusion-based framework for object-centric point cloud registration and pose estimation. The proposed pipeline estimates and refines a local and global geometric consistency-aware soft assignment matrix between source and target point clouds. The iterative refinement is driven by timestep-dependent spatial coordinate encodings and noise-level embeddings that enrich the backbone features and improve their discriminability across noise scales. As a result, the model first reconstructs coarse geometric structures in high-noise regimes and subsequently refines them with increasing precision over iterations. The framework includes a forward process during training, where noise is added to the ground-truth score matrix to guide denoising, and a reverse process during inference, where the model starts from a noisy score matrix and gradually recovers the target matrix. Experiments on real-world 6D object pose estimation, synthetic object-centric point cloud registration, and scene-level benchmarks show significant improvements in point matching and pose estimation, especially for low-overlap and noisy point clouds.

1 INTRODUCTION and RELATED WORK

The increasing use of 3D data, mainly in form of point clouds captured directly from 3D sensors or generated through estimation (reconstruction), has significantly impacted various fields within robotics and computer vision (Zhu et al., 2024). Accurately capturing environmental geometry in 3D is essential for numerous robotics applications, such as object grasping and manipulation (Jiang et al., 2024), visual servoing (Adjigble et al., 2023), Simultaneous Localization and Mapping (SLAM) (Shi et al., 2021), and robot-assisted surgery (Liu et al., 2022). Advancements in 3D point data processing have presented a critical challenge for the robotics field: Point Cloud Registration (PCR). The objective of PCR is to estimate the rigid transformation that aligns two overlapping point clouds, often captured by different sensors or from various viewpoints, into a standard coordinate system while minimizing both translation and rotation errors.

To meet these challenges, recent research has investigated the use of diffusion models for PCR (Wu et al., 2023; 2024; Jiang et al., 2024), building on their remarkable success across computer vision tasks (Croitoru et al., 2023). Diffusion models are highly advanced deep generative models that involve two core phases to generate data (Song & Ermon, 2019; Ho et al., 2020). The first phase, the forward diffusion process, progressively injects Gaussian perturbations into the data sample forming a Markov chain. During the second core phase, the reverse process, the model learns to recover clean data from noisy data through a reverse Markov chain. During inference, the model starts from pure Gaussian noise and iteratively applies the learned denoising steps to synthesize a data sample that resembles the original data distribution.

In the context of PCR, most previous methods focus directly on denoising the transformation matrix (Wu et al., 2023). Specifically, the authors in (Jiang et al., 2024) provided significant improvements to the learning baselines (*i.e*, DCP (Wang & Solomon, 2019a) and RPMNet (Yew & Lee, 2020)). To determine the 6 DOF pose, a forward and backward diffusion process is utilized within transformation learning, leveraging the Lie algebra linked to the space $SE(3)$. Practically, the denoising network preserves the original DCP framework and estimates the transformation through a soft singular value decomposition (SVD) without adding extra components. In contrast, PCRDiffusion (Wu et al., 2023) predicts the transformation matrix

through a latent-space regression module that serves as a denoiser. Diff-Reg (Wu et al., 2024) estimates soft point correspondences by formulating a denoising diffusion model over the space of non-square doubly stochastic matrices, tailored for cross-modal matching. Specifically, Diff-Reg begins with a binary ground-truth assignment matrix, which is progressively perturbed through the forward diffusion process. During the reverse process, a transformer-based denoising network is employed, and Sinkhorn projections (Cuturi, 2013) are integrated to ensure that the recovered matrices remain approximately doubly stochastic.

Inspired by recent diffusion-based registration methods, we propose DiffMatcher, a standalone diffusion-guided soft matching framework for correspondence learning. Unlike prior matching baselines such as iMatcher Slimani et al. (2025), DiffMatcher formulates the correspondence matrix prediction as an iterative denoising process. DiffMatcher introduces two key components, Noise-Conditioned Feature Embedding (NCFE) and Correspondence Fourier Embedding (CFE), to condition correspondence estimation on the diffusion timestep and the evolving geometric state. In the proposed architecture, the denoising process is guided along a noise-informed path, providing updated inputs at each step. This approach enables the model to refine its predictions while leveraging local geometric consistency and first-order spatial compatibility cues at every diffusion step. The thorough explanation of the suggested diffusion process is detailed in Sec. 3.

2 PROBLEM FORMULATION

Given source and target point clouds $\boldsymbol{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3 | i = 1 \dots M\}$ and $\boldsymbol{Y} = \{\boldsymbol{x}_i \in \mathbb{R}^3 | i = 1 \dots N\}$, point cloud registration (PCR) aims to estimate the rigid transformations $\mathbf{T} = \{\mathbf{R}, \mathbf{t}\}$ that best aligns the two point clouds, with $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ denote the desired rotation matrix and translation vector, respectively, which solve the equation:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i) \in \mathcal{C}} \|\mathbf{R}\hat{\boldsymbol{x}}_i + \mathbf{t} - \hat{\boldsymbol{y}}_i\|^2 \tag{1}$$

where $\mathcal{C}$ is the correspondence set between source and target point clouds. Due to the unordered nature of 3D point clouds, $\mathcal{C}$ is unknown in practice and its estimation is a crucial subproblem for PCR methods. Most learning-based methods (Wang & Solomon, 2019a; Yew & Lee, 2020; Zhang et al., 2022) are trained to predict a soft correspondence matrix $\boldsymbol{S} \in \mathbb{R}^{M \times N}$, where each entry $\boldsymbol{S}_{ij}$ represents the probability of pairing a point $\boldsymbol{x}_i \in \boldsymbol{X}$ with a point $\boldsymbol{y}_j \in \boldsymbol{Y}$. Typically, this is achieved by applying a softmax over the scalar projections of target features onto source features (Wang & Solomon, 2019a;b). Some methods iteratively refine the score matrix using the Sinkhorn algorithm (Cuturi, 2013; Shi et al., 2021). In contrast, VRNet (Zhang et al., 2022) introduces Virtual Corresponding Points (VCPs) that preserve the source shape by rectifying occluded or missing parts in the target space.

3 PROPOSED METHOD: a DIFFUSION MODEL for SOFT MATCHING

Following the standard pipeline of generative diffusion models (Song & Ermon, 2019; Ho et al., 2020), we propose a two-stage framework comprising a forward process and a reverse denoising process for score matrix prediction, aiming to incrementally generate an optimal score matrix $\boldsymbol{S}^0$ given a noisy starting matrix $\boldsymbol{S}^{(T_s)}$. This process in practice involves two main steps.

The forward process incrementally introduces Gaussian disturbances to the initial correspondence matrix $\boldsymbol{S}^{(0)}$, resulting in a completely noisy variant $\boldsymbol{S}^{(T_s)}$ after T_s steps. Formally, the conditional transition from timesteps $t-1$ to t is noted as $q_1(\boldsymbol{S}^{(t)} \mid \boldsymbol{S}^{(t-1)})$, which denotes the probability of generating $\boldsymbol{S}^{(t)}$ from $\boldsymbol{S}^{(t-1)}$ via a Gaussian distribution.

During the reverse process, the model starts from a sample of pure noise, denoted by $\boldsymbol{S}^{(T_s)}$, and iteratively denoises it to estimate the clean correspondence matrix $\boldsymbol{S}^{(0)}$. Formally, the objective is to approximate the reverse conditional distribution $q_2\big(\boldsymbol{S}^0 \mid \boldsymbol{S}^{(T_s)}\big)$, which describes the probability of recovering the clean data given the noisy input.

To achieve this, the model estimates the conditional transitions between each pair of consecutive timesteps $q_2\big(\boldsymbol{S}^{(t-1)} \mid \boldsymbol{S}^{(t)}\big)$. Each of these transitions represents the probability of generating $\boldsymbol{S}^{(t-1)}$ from $\boldsymbol{S}^{(t)}$ and is modeled as a Gaussian distribution.

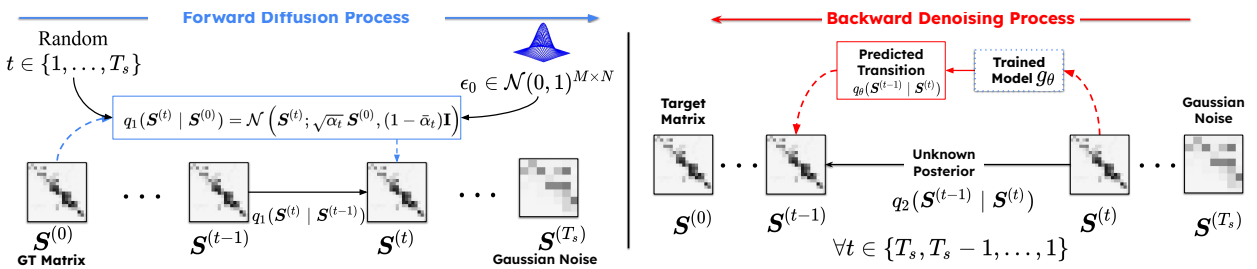

Figure 1: Illustration of the diffusion framework. (**Left**) Forward diffusion process: Gaussian noise is progressively added to the clean correspondence matrix $\boldsymbol{S}^{(0)}$ over T_s steps, producing $\boldsymbol{S}^{(T_s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. (**Right**) Reverse denoising process: starting from pure noise, the model iteratively recovers the clean correspondence matrix by sampling from the learned reverse transitions $q_\theta(\boldsymbol{S}^{(t-1)} \mid \boldsymbol{S}^{(t)})$.

A neural network, denoted by g_θ, is used to approximate these reverse transitions by predicting a parameterized distribution $q_\theta\big(\boldsymbol{S}^{(t-1)} \mid \boldsymbol{S}^{(t)}\big)$, which serves as an estimate of the true reverse conditional $q_2\big(\boldsymbol{S}^{(t-1)} \mid \boldsymbol{S}^{(t)}\big)$. The forward and reverse process are illustrated in Fig. 1, and detailed in Sec. 3.1 and 3.2 respectively.

3.1 Training Phase

As illustrated Fig. 1, given the ground-truth correspondence matrix $\boldsymbol{S}^{(0)} \in [0,1]^{M\times N}$, the forward noising process adds Gaussian perturbations at each timestep to generate a sequence $\boldsymbol{S}^{(1)}, \ldots, \boldsymbol{S}^{(T_s)}$ that forms a Markov chain of increasingly noisy samples. The conditional transition from timestep $t-1$ to t is defined by:

$$q_1(\boldsymbol{S}^{(t)} \mid \boldsymbol{S}^{(t-1)}) := \mathcal{N}(\boldsymbol{S}^{(t)}; \sqrt{1-\beta_t}\,\boldsymbol{S}^{(t-1)}, \beta_t \mathbf{I}) \tag{2}$$

where $\beta_1, \ldots, \beta_{T_s} \in [0,1)$ are the variance schedule parameters, computed using a cosine schedule as proposed in (Nichol & Dhariwal, 2021). Or equivalently:

$$\boldsymbol{S}^{(t)} = \sqrt{1-\beta_t}\,\boldsymbol{S}^{(t-1)} + \sqrt{\beta_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \in \mathcal{N}(0,1)^{M\times N} \tag{3}$$

As demonstrated in DDPM (Ho et al., 2020), Eq. (2) allows expressing the random variable $\boldsymbol{S}^{(t)} \sim q_1(\boldsymbol{S}^{(t)} \mid \boldsymbol{S}^{(t-1)})$ directly from the initial correspondence matrix $\boldsymbol{S}^0$ as follows:

$$q_1(\boldsymbol{S}^{(t)} \mid \boldsymbol{S}^{(0)}) := \mathcal{N}\left(\boldsymbol{S}^{(t)}; \sqrt{\bar{\alpha}_t}\,\boldsymbol{S}^{(0)}, (1-\bar{\alpha}_t)\mathbf{I}\right) \tag{4}$$

where $\bar{\alpha}_t = \prod_{s=1}^{(t)} \alpha_s = \prod_{s=1}^{(t)} (1-\beta_s)$ denotes the cumulative product of the noise schedule. This closed-form expression defines the marginal distribution of $\boldsymbol{S}^{(t)}$ conditioned on the initial correspondence matrix $\boldsymbol{S}^{(0)}$. It allows direct sampling at any timestep without recursively computing intermediate states:

$$\begin{aligned} \boldsymbol{S}^{(t)} \sim q_1(\boldsymbol{S}^{(t)} \mid \boldsymbol{S}^{(0)}) &= \sqrt{\bar{\alpha}_t}\,\boldsymbol{S}^{(0)} + \sqrt{1-\bar{\alpha}_t}\,f_\epsilon(\boldsymbol{\epsilon}), \\ \boldsymbol{\epsilon} &\in \mathcal{N}(0,1)^{M\times N} \end{aligned} \tag{5}$$

The transformation function$f_\epsilon = (\epsilon \bmod 1) \cdot \frac{|\epsilon|}{\epsilon} \cdot \upsilon$ introduced in (Wu et al., 2024), maps the added noise ϵ approximately to the interval $[0,1]$. This transformation preserves both the magnitude and the sign of the noise while normalizing its range. The scaling factor $\upsilon = 1.5$ is chosen empirically.

The expression in Eq. (5) is crucial in practice, as it enables sampling $\boldsymbol{S}^{(t)}$ at a randomly chosen timestep t in a single operation. This eliminates the need for computationally expensive sequential sampling of the form $\boldsymbol{S}^{(0)} \to \boldsymbol{S}^{(1)} \to \cdots \to \boldsymbol{S}^{(t)}$, significantly improving efficiency during training.

The training consists of learning a denoising network g_θ able to predict the correspondence matrix S^0 at any timestep t, from the noised matrix $S^{(t)}$ and learned features as follows:

$$\boldsymbol{S}_{est} = g_\theta(\boldsymbol{S}^{(t)}, t, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{F}_x, \boldsymbol{F}_y) \tag{6}$$

with $\boldsymbol{F}_x$ and $\boldsymbol{F}_y$ are the source and target features.

3.2 Inference Phase

During inference, we adopt a reverse process to generate the optimal soft correspondence matrix by simulating the reverse diffusion process. This begins from a pure noise sample $\boldsymbol{S}^{(T_s)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, conditioned by the data, and iteratively denoises it over T_s steps, using a learned neural network g_θ to obtain a refined estimate of the original correspondence matrix. Specifically, estimating the optimal correspondence matrix $\boldsymbol{S}^{(0)}$ from a Gaussian noise matrix $\boldsymbol{S}^{(T_s)}$ involves inverting the Markov chain and assessing the sequence:

$$\boldsymbol{S}^{(T_s)} \to \boldsymbol{S}^{(T_s-1)} \to \cdots \to \boldsymbol{S}^{(0)}$$

Using the denoising network g_θ, we predict $\boldsymbol{S}_{est} = g_\theta(\boldsymbol{S}^{(t)}, t, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{F}_x, \boldsymbol{F}_y)$. Eq. (5), estimates the noise added between $S^{(t-1)}$ and $S^{(t)}$:

$$\epsilon_\theta(\boldsymbol{S}^{(t)}, t) = \frac{\boldsymbol{S}^{(t)} - \sqrt{\bar{\alpha}_t}\, \boldsymbol{S}_{est}}{\sqrt{1-\bar{\alpha}_t}} \tag{7}$$

Denoising Diffusion Implicit Model (DDIM) (Song et al., 2020) introduces a parameter $\eta \in [0, 1]$ in Eq. (3) that controls the stochasticity of the sampling process and allows computing $S^{(t-1)}$ from $S^{(t)}$:

$$\boldsymbol{S}^{(t-1)} = \sqrt{\bar{\alpha}_{t-1}} \cdot \boldsymbol{S}_{est} + \sqrt{1 - \bar{\alpha}_{t-1} - \eta^2 \sigma_t^2} \cdot \epsilon_\theta(\boldsymbol{S}^{(t)}, t) + \eta \sigma_t \cdot \boldsymbol{z}_t, \quad \boldsymbol{z}_t \sim \mathcal{N}(0, \mathbf{I}) \tag{8}$$

with $\sigma_t \leftarrow \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \cdot \sqrt{1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}$.

This rule is then applied during inference to incrementally recover the optimal target correspondence matrix after T_s iterations (*i.e*,, time steps) with a fixed $\eta = 1$.

3.3 Optimization

Computing the log-likelihood of the data directly under the diffusion model is intractable, as it requires integrating over all latent variables:

$$\log q_\theta(\boldsymbol{S}^{(0)}) = \log \int q_\theta(\boldsymbol{S}^{(0:T_s)})\, d\boldsymbol{S}^{(1:T_s)} \tag{9}$$

To address this, we follow prior work on diffusion models (Ho et al., 2020; Jiang et al., 2024; Wu et al., 2024) and optimize a variational lower bound on the log-likelihood, known as the *Evidence Lower Bound* (ELBO):

$$\log q_\theta(\boldsymbol{S}^{(0)}) \geq \mathbb{E}_{q(\boldsymbol{S}^{(1:T_s)} | \boldsymbol{S}^{(0)})} \left[\log \left(\frac{q_\theta(\boldsymbol{S}^{(0:T)})}{q(\boldsymbol{S}^{(1:T_s)} \mid \boldsymbol{S}^{(0)})} \right) \right] \tag{10}$$

This bound can be simplified into a practical training loss (Wu et al., 2024):

$$\mathcal{L}_{\text{simple}} = -\mathbb{E}_{q(\boldsymbol{S}^{(0)})} \left[\sum_{t=1}^{(T_s)} \mathbb{E}_{q_1(\boldsymbol{S}^{(t)} | \boldsymbol{S}^{(0)})} \log q_\theta(\boldsymbol{S}^{(0)} \mid \boldsymbol{S}^{(t)}) \right] \tag{11}$$

This objective corresponds to a noise-weighted mean-squared error loss that trains the model to denoise $\boldsymbol{S}^{(t)}$ back to the original clean data $\boldsymbol{S}^{(0)}$ across varying noise levels. In practice, we minimize the gap loss from (Shi et al., 2021) defined as:

$$L_{\text{Gap}} = \sum_{i=1}^{M} \ell_i + \sum_{j=1}^{N} \tilde{\ell}_j. \tag{12}$$

with:

$$\ell_i = \log\left(\sum_{n=1}^{N+1} in + 1\right) \tag{13a}$$

$$\tilde{\ell}_j = \log\left(\sum_{n=1}^{M+1} jn + 1\right) \tag{13b}$$

where α is a positive scalar set to 0.5, and $\boldsymbol{S}_{est_{i,\bar{i}}}$ and $\boldsymbol{S}_{est_{j,\bar{j}}}$ are the scores for the ground truth true matches of the points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, respectively.

3.4 Proposed Denoising Network g_θ

Feature Extraction. Starting from the source and target point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$, we follow DCP (Wang & Solomon, 2019a) and adopt DGCNN (Wang et al., 2019) as the backbone to project each point into a discriminative descriptor space. DGCNN constructs a local graph by connecting each point to its k nearest neighbors (kNN), then applies edge convolution layers to aggregate high-dimensional graph features. This procedure is repeated by dynamically updating the graphs and recomputing kNN relationships in the feature space generated by the previous layer. The resulting features are then fed into a Transformer (Vaswani et al., 2017) to capture global contextual cues and enable information exchange between the two point clouds. We denote the obtained features as $\boldsymbol{F}_x = \{\boldsymbol{f}_{x_i} \in \mathbb{R}^d | i = 1 \dots M\}$ and $\boldsymbol{F}_y = \{\boldsymbol{f}_{y_i} \in \mathbb{R}^d | i = 1 \dots N\}$ for source and target points, respectively, with d the feature dimension. These steps are performed once, and the obtained features feed into the iterative denoising network described below.

Noise-Conditioned Feature Embedding (NCFE). To help the denoising block capture the level of noise at each iteration, we propose associating the features $\boldsymbol{F}_x$ and $\boldsymbol{F}_y$ with a time step embedding. To do so, we encode the current time step $t \in \{1, \dots, T_s\}$ by a frequency embedding in the same spirit as positional encoding from (Vaswani et al., 2017). This operation is crucial for guiding the model in its trajectory of denoising the score matrix, focusing on the coarse structure for large timesteps and refining it as noise levels decrease. For a given timestep t, the positional encoding is defined as a vector $\boldsymbol{u}^{(t)} \in \mathbb{R}^d$ sharing the same dimension as the point features:

$$\boldsymbol{u}^{(t)}_{2p} = \cos\left(\frac{t}{\tau^{2p/d}}\right), \quad \boldsymbol{u}^{(t)}_{2p+1} = \sin\left(\frac{t}{\tau^{2p/d}}\right). \tag{14}$$

with p the index of each channel in the embedding vector, $p \in \{0, \dots, d/2 - 1\}$, and $\tau = 10000$. The noisy features of each source and target point $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ are then defined for each iteration as follows:

$$\boldsymbol{f}^{(t)}_{x_i} = \boldsymbol{f}_{x_i} + \boldsymbol{u}^{(t)}, \quad \boldsymbol{f}^{(t)}_{y_i} = \boldsymbol{f}_{y_i} + \boldsymbol{u}^{(t)}. \tag{15}$$

Graph CNN-based Local Consistency Enforcement (GCNN).

At this stage, DiffMatcher employs a graph-based local consistency module to aggregate neighborhood-level geometric cues. This design follows the general principle of local consistency modeling used in recent matching baselines, including iMatcher (Slimani et al., 2025). Specifically, given the source and target features, $\boldsymbol{F}^{(t)}_x$ and $\boldsymbol{F}^{(t)}_y$, we build local graphs around each point $\boldsymbol{x}_i$ by collecting their kNN in the time-conditioned feature space $\boldsymbol{F}^{(t)}_{x_i}$ and apply the *EdgeConv* operation form (Wang & Solomon, 2019a), followed by a three-layer MLP of output dimensions d, d and $d/2$ followed by a *MaxPool* over the k neighbors. The so obtained source and target features are noted $\tilde{\boldsymbol{H}}^{(t)}_x \in \mathbb{R}^{N \times d}$ and $\tilde{\boldsymbol{H}}^{(t)}_y \in \mathbb{R}^{N \times d}$ respectively.

Noisy Bilateral Matching (NBM). *iMatcher* proposes to use **GCNN** features to compute an intermediate score matrix using a dot product between source and target features. Instead, we use the ground truth correspondences matrix $\boldsymbol{S}^{(0)}$, perturbed with sampled Gaussian noise ($\boldsymbol{S}^{(t)}$) during training. This matrix is then used to estimate a coarse rigid transformation through a weighted SVD (Wang & Solomon, 2019a). The obtained noisy rotation matrix and translation vector, denoted by $\mathbf{R}^{(t)} \in \mathbb{R}^{3\times3}$ and $\mathbf{t}^{(t)} \in \mathbb{R}^3$ respectively, are used to warp the source point cloud by projecting each point $\boldsymbol{x}_i$ into $\boldsymbol{x}^{(t)}_i$ as follows:

$$\boldsymbol{x}^{(t)}_i = \mathbf{R}^{(t)}\boldsymbol{x}_i + \mathbf{t}^{(t)}, \quad \forall i = 1, \dots, N \tag{16}$$

Based on the transformed source points, bilateral pairings are constructed by associating each point with its nearest neighbor in the opposite point cloud (source-to-target and target-to-source), we note these correspondences $\boldsymbol{\mathcal{C}}_{xy}^{(t)} = \{(\boldsymbol{x}_i^{(t)}, \boldsymbol{y}_{\tilde{i}}) \;\; ; \;\; \forall\, 1 \leq i \leq M\}$ and $\boldsymbol{\mathcal{C}}_{yx}^{(t)} = \{(\boldsymbol{y}_i, \boldsymbol{x}_{\tilde{i}}^{(t)}) \;\; ; \;\; \forall\, 1 \leq i \leq N\}$. The features of each paired correspondence are concatenated, yielding source features $\hat{\boldsymbol{H}}_x^{(t)} \in \mathbb{R}^{M\times 2d}$ and target features $\hat{\boldsymbol{H}}_y^{(t)} \in \mathbb{R}^{N\times 2d}$. This step is crucial as it allows the denoising model to capture the dependence of the score matrix on the sampled noise. At each time step t, the sampled noise yields a different correspondence matrix $\boldsymbol{S}^{(t)}$. By computing the perturbed rotation $\mathbf{R}^{(t)} \in \mathbb{R}^{3\times 3}$ and translation $\mathbf{t}^{(t)} \in \mathbb{R}^3$ for the warping process, the bilateral correspondences are updated, leading to new features $\hat{\boldsymbol{H}}_x^{(t)}$ and target features $\hat{\boldsymbol{H}}_y^{(t)}$. We then reduce the dimensionality of the these concatenated features by we mapping them to $d/2$, producing:

$$\boldsymbol{H}_x^{(t)} \in \mathbb{R}^{M\times \frac{d}{2}}, \quad \boldsymbol{H}_y^{(t)} \in \mathbb{R}^{N\times \frac{d}{2}}.$$

This reduction makes room for an additional encoding, based on a *correspondence Fourier embedding*, which complements the updated features as detailed in the following section.

Correspondence Fourier Embedding (CFE). To further encourage the distinctiveness between different noise levels at each time step t, we encode the coordinates of the correspondences and incorporate their Fourier positional encoding (Qin et al., 2023; Mildenhall et al., 2021) into the model. Specifically, for each source point $\boldsymbol{x}_i$, given the coordinates of the correspondence pair $(\boldsymbol{x}_i^{(t)}, \boldsymbol{y}_{\tilde{i}})$, we first normalize them and then construct the following encoding vector:

$$\hat{\boldsymbol{e}}_{x_i}^{(t)} = [\hat{\boldsymbol{c}}_{x_i}^{(t)},\ \sin(2^{-1}\hat{\boldsymbol{c}}_{x_i}^{(t)}),\ \cos(2^{-1}\hat{\boldsymbol{c}}_{x_i}^{(t)})] \in \mathbb{R}^{18} \tag{17}$$

where $\hat{\boldsymbol{c}}_{x_i}^{(t)}$ represents the concatenation of the normalized coordinates from the correspondence pair $(\boldsymbol{x}_i^{(t)}, \boldsymbol{y}_{\tilde{i}})$. $\hat{\boldsymbol{e}}_{x_i}^{(t)}$ is then projected using a three-layer multilayer perceptron (MLP) with each layer followed by Group Normalization and a LeakyReLU activation. These new features are called $\boldsymbol{e}_{x_i}^{(t)} \in \mathbb{R}^d$. Using the same steps for target-to-source pairings leads to $\boldsymbol{e}_{y_i}^{(t)} \in \mathbb{R}^d$ for target points.

First-Order Spatial Compatibility (FOSC). Following the *iMatcher* framework, we estimate a spatial consistency encoding score for each correspondence by computing the pairwise distance deviations between all pairs in $\boldsymbol{\mathcal{C}}_{xy}^{(t)}$ and $\boldsymbol{\mathcal{C}}_{yx}^{(t)}$. For each correspondence, these deviations are averaged and weighted by a learned parameter to predict a confidence score. This confidence score is then passed through an MLP, which maps it into a $(\frac{d}{2})$-dimensional feature space, noted $\hat{\boldsymbol{V}}_x^{(t)} = \{\hat{\boldsymbol{v}}_{x_i}^{(t)} \in \mathbb{R}^{d/2} | i = 1 \dots M\}$ for source and $\hat{\boldsymbol{V}}_y^{(t)} = \{\hat{\boldsymbol{v}}_{y_i}^{(t)} \in \mathbb{R}^{d/2} | i = 1 \dots N\}$ for target.

Final assignment matrix estimation. Given the features from the previous **FOSC**, **CFE**, and **NBM** blocks, the final geometry-aware source features $\boldsymbol{V}_x^{(t)} = \{\boldsymbol{v}_{x_i}^{(t)} \in \mathbb{R}^{2d} | i = 1 \dots M\}$ are estimated by a concatenation of the output of each block:

$$\boldsymbol{v}_{x_i}^{(t)} := [\hat{\boldsymbol{v}}_{x_i}^{(t)}, \boldsymbol{h}_{x_i}^{(t)}, \boldsymbol{e}_{x_i}^{(t)}] \in \mathbb{R}^{2d} \tag{18}$$

The same operations on the target points give the associated features $\boldsymbol{V}_y^{(t)} = \{\boldsymbol{v}_{y_i}^{(t)} \in \mathbb{R}^{2d} | i = 1 \dots N\}$. The so obtained features are then used to compute an intermediate soft assignment matrix $\boldsymbol{P}^{(t)} \in \mathbb{R}^{N\times M}$ by a normalized dot product:

$$\boldsymbol{P}_{ij}^{(t)} = \left(\boldsymbol{v}_{x_i}^{(t)}\right)^\top \boldsymbol{v}_{y_i}^{(t)} \tag{19}$$

To refine the assignment matrix $\boldsymbol{P}^{(t)}$, source and target feature maps are feed into a shared MLP (with dimensions $\frac{d}{4}, \frac{d}{4}, \frac{d}{2}$ and 1), followed by a sigmoid operation to predict a final matchability score for each point as follows:

$$\tau_{x_i}^{(t)} = \text{Sigmoid}\Big(\boldsymbol{MLP}(\boldsymbol{v}_{x_i}^{(t)})\Big), \quad \tau_{y_i}^{(t)} = \text{Sigmoid}\Big(\boldsymbol{MLP}(\boldsymbol{v}_{y_i}^{(t)})\Big). \tag{20}$$

One can now estimate a *matchability matrix* $\boldsymbol{M}^{(t)} \in [0,1]^{M\times N}$as the outer product:

$$\boldsymbol{M}_{ij}^{(t)} = \tau_{x_i}^{(t)} \cdot \tau_{y_j}^{(t)} \tag{21}$$

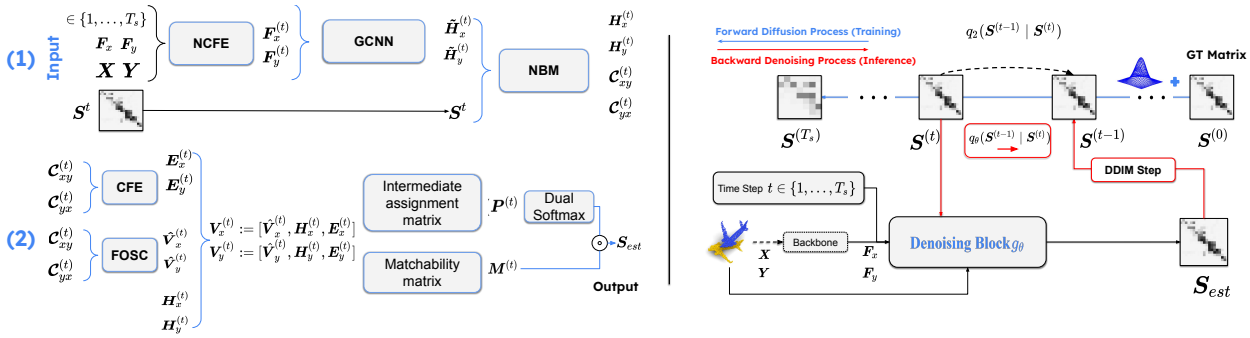

Figure 2: (**Left**) Architecture of the denoiser g_θ for estimating the denoised score matrix $\boldsymbol{S}_{est}$. (**Right**) Overall diffusion process: g_θ takes feature descriptors $\boldsymbol{F}_X$, $\boldsymbol{F}_Y$, spatial coordinates $\boldsymbol{X}$, $\boldsymbol{Y}$, timestep t, and a noisy score matrix $\boldsymbol{S}^{(t)}$ as input, and iteratively recovers the clean matrix over T_s steps using DDIM.

The predicted denoised matrix $\boldsymbol{S}_{est}$ is given by:

$$\boldsymbol{S}_{est} = \boldsymbol{M}^{(t)} \circ \left(\text{Softmax}_{\text{row}}(\boldsymbol{P}^{(t)}) \circ \text{Softmax}_{\text{col}}(\boldsymbol{P}^{(t)})\right) \tag{22}$$

During training, the loss described in Eq. (12) and (13) is applied to $\boldsymbol{S}_{est}$, while during inference the operations described before are repeated for each timestep t in $\{T_s, T_s - 1, \ldots, 1\}$, where the matrix $\boldsymbol{S}^{(t-1)}$ is computed using Eq. (8). Figure 2 illustrates the denoiser pipeline and summarizes the diffusion process, while the forward and backward processes algorithms are described in Algorithms 1 and 2.

Algorithm 1: DiffMatcher: Forward Process

Input: $\boldsymbol{S}^{(0)}$, $\boldsymbol{X}, \boldsymbol{Y}$, g_θ
Output: Diffusion loss
$F^X, F^Y \leftarrow \text{Backbone}(\boldsymbol{X}, \boldsymbol{Y})$
$t \sim \text{Uniform}(\{1, \ldots, T_s\})$
$\epsilon_0 \sim \mathcal{N}(0,1)^{M \times N}$
$\boldsymbol{S}^{(t)} \leftarrow \sqrt{\alpha_t}\, \boldsymbol{S}^{(0)} + \sqrt{1-\alpha_t}\, f_\epsilon(\epsilon_0)$
$\boldsymbol{S}_{est} \leftarrow g_\theta(\boldsymbol{S}^{(t)}, t, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{F}_x, \boldsymbol{F}_y)$
$L_{\text{diff}} \leftarrow \text{Gap_loss}(\boldsymbol{S}_{est}, \boldsymbol{S}^{(0)})$
Return L_{diff}

Algorithm 2: DiffMatcher: Backward Process

Input: $\boldsymbol{X}, \boldsymbol{Y}$, g_θ, η
Output: $\boldsymbol{S}_{est}$
$F^X, F^Y \leftarrow \text{Backbone}(\boldsymbol{X}, \boldsymbol{Y})$
$\boldsymbol{S}^{(t)} \sim \mathcal{N}(0,1)^{M \times N}$
for $t = T_s, \ldots, 1$ **do**
  $\boldsymbol{S}_{est} \leftarrow g_\theta(\boldsymbol{S}^{(t)}, t, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{F}_x, \boldsymbol{F}_y)$
  $\epsilon_\theta \leftarrow \dfrac{\boldsymbol{S}^{(t)} - \sqrt{\bar{\alpha}_t}\, \boldsymbol{S}_{est}}{\sqrt{1-\bar{\alpha}_t}}$
  $\sigma_t \leftarrow \eta \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \sqrt{1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}$
  $c \leftarrow \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}; \quad z_t \sim \mathcal{N}(0,1)^{M \times N}$
  $\boldsymbol{S}^{(t-1)} \leftarrow \sqrt{\alpha_{t-1}}\, \boldsymbol{S}_{est} + c\, \epsilon_\theta + \sigma_t z_t$
  $\boldsymbol{S}^{(t)} \leftarrow \boldsymbol{S}^{(t-1)}$
end
Return $\boldsymbol{S}_{est}$

4 EXPERIMENTAL VALIDATION

In this section, we aim to assess the effects of the proposed diffusion model framework. We achieve this by conducting experiments using object-centric registration datasets, specifically TUD-L (Hodan et al., 2018) and ModelNet40 (Wu et al., 2015), as well as scene-level registration benchmarks such as 7Scenes (Shotton et al., 2013) and the augmented ICL-NUIM (Handa et al., 2014). For the feature extraction block described in Sec. 3.4, we adopt DCP (Wang & Solomon, 2019a) due its efficiency relative to newer, more complex models. The diffusion process can enhance a backbone's performance when faced with noisy inputs, as demonstrated in results from (Li et al., 2022; Zhang et al., 2022; Slimani et al., 2024), and through observations on real-world, low-overlap point clouds noted in (Jiang et al., 2024). The results and experimental setups are detailed in the following sections. The values reported in the tables correspond to DiffMatcher initialized with pairwise point-feature similarity **(Ours)** and DiffMatcher initialized using iMatcher **(Ours-v2)**.

4.1 Implementation Details

The method is implemented in `PyTorch` and trained on an NVIDIA RTX A5000 GPU. On the TUD-L dataset, the model is trained with a batch size of 1 for 15 epochs when combined with DCP, and for 10 epochs when using RPMNet for feature extraction. For ModelNet40, the model is trained for 100 epochs with a batch size of 4, while for ICL-NUIM and 7-Scenes it is trained for 200 epochs with the same batch size. The model is trained for 100 epochs on ModelNet40 and 200 epochs on ICL-NUIM and 7-Scenes, with a batch size of 4. Across all experiments, we adopt the Adam optimizer with a fixed learning rate of 10^{-4}, set the feature dimension to $d = 96$, and use $k = 16$ neighbors in the graph construction. The gap loss described in Eq. (12) and (13) is applied to minimize the error with respect to the ground-truth correspondences. The total training loss is defined as the mean of the diffusion loss and the feature learning loss estimated using the score matrix output by *iMatcher*. During inference, the initial noisy correspondence matrix $\boldsymbol{S}^{(T_s)}$ is initialized in three different ways: with Gaussian noise sampled from a normal distribution, with a dual-softmax computed from pairwise similarities between the point features $\boldsymbol{F}_x$ and $\boldsymbol{F}_y$, or with a single iteration of *iMatcher* applied to $\boldsymbol{F}_x$ and $\boldsymbol{F}_y$.

4.2 Object-centric Registration

TUD-L We first replicate the experiment from (Jiang et al., 2024) on TUD-L, which stands as a challenging benchmark for 6D object pose estimation. It consists of captured scans of three household objects in motion under varying ambient illumination and different poses, offering more than 11,000 RGB-D frames per object for training and 600 frames for testing. For this specific dataset, we also adopt RPMNet (Yew & Lee, 2020) as a feature extractor to maintain consistency with (Jiang et al., 2024). The results for the mean average precision (mAP) at two different rotation and translation thresholds, reported in Tab. 1, show that DiffMatcher provides significant improvements for both translation and rotation with both feature extractors (DCP and RPMNet). Using DCP, our method achieves the best score and improves the rotation metrics by between 29% and 55%, and the translation evaluation by up to 86%. It surpasses other DCP-based methods, such as Diff-DCP (Jiang et al., 2024) and MN-DCP (Dang et al., 2022), by a significant margin across the different metrics, *i.e.*, between 8% and 19% for MN-DCP and between 6% and 17% for Diff-DCP, except for mAP(2 cm), where the scores are equal. When combined with RPMNet, DiffMatcher achieves improvements of 10% to 22% on rotation mAP and 3% to 7% on translation mAP. However, it is slightly outperformed by Diff-DCP, with a margin of approximately 1% to 2%.

ModelNet40 ModelNet40 (Wu et al., 2015) is a synthetic dataset containing 12,311 CAD models of real-world objects, split into 9,843 training and 2,468 test point clouds. Each pair is generated by applying a random rigid transformation, with rotations between 0° and 45° on each axis and translations up to 50cm. To simulate partial occlusions, we sample the 768 nearest neighbours of a random point in both source and target clouds, following (Wang & Solomon, 2019b; Li et al., 2022; Zhang et al., 2022; Slimani et al., 2024). Gaussian noise ($\mu = 0\ m$, variance $= 0.01\ m$) is added and clipped to $[-0.05\ m, 0.05\ m]$ per coordinate. We use the RMSE and R^2 metrics from WsDesc (Li et al., 2022). As shown in Tab. 2, DiffMatcher outperforms

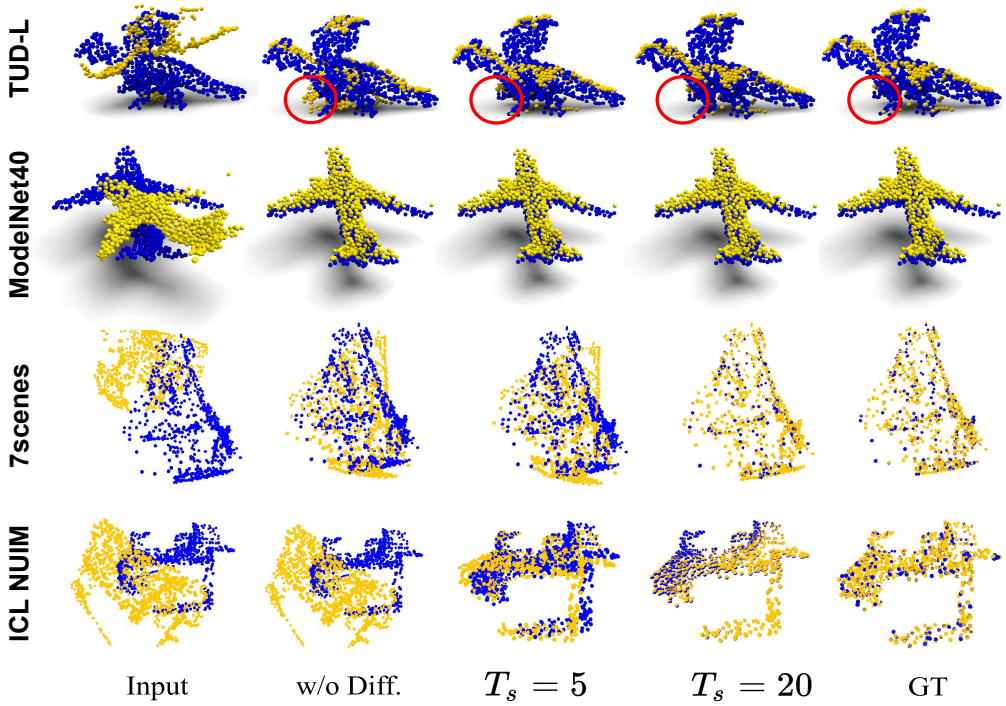

Figure 3: Qualitative registration results on TUD-L, ModelNet40 7scenes and ICL-NUIM for the baseline and for DiffMatcher with varying T_s

Table 1: Performance on the TUD-L dataset. mAP are computed on Relative Rotation Error (RRE) and Relative Translation Error (RTE). Best results in **bold**, second best underlined.

Method	mean Average Precision (mAP) ↑			
	5°	10°	1cm	2cm
FGR (Zhou et al., 2016)	0.00	0.01	0.04	0.25
S4PCS (Mellado et al., 2014)	0.30	0.50	0.05	0.40
IDAM (Li et al., 2020)	0.03	0.05	0.02	0.25
FMR (Huang et al., 2020)	0.02	0.09	0.02	0.23
RIENet (Shen et al., 2022)	0.00	0.00	0.06	0.13
MN-IDAM (Dang et al., 2022)	0.36	0.46	0.23	0.47
RPMNet (Yew & Lee, 2020)	0.73	0.97	0.89	0.94
Diff-RPMNet (Jiang et al., 2024)	**0.90**	**0.98**	**0.98**	**0.99**
Ours ($T^s = 10$)	0.88	0.97	0.96	0.97
Ours ($T^s = 20$)	0.89	0.97	0.95	0.97
Ours-v2 ($T^s = 10$)	0.89	0.96	0.95	0.97
Ours-v2 ($T^s = 20$)	0.89	0.96	0.97	0.97
Improvement (%)	*22*	*10*	*7*	*3*
DCP (Wang & Solomon, 2019a)	0.23	0.62	0.04	0.26
Diff-DCP (Jiang et al., 2024)	0.65	0.85	0.73	**0.94**
MN-DCP (Dang et al., 2022)	0.70	0.81	0.71	0.86
Ours ($T^s = 10$)	0.70	0.84	0.81	0.88
Ours ($T^s = 20$)	0.73	0.86	0.85	0.90
Ours-v2 ($T^s = 10$)	0.75	**0.91**	0.86	0.93
Ours-v2 ($T^s = 20$)	**0.78**	**0.91**	**0.90**	**0.94**
Improvement (%)	*55*	*29*	*86*	*68*

Table 2: Performance on noisy point clouds from ModelNet40. Best results in **bold**, second best underlined.

Method	RMSE(**R**)↓	R^2(**R**)↑	RMSE(**t**)↓	R^2(**t**)↑
PRNet (Wang & Solomon, 2019b)	4.323	0.889	0.017	0.995
RPM-Net (Yew & Lee, 2020)	1.870	0.979	0.011	0.998
OMNet (Xu et al., 2021)	4.504	0.880	0.021	0.995
FGR (Zhou et al., 2020)	27.653	-3.491	0.070	0.941
WSDesc (Li et al., 2022)	3.500	0.928	0.006	0.999
VRNet (Zhang et al., 2022)	3.615	-	0.010	-
RoCNet (Slimani et al., 2024)	1.920	-	**0.003**	-
DCP (Wang & Solomon, 2019a)	6.883	0.718	0.028	0.991
Ours (T^s = 10)	1.813	0.980	**0.003**	**1**
Ours (T^s = 20)	1.745	0.982	**0.003**	**1**
Ours-v2 (T^s = 10)	1.686	0.983	**0.003**	**1**
Ours-v2 (T^s = 20)	**1.621**	**0.984**	**0.003**	**1**
Relative Improv.(%)	*70.4*	*35.8*	*89.3*	*0.91*

Table 3: Comparison of RMSE and MAE on 7Scenes and ICL-NUIM. Best results in **bold**, second best underlined.

Method	7Scenes				ICL-NUIM			
	RMSE(R)↓	RMSE(t)↓	MAE(R)↓	MAE(t)↓	RMSE(R)↓	RMSE(t)↓	MAE(R)↓	MAE(t)↓
S4PCS (Mellado et al., 2014)	19.2603	0.3646	15.7001	0.2898	28.8616	0.3091	24.1373	0.2502
SDRSAC (Le et al., 2019)	0.3501	0.4997	0.2925	0.4997	9.4074	0.2477	7.8627	0.2076
FGR (Zhou et al., 2016)	0.2724	0.0011	0.1380	0.0006	3.0423	0.1275	1.9571	0.0659
IDAM (Li et al., 2020)	10.5306	0.0539	5.6727	0.0303	9.4539	0.3040	4.4153	0.1385
FMR (Huang et al., 2020)	8.6999	0.0199	3.6569	0.0101	1.8282	0.0685	1.1085	0.0398
CEMNet (Jiang et al., 2021)	0.1768	0.0012	0.0434	0.0002	**0.0821**	**0.0002**	**0.0211**	**0.0001**
DCP (Wang & Solomon, 2019a)	7.5548	0.0411	5.6991	0.0303	9.2142	0.0191	6.5826	0.0134
Ours ($T^s = 10$)	0.0496	0.0004	0.0148	**0.0000**	1.1108	0.0031	0.1698	0.0005
Ours ($T^s = 20$)	**0.0292**	**0.0001**	**0.0083**	**0.0000**	0.4259	0.0020	0.0537	0.0002
Ours-v2 ($T^s = 10$)	0.0453	0.0004	0.0142	0.0001	7.3844	0.0038	0.3919	0.0005
Ours-v2 ($T^s = 20$)	**0.0292**	**0.0001**	**0.0083**	**0.0000**	7.2591	0.0043	0.3643	0.0005
Relative Improv.(%)	*99.6*	*99.8*	*99.9*	*100.0*	*95.4*	*89.5*	*99.2*	*98.5*

all baselines, with significant RMSE improvements in rotation (70.4%) and translation (89.3%). Overall, it achieves the best results, only matched by RoCNet (Slimani et al., 2024) in translation RMSE.

4.3 Scene-wise Registration

In this section, we evaluate DiffMatcher for scene-level registration, following the protocol of (Jiang et al., 2021) on two indoor datasets: 7-Scenes (real RGB-D scans, 296 training / 57 test samples) and ICL-NUIM (synthetic scans, 1,278 training / 200 test samples). We apply random rotations ($[0, \pi/4]$) and translations ($[-0.5, 0.5]$ m) for augmentation, and use farthest point sampling to simulate partial views (768 points). As shown in Table 3, DiffMatcher significantly outperforms baselines on both datasets, with RMSE and MAE improvements of over 90% versus DCP. On 7-Scenes, it is the most accurate method, nearly 10 times more precise than CEMNet. On ICL-NUIM, it ranks second, just behind CEMNet. Qualitative registration examples for $T_s = 5$ and $T_s = 10$, compared with iMatcher without diffusion refinement, are shown in Fig. 3.

4.4 Parametric Study and Analysis

To expand on previous research and minimize the effects of $\boldsymbol{S}^{(T_s)}$ initialization and the number of inference iterations, we conduct a new comparative analysis. We examine three different initializations: Gaussian noise, dual softmax feature, and a single *iMatcher* iteration. Then, we compare performance across various numbers of denoising iterations (2, 5, 10, and 20). The evaluation uses precision (P), accuracy (A), and recall (R) on noisy point clouds from ModelNet40, as well as the translation and rotation mean average precision (mAP) on the TUD-L dataset. For the ModelNet40 results shown in Fig. 4, performance consistently improves as the number of iterations increases across all initialization methods. After just two iterations,

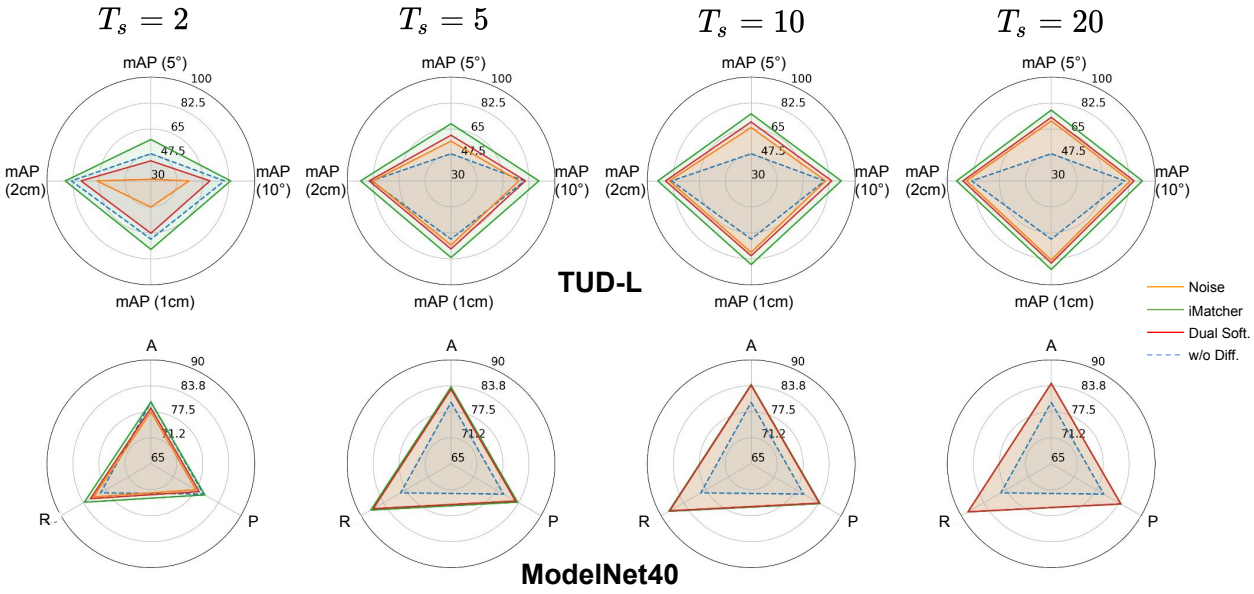

Figure 4: DiffMatcher performance across TUD-L and ModelNet40, under different initialization setups. Larger filled regions indicate better performances.

the scores reach roughly 80%, then increase by 5-6% as the process continues. Additionally, the figure shows that the *iMatcher* initialization produces better scores with a low number of iterations (e.g., two). In contrast, the noise and dual softmax methods require more iterations to achieve this performance before all three approaches converge at a similar plateau. In the TUD-L dataset, the choice of initialization has a more significant impact. The *iMatcher* approach improves rapidly, achieving significantly higher scores and outperforming the noise and dual softmax variants by a substantial margin after just a few iterations. In contrast, noise initialization initially struggled, only gradually achieving competitive performance with more iterations. Although it eventually converges with the dual softmax method, both remain significantly behind DiffMatcher initialized with the *iMatcher* baseline. These findings highlight the significance of proper initialization, particularly for challenging datasets like TUD-L, as it allows for strong performance with fewer iterations. On an A5000 GPU, the inference time varies from 94.6 ms for two iterations to 441.6 ms for twenty iterations, which is still acceptable for many robotics applications that require near real-time performance. Finally, it is worth noting that these experiments highlight the specific contribution of DiffMatcher. When diffusion is removed and only the *iMatcher* matching block is used, performance drops markedly. DiffMatcher consistently achieves superior performance from as early as five iterations on both datasets. The advantage is particularly evident on the challenging real-world TUD-L benchmark, where the rotation mAP at 5° improves by 9% with noise initialization, 13% with dual-softmax initialization, and 20% with *iMatcher* initialization. After 20 iterations, these gains increase further to 22%, 25%, and 29%, respectively. These results highlight the essential role of the diffusion component in supporting reliable and precise matching, regardless of the initialization approach.

5 CONCLUSION

In this paper, we introduce DiffMatcher, a new diffusion-based framework for predicting soft assignment scores in 3D point cloud registration. DiffMatcher uses a denoising network trained to estimate noise in point cloud features at each diffusion step, enhanced by noise-conditioned feature encoding. This design allows the subsequent modules, the Noisy Bilateral Matching and the Correspondence Fourier Embedding, to gradually reduce noise during the reverse process, refining the score matrix while maintaining the local coherence and global spatial consistency of the predicted correspondences. Extensive results on real-world and synthetic

datasets show that DiffMatcher significantly improves the matching process and pose estimation. Several experiments on 6DoF object pose estimation and scene registration have demonstrated that DiffMatcher can surpass baseline methods that struggle with noisy or low-overlap data, reaching the level of state-of-the-art approaches or even exceeding them. At the same time, it maintains reasonable efficiency and achieves relative performance improvements that in some cases approach 100%.

In our future work, we intend to enhance DiffMatcher's generalization and robustness by pre-training it on more extensive object-focused datasets. Additionally, we plan to incorporate DiffMatcher into closed-loop robotic servo control, employing reverse diffusion dynamics as the direct basis for controller updates. This integration would shift DiffMatcher from merely offline registration to an online control system, allowing robots to continually align with targets in real-time, despite obstacles such as partial visibility, noise, and clutter.

References

Maxime Adjigble, Brahim Tamadazte, Cristiana De Farias, Rustam Stolkin, and Naresh Marturi. 3d spectral domain registration-based visual servoing. In *IEEE ICRA*, pp. 769–775, 2023.

F.-A. Croitoru, V. Hondru, et al. Diffusion models in vision: A survey. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 45(9):10850–10869, 2023.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Adv. Neural. Inf. Process. Syst.*, 26, 2013.

Z. Dang, L. Wang, et al. Learning-based point cloud registration for 6d object pose estimation in the real world. In *ECCV*, pp. 19–37, 2022.

A. Handa, T. Whelan, et al. A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *IEEE ICRA*, pp. 1524–1531, 2014.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Adv. Neural. Inf. Process. Syst.*, 33: 6840–6851, 2020.

T. Hodan, F. Michel, et al. Bop: Benchmark for 6d object pose estimation. In *ECCV*, pp. 19–34, 2018.

X. Huang, G. Mei, and J. Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *IEEE/CVF CVPR*, pp. 11366–11374, 2020.

H. Jiang, Y. Shen, et al. Sampling network guided cross-entropy method for unsupervised point cloud registration. In *IEEE/CVF ICCV*, pp. 6128–6137, 2021.

H. Jiang, M. Salzmann, et al. Se (3) diffusion model-based point cloud registration for robust 6d object pose estimation. *Adv. Neural. Inf. Process. Syst.*, 36, 2024.

H. M Le, T.-T. Do, et al. Sdrsac: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *IEEE/CVF CVPR*, pp. 124–133, 2019.

J. Li, C. Zhang, et al. Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration. In *ECCV 2020*, pp. 378–394, 2020.

L. Li, H. Fu, and M. Ovsjanikov. Wsdesc: Weakly supervised 3d local descriptor learning for point cloud registration. *IEEE Trans. Vis. Comput. Graph.*, 2022.

Y. Liu, D. Yao, et al. Fusion of multimodality image and point cloud for spatial surface registration for knee arthroplasty. *Int. J. Med. Robot. and Ass. Surg.*, 18:e2426, 2022.

N. Mellado, D. Aiger, et al. Super 4pcs fast global pointcloud registration via smart indexing. In *Comp. Graph. Forum*, volume 33, pp. 205–215, 2014.

B. Mildenhall, P. Srinivasan, et al. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *Int. Conf. on Mach. Learn.*, pp. 8162–8171, 2021.

Z. Qin, H. Yu, et al. Deep graph-based spatial consistency for robust non-rigid point cloud registration. In *IEEE/CVF CVPR*, pp. 5394–5403, 2023.

Y. Shen, L. Hui, et al. Reliable inlier evaluation for unsupervised point cloud registration. In *AAAI Conf. on Art. Intell.*, volume 36, pp. 2198–2206, 2022.

C. Shi, X. Chen, K. Huang, et al. Keypoint matching for point cloud registration using multiplex dynamic graph attention networks. *IEEE Rob. and Auto. Let.*, 6:8221–8228, 2021.

J. Shotton, B. Glocker, et al. Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE/CVF CVPR*, pp. 2930–2937, 2013.

K. Slimani, B. Tamadazte, and C. Achard. Rocnet: 3d robust registration of points clouds using deep learning. *Mach. Vis. and Appl.*, 35(4):100, 2024.

K. Slimani, C. Achard, and B. Tamadazte. imatcher: Improve matching in point cloud registration via local-to-global geometric consistency learning. *arXiv preprint arXiv:2509.08982*, 2025.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Adv. Neural. Inf. Process. Syst.*, 32, 2019.

A. Vaswani, N. Shazeer, et al. Attention is all you need. *Adv. Neural. Inf. Process. Syst.*, 30, 2017.

Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE/CVF ICCV*, 2019a.

Y. Wang and J. M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. *Adv. Neural. Inf. Process. Syst.*, 32, 2019b.

Y. Wang, Y. Sun, et al. Dynamic graph cnn for learning on point clouds. *ACM Trans. on Grap.*, 2019.

Q. Wu, H. Jiang, et al. Diff-reg: Diffusion model in doubly stochastic matrix space for registration problem. In *ECCV*, pp. 160–178, 2024.

Y. Wu, Y. Yuan, et al. Pcrdiffusion: Diffusion probabilistic models for point cloud registration. *arXiv preprint arXiv:2312.06063*, 2023.

Z. Wu, S. Song, A. Khosla, et al. 3d shapenets: A deep representation for volumetric shapes. In *IEEE/CVF CVPR*, pp. 1912–1920, 2015.

H. Xu, S. Liu, G. Wang, et al. Omnet: Learning overlapping mask for partial-to-partial point cloud registration. In *IEEE/CVF ICCV*, pp. 3132–3141, 2021.

Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *IEEE/CVF CVPR*, pp. 11824–11833, 2020.

Z. Zhang, J. Sun, et al. Vrnet: Learning the rectified virtual corresponding points for 3d point cloud registration. *IEEE Trans. on Cir. and Sys. for Video Tech.*, 32:4997–5010, 2022.

L. Zhou, S. Wang, et al. A fast and accurate solution for pose estimation from 3d correspondences. In *IEEE ICRA*, pp. 1308–1314, 2020.

QY Zhou, J Park, et al. Fast global registration. In *ECCV*, volume 9906, pp. 694–711, 2016.

H. Zhu, Y. Wang, et al. Point cloud matters: Rethinking the impact of different observation spaces on robot learning. *Adv. Neural. Inf. Process. Syst.*, 37:77799–77830, 2024.