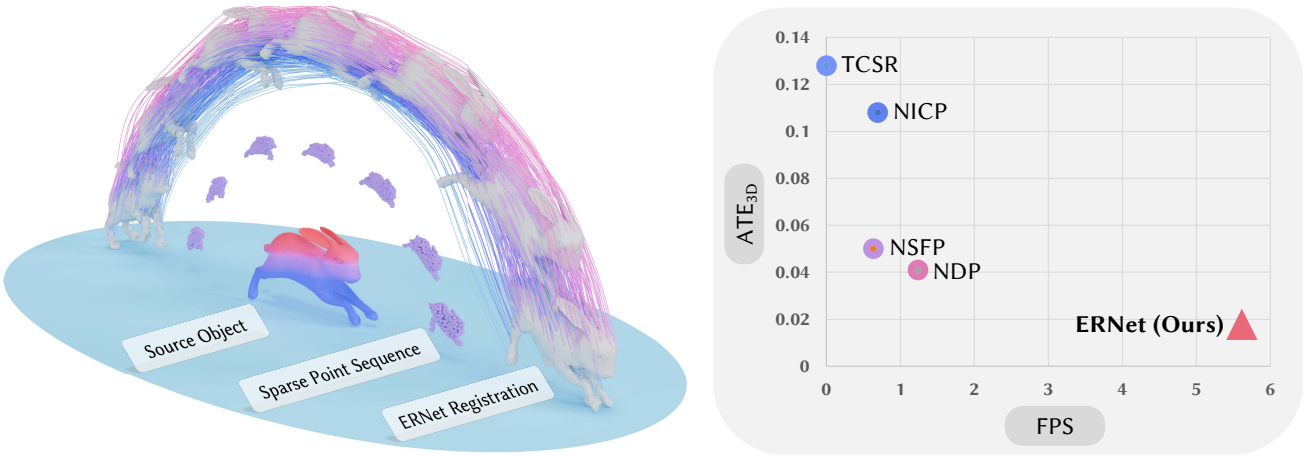
ERNet: Efficient Non-Rigid Registration Network for Point Sequences

Guangzhao He Yuxi Xiao Zhen Xu Xiaowei Zhou Sida Peng†

Zhejiang University



(a) ERNet Registered Trajectories **(b) Comparison Results on D-FAUST**

Figure 1. (a) Given a source object (center bunny) and a sequence of sparse or partial point clouds (purple bunnies), ERNet efficiently and accurately predicts feed-forward registrations. The gradient texture on the center bunny corresponds to the colors of its vertex-wise trajectories shown above. (b) ERNet achieves the lowest Average Trajectory Error in 3D (ATE_{3D}) among all baselines on the D-FAUST dataset while delivering over 4x speedup in FPS.

Abstract

Registering an object shape to a sequence of point clouds undergoing non-rigid deformation is a long-standing challenge. The key difficulties stem from two factors: (i) the presence of local minima due to the non-convexity of registration objectives, especially under noisy or partial inputs, which hinders accurate and robust deformation estimation, and (ii) error accumulation over long sequences, leading to tracking failures. To address these challenges, we introduce to adopt a scalable data-driven approach and propose ERNet, an efficient feed-forward model trained on large deformation datasets. It is designed to handle noisy and partial inputs while effectively leveraging temporal information for accurate and consistent sequential registration. The key to our design is predicting a sequence of deformation graphs through a two-stage pipeline, which first estimates frame-wise coarse graph nodes for robust initialization, before refining their trajectories over time in a sliding-window fashion. Extensive experiments show that our proposed approach (i) outperforms previous state of the art on both the DeformingThings4D and D-FAUST datasets, and (ii) achieves more than 4x speedup compared to the previous best, offering significant efficiency improvement.

The authors are affiliated with the State Key Lab of CAD&CG. †Corresponding author.

1. Introduction

As Carl Jung once said, "In all chaos there is a cosmos, in all disorder a secret order." Predicting structured motion representation, more specifically 3D trajectories, within unordered point sequences has always been a fundamental challenge in computer vision and robotics, with wide applications among dynamic reconstruction, scene understanding, robot manipulation and more. In this work, we target on the problem of sequential non-rigid registration, which aims to register a source mesh of an object onto a sequence of observed point clouds, often sparse or partial, resulting in dense trajectories of the object over time. Traditional methods [4, 5] generally formulate this task as an optimization problem that solves point-wise transformations by minimizing the distance between source and target point clouds. While many follow-up works [1, 7, 15, 33, 34] have introduced regularizers, such as

As-Rigid-As-Possible constraint [26, 33] and deformation graph [1, 37], to improve the robustness of their optimization process, they still easily get stuck in local optima due to the non-convexity of the objective functions.

With the success of deep learning and neural networks, researchers have been exploring the potentials of neural networks in solving this problem. One type of this research leverages the functionality of neural network in representing deformations [3, 6, 14, 27, 36] to register point clouds by optimizing neural deformation fields, partially alleviating the local optimal problem. However, because of their optimization-based nature, they still struggle to generalize to noisy or partial inputs. To address this, another line of work [17, 18, 30, 40] propose to train registration prior models, leveraging learned deformation knowledge to improve robustness and mitigate local minima. While effective, these methods perform frame-wise registration rather than directly operating on sequential data, leading to time-consuming processing and issues such as error accumulation and temporal inconsistency.

In this paper, we propose a novel framework, named ER-Net, for efficient sequential non-rigid registration. Our core innovation lies in a feed-forward network that predicts a sequence of deformation graphs based on spatio-temporal matching across point clouds. Specifically, instead of implicitly predicting a set of 3D shape keypoints as the graph nodes [12], we explicitly obtain a set of nodes from the source object using the farthest point sampling algorithm, considering its applicability for diverse 3D shapes. Then, our approach leverages a coarse-to-fine strategy to regress frame-wise node positions throughout the point cloud sequence, which first estimates coarse node positions in each frame through spatial matching between source nodes and point clouds, and then globally refines the node trajectories across frames in a sliding-window manner. As shown in our experiments, the use of deformation graphs enables efficient feed-forward registration, while the two-stage strategy, taking into account the spatio-temporal relationship between node positions, enhances robustness and consistency under noisy and partial input.

A remaining problem is how to obtain the blending weights and SE(3) transformations of the graph nodes to drive the source object onto each frame. We find that naively predicting these properties with a neural network suffers from their non-linearity and high-dimensional characteristics, resulting in sub-optimal performance. To overcome this issue, our approach exploits the local rigidity of non-rigidly deforming objects to infer these properties. During the refinement stage, we jointly predict the radii of nodes, which can be easily inferred based on local geometric cues and motion correlations. Then, using the radial basis function, we define blending weights for each pair of source point and deformation node. For the SE(3) transformation, we follow Procrustes analysis and group a set of local nodes to calculate their rotations and translations through the singular value decomposition algorithm. Our experimental results demonstrate that this strategy works well and produces high-quality registration results.

We evaluate our approach on the DeformingThings4D [20] and D-FAUST [2] datasets, which are challenging benchmarks for estimating sequential non-rigid deformation. Across these datasets, our method achieves state-of-the-art performance in both accuracy and efficiency. Additionally, we demonstrate its robustness to sparse and partial inputs, and conduct ablation studies to validate the effectiveness of our proposed modules.

In summary, our contributions are:

- We propose an architecture for feed-forward sequential non-rigid registration, incorporating a novel two-stage prediction strategy for improved robustness and temporal consistency.
- We propose to directly regress deformation graphs as an efficient representation for non-rigid registration.
- We evaluate the proposed pipeline on several different deformation datasets, and demonstrate significant improvements in both accuracy and speed compared to the state-of-the-art.

2. Related Work

Representations of Deformation Field. Representations of deformation field often involve the trade-off between its expressiveness and the computational cost. Point-wise affine transformation is one of the simplest ways to define the deformation field [1, 10, 22]. Although it is highly expressive for modeling complex motion, its redundancy in degrees of freedom often leads to expensive computational cost and under-constrained problem. To mitigate this, deformation graphs [35] are proposed to represent point-wise motion with a set of graph nodes, each associated with an SE(3) transformation. Individual deformation for each point can then be calculated with weighted skinning. Since the number of graph nodes is typically several orders of magnitude smaller than the original point cloud, they can offer significant efficiency improvement while reducing the solution space for deformation optimization. Recently, with the success of implicit neural fields [24, 25], some works [14, 21, 28] propose to represent the deformation field as a continuous implicit mapping from 3D coordinates to deformation vectors, often implemented as a multilayer perceptron (MLP). Furthermore, to reduce the high complexity of modeling deformations with neural networks, [19] proposes to use several levels of MLPs to hierarchically represent deformations with different levels of details. However, such implicit representations are often inefficient, requiring per-frame optimization which is impractical for registering long sequences. In contrast, we adopt deformation graphs as an explicit representation of

deformation, and propose to regress them with a neural network in a feed-forward manner, striking a balance between efficiency and expressiveness.

Non-rigid Registration. Non-rigid registration aims to find point-wise deformation from the source to the target point cloud. Registration algorithms are typically designed to accommodate specific deformation representations. Non-rigid iterative closest point (NICP) [16] is a classic optimization-based registration algorithm, and is commonly used for solving either point-wise transformations or deformation graphs. In order to regularize the optimization process and better preserve local topology details, As-Rigid-As-Possible [11] constraint is proposed to regularize the neighboring nodes to deform as rigidly as possible. However, they are still sensitive to initialization and often get stuck in local minima. To further regularize the deformation and incorporate temporal information, OccupancyFlow [27] first proposes to model sequential deformation as a neural velocity field and predicts deformation with an ordinary differentiable equation (ODE) solver. Similarly, CaDeX [14] employs implicit neural network by formulating the deformation field as bijective mappings between each frame and a shared canonical space. Nevertheless, both methods struggle to capture high-frequency deformations effectively with implicit networks. Another line of work [23, 31, 38] predicts dense point correspondences, and estimates frame-wise deformations in a feed-forward manner. While these methods achieve impressive accuracy, extending them for efficient sequential registration is non-trivial due to their limited ability to aggregate temporal information. Moreover, their computational complexity scale poorly with the number of input points, making them inefficient for large-scale inputs. In this paper, we address these limitations by making efficient feed-forward prediction of sparse deformation graphs, and incorporating a coarse-to-fine strategy to fully utilize temporal information for robust and accurate registration.

Temporal Tracking. Recent advancements in Tracking Any Points (TAP) algorithms [8, 9, 13, 39] have established an effective framework for tracking arbitrary 2D points over long video sequences. More specifically, CoTracker [13] introduces a sliding-window approach with overlapping segments, ensuring temporal consistency through a spatio-temporal transformer. SpatialTracker [39] extends this concept by lifting 2D image features using off-the-shelf depth estimators and perform tracking in the 3D space. However, these methods struggle to re-track occluded points over extended sequences due to limited window size. To address this limitation, we propose a two-stage registration pipeline. In the first stage, we employ a dedicated frame-wise matching module for coarse yet robust initialization, effectively handling occlusions and preventing error accumulation in long sequences. In the second stage, we incorporate a 3D temporal refinement module for sequential non-rigid registration, which predicts temporally consistent and accurate trajectories for graph nodes while also estimating node radii for weighted skinning.

3. Method

Given a dense source point cloud $\mathbf{X}_s \in \mathbb{R}^{N_s \times 3}$ and a temporal sequence of sparse target points $\mathcal{P} = \{\mathbf{P}_i \in \mathbb{R}^{N_i \times 3} \,|\, i = 1, \cdots, T\}$, our goal is to design and train a feed-forward model to predict a series of non-rigid deformation fields:

$$\mathcal{W}_i : (\mathbf{X}_s, \mathcal{P}) \mapsto \mathbf{X}_p^i \,,\; i = 1, \cdots, T, \tag{1}$$

which register the source point cloud $\mathbf{X}_s$ to its corresponding point cloud $\mathbf{X}_p^i$ at each frame. In this paper, we focus on developing an learning-based framework that can generalize to diverse object categories and shape deformations while producing high-accuracy and temporally consistent registrations throughout the point cloud sequence.

The overview of our approach is illustrated in Fig. 2. We first encode the source point cloud $\mathbf{X}_s$ and target point cloud sequence $\mathcal{P}$ with an efficient triplane encoder (Sec. 3.1). Then we represent the non-rigid deformations as a sequence of graph nodes. We initialize their per-frame positions with a matching network, and apply a spatio-temporal transformer to iteratively refine their trajectories and radii (Sec. 3.2). Finally, dense deformation fields are applied to $\mathbf{X}_s$ to complete the sequential registration using radial basis blend skinning (Sec. 3.3). To train our model, we introduce a two-stage training strategy that first pre-trains the matching module before optimizing the remaining pipeline end-to-end (Sec. 3.4).

3.1. Efficient point cloud encoding

We encode point clouds using a triplane encoder, which encodes arbitrary number of 3D points into three orthogonal feature planes. With this design, per-point features are compressed continuously into three planes, enabling fast indexing and reducing memory consumption while preserving detailed geometry information. Both per-frame point sequence $\mathcal{P}$ and source point cloud $\mathbf{X}_s$ are encoded using the same encoder with shared weights. The overall point cloud encoder $\mathcal{E}$ is then formalized as:

$$\mathcal{F}_{xy}, \mathcal{F}_{yz}, \mathcal{F}_{xz} = \mathcal{E}(\mathbf{X}) \,,\; \mathbf{X} \in \{\mathbf{X}_s\} \cup \mathcal{P}. \tag{2}$$

Local feature encoder. To extract low-level geometry information, we utilize a modified shallow PointNet [29] with local pooling layers instead of global pooling. More specifically, each residual block is followed by a local pooling operation, which we achieve by projecting per-point features orthographically onto triplane grids and aggregating the averaged results as local features. The aggregated features are

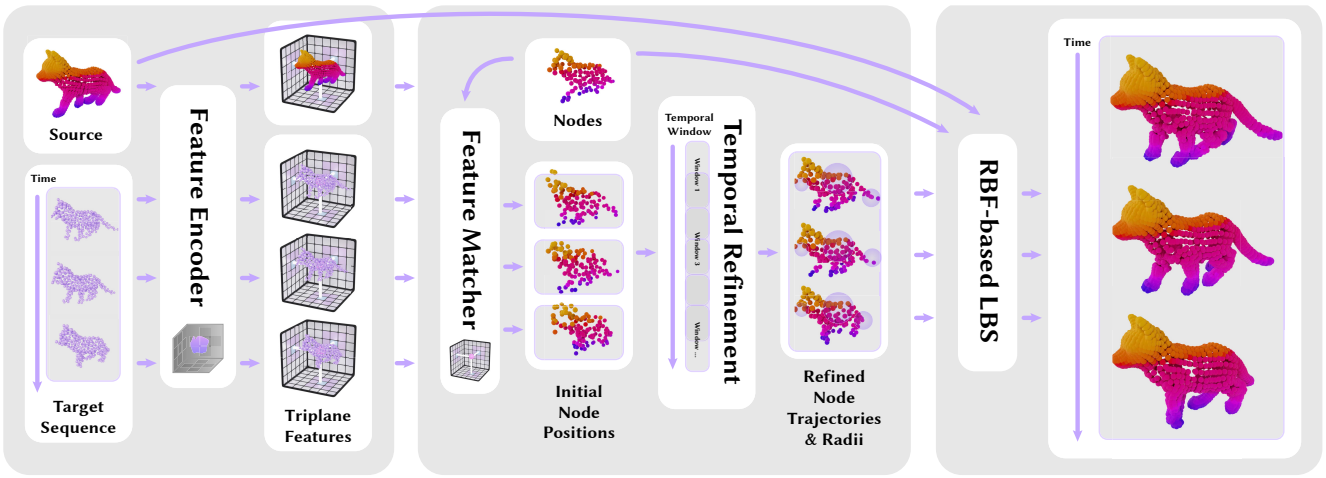


Figure 2. **Overview of our proposed pipeline.** (a) Given a source point cloud and input target point cloud sequence, we first encode them independently using a shared local feature encoder and splat per-point features onto triplane grids (Sec. 3.1). (b) Then we initialize graph nodes based on source point cloud and perform the coarse-to-fine matching to predict the node positions and radii of the deformation graph using encoded features (Sec. 3.2). (c) With the predicted node trajectories and radii, we calculate node transformations via the Procrustes analysis and utilize the RBF-based LBS to produce dense registration (Sec. 3.3).

then concatenated with per-point features to form the inputs for following residual blocks. This ensures that the encoded feature contains semi-global information while maintaining accuracy around local areas, which benefits accurate feature matching.

Triplane feature maps. After embedding local feature for every point, the geometry is still discrete and contains spatial gaps. One naive solution would be to linearly interpolate between nearby points to form a continuous feature field. However, this leads to inaccurate approximation of geometry features and introduces expensive nearest-neighbors searching overhead. Similar to Xiao et al. [39], we project per-point features onto triplane grids to enable fast indexing, and apply a shallow U-Net [32] to fill gaps and complete the geometry features. Compared to using 3D CNN, our approach of triplane factorization significantly reduces memory consumption.

3.2. Deformation graph prediction

Deformation graph representation. Deformation graph is an efficient and global representation for motion [35]. It consists of a set of sparse graph nodes $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T})$, and each graph node is associated with its positions $\mathcal{V} = \{\mathbf{V}_p \in \mathbb{R}^3 \,|\, p = 1, \cdots, B\}$, radii $\mathcal{R} = \{\mathbf{R}_p \in \mathbb{R} \,|\, p = 1, \cdots, B\}$ and series of $\mathbf{SE}(3)$ transformations $\mathcal{T} = \{\mathcal{T}_p^i \,|\, i = 1, \cdots, T, p = 1, \cdots, B\}$ resulting the final deformation. Here, we sample B points from the source point cloud as positions of graph nodes using the farthest point sampling algorithm.

Challenges for deformation graph prediction. For deformation graph estimation, one of the naive solution is to extract the features of graph nodes at each time step, and regress the graph nodes attributes directly. However, we found that primitively regressing graph nodes via a neural network is very challenging, especially the $\mathbf{SE}(3)$ transformations. Meanwhile, the temporal consistency is hard to be ensured for the cases of large motions. To overcome this, we firstly estimate the trajectories of graph nodes following coarse-to-fine fashion. The trajectories are actually the linear offsets which is easy for network prediction. The coarse-to-fine strategy first coarsely match the graph nodes with each frame and refine them with an iterative spatio-temporal transformer, which ensures the temporal and global consistency. After that, we leverage the local rigidity to estimate $\mathbf{SE}(3)$ transformations from the predicted trajectories by Procrutes analysis.

Node-to-frame matching. Since $\mathbf{X}_s$ and $\mathcal{P}$ can be in arbitrary poses with large motions, we find it necessary to first initialize graph nodes positions for each frame by using a node-to-frame matching network ϕ. It takes the source nodes $\mathbf{V}_s$ as well as their embedded triplane features $\mathbf{F}_s$, and produces a coarse registration for all given frames:

$$\mathbf{V}_p^i = \phi(\mathbf{V}_s, \mathbf{F}_s, (\mathcal{F}_{xy}^i, \mathcal{F}_{yz}^i, \mathcal{F}_{xz}^i)). \tag{3}$$

More specifically, ϕ is a spatial transformer module, and its input token in frame i is defined as:

$$\mathbf{G}^i = (\gamma(\mathbf{V}_p^i), \gamma(\mathbf{V}_p^i - \mathbf{V}_s), \mathbf{F}_p^i, \mathbf{F}_s, \mathbf{C}_p^i), \tag{4}$$

where γ denotes sinusoidal positional encoding function, $\mathbf{V}_p^i$ and $\mathbf{F}_p^i$ are node positions and features initialized with $\mathbf{V}_s$ and $\mathbf{F}_s$, and $\mathbf{C}_p^i$ denotes the correlations between node features and triplane features near $\mathbf{V}_p^i$. The transformer consists of multiple self-attention blocks, and outputs residuals for updating node positions and features. We apply the transformer for $\mathbf{M} = 6$ times, and after the m-th iteration the updated results are given by:

$$(\mathbf{V}_p^m, \mathbf{F}_p^m) = (\mathbf{V}_p^{m-1}, \mathbf{F}_p^{m-1}) + \phi(\mathbf{G}^{m-1}). \quad (5)$$

Here the sequence index i is omitted for clarity.

Spatio-temporal refinement. After applying node-to-frame matching, the initialized node positions are still inaccurate and lack temporal consistency. To solve this issue, we utilize a spatio-temporal transformer Φ to refine 3D node trajectories based on triplane geometry features. The T frames are first partitioned into overlapping windows of the same length T_w, where the second half of frame overlaps with the first half in the following window. Temporal node graph updates are carried out one window at a time, so that long sequences can be handled in an online fashion. When updating node positions within a window $W = [i_0, i_0 + T_w)$, the transformer Φ's input token is defined as:

$$\mathbf{G} = \cup_{i=i_0}^{i_0+T_w} \{(\gamma(\mathbf{V}_p^i), \gamma(\mathbf{V}_p^i - \mathbf{V}_p^{i_0}), \mathbf{F}_p^i, \mathbf{F}_s, \mathbf{C}_p^i)\}, \quad (6)$$

where notations are similar to those in Eq. 4 except that $\mathbf{V}_p^i$ is initialized using node-to-frame matching results, and $\mathbf{V}_p^{i_0}$ denotes node positions for the first frame in the window. Similar to Eq. 5, the output from the transformer Φ are used to to update the node positions and features. Here we perform trajectory updates within one window for $M = 6$ iterations, and move on to the next window by initializing its first $\frac{T_w}{2}$ frames with the results of last $\frac{T_w}{2}$ frames in the previous window.

Transformation estimation. After recovering the temporal consistent nodes trajectories, our approach uses the predicted node trajectories to estimate transformations from source object to a particular time step for each graph node. We first find a set of source graph nodes that exhibits the local rigidity, then find the corresponding graph nodes at the target time step, and finally solve the transformations between two sets of nodes via the Procrustes analysis process.

Specifically, for a source graph node $\mathbf{v}_s$, our approach first searches for its K-nearest nodes $\mathcal{N} = \{\mathbf{v}_s^k \,|\, k = 1, \cdots, K\}$ in source graph nodes $\mathbf{V}_s$ as the candidates, and finds the trajectories for all K nodes $\{\mathbf{v}_i^k \,|\, i = 1, \cdots, T,\ k = 1, \cdots, K\}$. We assume that the nearest neighbor node $\mathbf{v}_s^1$ is always correctly assigned, and include other graph nodes if they satisfy the following criteria:

$$\mathcal{N}_f' = \{\mathbf{v}_s^k \mid \max_i \{|\frac{\|\mathbf{v}_i^k - \mathbf{v}_i^0\|_2}{\|\mathbf{v}_s^k - \mathbf{v}_s^0\|_2} - 1|\} < \epsilon,\ k = 2, \cdots, K\}, \quad (7)$$

where ϵ is initialized as 0.2. This is used to only include nodes that stay relatively rigid w.r.t. $\mathbf{v}_s$. The resulting node set is thus denoted as $\mathcal{N}_f = \{\mathbf{v}_s^1\} \cup \mathcal{N}_f'$.

To enhance the robustness of the Procrustes analysis, it is important that the number of nodes within $\mathcal{N}_f$ is not less than 4. Therefore, if the number of nodes obtained is less than 4, we increment ϵ by 0.1 and recalculate Eq. 7 to derive a new set of nodes. Then, our approach regards these nodes as one rigid part, and uses the estimated node trajectories to produce corresponding graph nodes at the target time step. Finally, the transformation from two sets of graph nodes are calculated via the Procrustes analysis process. The detailed calculations are provided in the supplemental material.

Node radius prediction. To convert the deformation graph into a dense deformation field, the linear blend skinning (LBS) algorithm [34] is applied here to construct the warping function. However, direct estimate the blending weights is non trivial due to its high dimensionality and non-linearity. Therefore, we utilize an extra spatio-temporal transformer Φ_R to regress a radius $\mathbf{R}_p$ for each node. Then the estimated radii are used to calculate the blending weights with Radial Base Functions (RBF), which is introduced in Sec. 3.3.

3.3. Skinning-based registration

Selective graph nodes assignment. Skinning is an indispensable step for converting the estimated deformation graph into the dense warping function, which is used to establish correspondences between source and observed point clouds. The key challenges here lie on correct nodes assignment for each source point. The incorrect neighbours selection will lead to the wrong skinned deformation when the topology changes occur.

To increase the robustness to close-to-open topology changes, we propose to assign graph nodes to each point by considering the local rigidity. For each point $\mathbf{x}_s$ in source point cloud $\mathbf{X}_s$, we firstly search for its K'_{init}-nearest nodes in source graph nodes $\mathbf{V}_s$ as the candidates, and obtain corresponding nodes based on the estimated node trajectories. Then, these nodes are filtered based on a process similar to Eq. 7. Here ϵ is set as 0.2. This metric effectively filter those graph nodes with substantial relative position shifts, which always indicates for the close-open topology change.

Radial basis skinning. After knowing the graph node neighbors for each source point, the deformation for each point can be calculated with the LBS algorithm. In order to simplify the blending process and ensure the training stable,

Table 1. **Quantitative results** on the test subsets of DT4D-H and D-FAUST datasets. Note that, our approach is only trained on D-FAUST and DT4D-A datasets and can generalize to DT4D-H dataset. Green and yellow cell colors indicate the best and the second best results, respectively.

Method	DT4D-H				D-FAUST			
	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$	$T_{avg} \downarrow$	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$	$T_{avg} \downarrow$
C-NICP [16]	0.107	0.038	0.167	1.569	0.108	0.022	0.188	1.445
C-NSFP [17]	0.070	0.062	0.418	2.040	0.050	0.056	0.437	1.570
C-NDP [19]	0.058	0.192	0.603	0.832	0.041	0.057	0.522	0.808
ERNet (ours)	0.037	0.238	0.678	0.188	0.016	0.313	0.863	0.178

we select the radial basis function (RBF) to calculate the blended weights for skinning. Specifically, for a point $\mathbf{x}_s$, we calculate the blending weight of its assigned node $\mathbf{v}_s(\mathbf{x}_s)$ with respect to frame i using:

$$\mathbf{w}_i = \exp(-\frac{\|\mathbf{x}_s - \mathbf{v}_s(\mathbf{x}_s)\|_2^2}{2\,\mathbf{r}_i(\mathbf{x}_s)^2}). \tag{8}$$

where $\mathbf{r}_i(\mathbf{x}_s)$ is the node radius estimated in Sec. 3.2. To deform the point $\mathbf{x}_s$ from the source object to the position $\mathbf{x}_i$ at frame i, our approach first finds its assigned graph nodes $\{\mathbf{v}_s^k(\mathbf{x}_s)\}$ and corresponding transformations $\{\mathcal{T}_i^k(\mathbf{v}_s)\}$, and then performs the LBS algorithm:

$$\mathbf{x}_i = \frac{\Sigma_{k=1}^{K'} \mathbf{w}_i^k \mathcal{T}_i^k(\mathbf{v}_s)}{\Sigma_{k=1}^{K} \mathbf{w}_i^k}, \tag{9}$$

where K' is the number of graph nodes assigned to point $\mathbf{x}_s$. This is performed for every point in $\mathbf{X}_s$ across all frames, forming the final sequential registrations $\{\mathbf{X}_p^i \mid i = 1, \cdots, T\}$.

3.4. Training

Our model parameters include triplane geometry encoder $\mathcal{E}$, node-to-frame matching network ϕ and spatio-temporal transformers Φ_R and Φ. To optimize them, we design an efficient and stable two-stage training strategy.

We first start with training $\mathcal{E}$ and ϕ from scratch on point cloud pairs, where $\mathcal{P}$ only contains one target point cloud. The optimization is regularized using node position regression loss:

$$\mathcal{L}_{match} = \Sigma_{m=1}^{M} \alpha^{M-m} \|\hat{\mathbf{V}}_p^m - \mathbf{V}_p\|_1, \tag{10}$$

where $\hat{\mathbf{V}}_p^m$ denotes the predicted node position at m-th iteration, and $\mathbf{V}_p$ denotes the ground-truth position. We set $\alpha = 0.8$ to guide the matching network to gradually update nodes from source position towards target position.

After the encoder $\mathcal{E}$ and matching network ϕ converges, we freeze their weights to go on and optimize transformers Φ and Φ_R. This enables us to train them on larger windows and longer sequences with wider temporal context. We give the same supervision on the predictions of each sliding window. So, to avoid the complicated notations, we introduce the total loss on arbitrary sliding window T_w. The total loss is a combination of registration loss, node regression loss and local rigidity constraint:

$$\mathcal{L}_{\text{total}} = \Sigma_{m=1}^{M} \alpha^{M-m} (\mathcal{L}_{\text{reg}}^m + \lambda_{\text{node}} \mathcal{L}_{\text{node}}^m + \lambda_{\text{rigid}} \mathcal{L}_{\text{rigid}}^m), \tag{11}$$

where λ_{node} and λ_{rigid} are weight coefficients, which in practice is set to 1.0 and 0.1, respectively. The losses are summed for every frame in every iteration across all windows. We present the details of these losses in the supplementary material.

4. Experiments

4.1. Implementation details

Our model is trained from scratch with two 48GB A6000 GPUs. The first stage training takes 200k iterations (7 days) to converge, while the second stage is trained for another 300k iterations. For triplane feature encoding, we utilize five consecutive encoding-splatting blocks and set the plane resolution to be 256×256 with 128 feature channels. For node graph estimation, we set $B = 256$ and find it to be the balance between modeling complex deformations and computational efficiency, which is shown in our ablation study 4.4. In the second training stage, we set the window size T_w to 8 and the total training frames to be 12. Both node-to-frame transformer ϕ and sequential transformer Φ have a layer depth of 12, while the depth of node radius regressor Φ_R is set to 6 for efficiency. When performing blend skinning for source points, we choose $K = 4$ to allow smooth transformation while maintaining accuracy by enforcing spatial locality.

4.2. Datasets and metrics

To showcase the generalizability and accuracy of our design, we train one single model on two challenging deformation datasets: DeformingThings4D [20] and Dynamic FAUST [2].

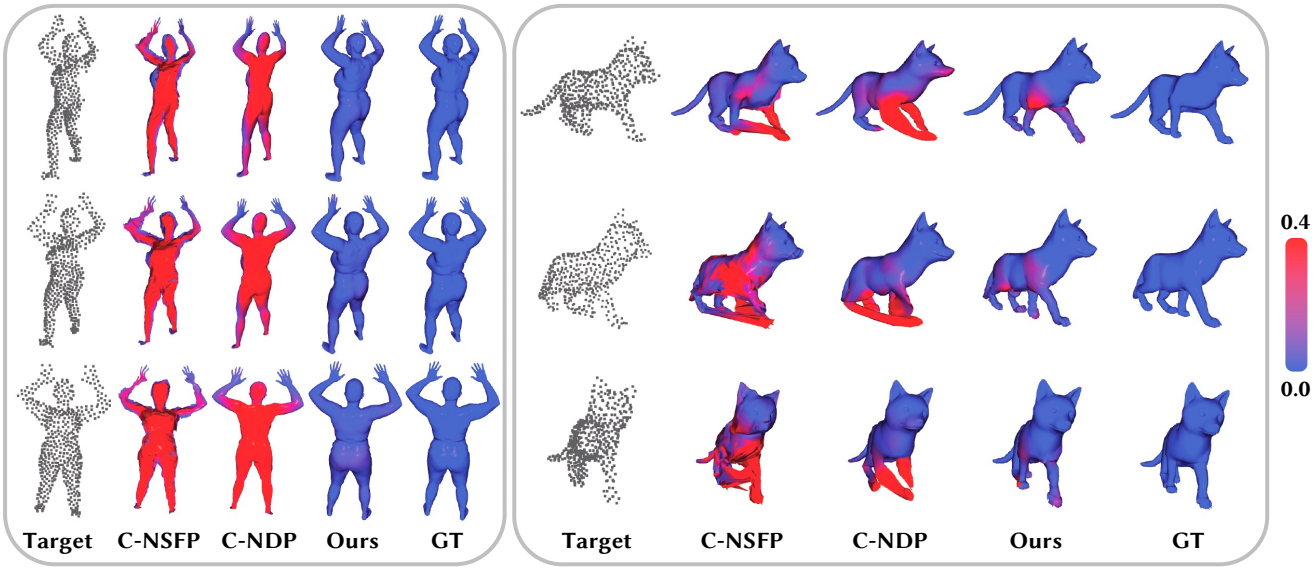


Figure 3. **Qualitative results** on two challenging examples from the depth-sampled D-FAUST and DT4D-A datasets. The point color reflects the L2 distance from ground truth, where blue indicates less error and red indicates more.

DeformingThings4D. It is a synthetic dataset consisted of 1772 animal deformation sequences (DT4D-A) and 200 human deformation sequences (DT4D-H). It is especially challenging thanks to its wide variety of complex shapes and inclusion of highly-deformed sequences. Following CaDeX [14], we filter out the sequences where meshes contain ill-behaved areas and partition a subset of the DT4D-A dataset into training (75%), validation (7.5%) and test (17.5%) subsets. We do not include the DT4D-H dataset for training and sample test subset for evaluation of generalizability.

Dynamic FAUST (D-FAUST). It is a human motion dataset consisting of 10 subjects and 129 deformation sequences. It contains challenging large deformations, e.g., "running on spot" and "punching". To produce partial point clouds, we render depth images through a randomly posed camera per sequence and back-projecting them into 3D space. We follow [27] and partition it into training (70%), validation (10%), and test (20%) subsets.

Metrics. To evaluate sequential registration accuracy, we calculate the average trajectory error ATE_{3D} for each scene, which is the average l1 distance between the predicted registration targets and ground truths. In addition, we compute $\delta_{0.01}$ and $\delta_{0.05}$ to evaluate the stability of registration accuracy, where $\delta_{0.01}$ denotes the fraction of predicted points that are within 0.01 unit length from ground truths, with $\delta_{0.05}$ using a relaxed threshold at 0.05 unit length. To evaluate method efficiency, we provide average registration time per frame T_{avg} in second for every method.

4.3. Sequential non-rigid registration

Since there are no existing works that can be directly used for sequential registration, we construct three baselines using pair-wise non-rigid registration methods, namely NDP [19], NICP [16] and NSFP [17]. To perform registration, we first utilize these methods to predict the deformation from source point cloud to the first frame, and then chaining the registrations by iteratively predicting deformations based on the last predicted registration. We refer to these baselines as chained NDP (C-NDP), chained NICP (C-NICP) and chained NSFP (C-NSFP) respectively. We evaluate our method as well as the constructed baselines using the test sets of DT4D-H and D-FAUST.

The results are shown in Tab. 1. Our method outperforms all baselines by a large margin both in accuracy and efficiency, even though all baselines are optimized per sequence while we utilize one single model across all data. This is especially surprising since we do not utilize the DT4D-H dataset for training. The results demonstrate that our model is highly generalizable and scalable thanks to our trajectory representation of node graphs. We show qualitative results comparing our method with baseline methods in Fig. 3

4.4. Ablation study and analysis

Effectiveness of node graph. We show the effectiveness of our blending skinning approach by constructing a pipeline where we use our network to predict the temporal trajectory across observed point clouds for any point on the source object, which is named “Dense matching”. Since the memory consumption grows exponentially with node number, we iteratively pass all source points through the node graph predictor and aggregate the results. The results on D-FAUST

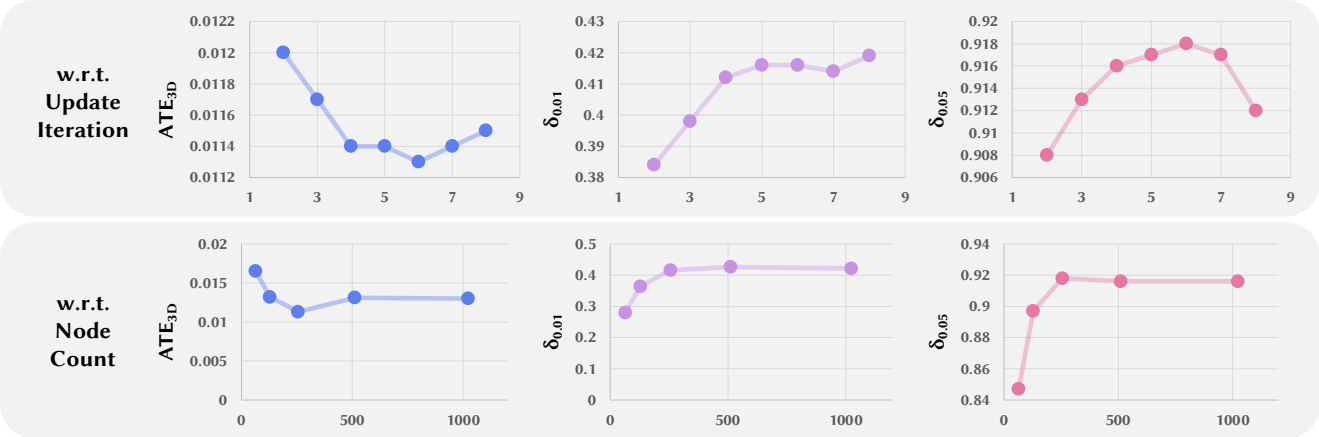


Figure 4. **Analysis on the effect of update iteration and node count.** Metrics ATE_{3D}, $\delta_{0.01}$ and $\delta_{0.05}$ w.r.t. transformer update iterations and node counts are presented. We update trajectories 6 times and set node count $B = 256$ as a balance between performance and efficiency.

Table 2. **Ablation studies.** The results are evaluated on the D-FAUST test set. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported. The average registration time per frame T_{avg} is reported for "Ours" and "Dense matching" for efficiency comparison.

Method	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$	$T_{avg} \downarrow$
Ours	**0.011**	**0.416**	**0.918**	0.172
Dense matching	0.014	0.299	0.885	4.645
Ours *w/o* ϕ	0.022	0.289	0.845	**0.060**
Ours *w/o* Φ	0.014	0.366	0.889	0.169

[2] dataset in Tab. 2 show our skinning approach not only significantly improves registration efficiency but also achieves higher accuracy compared with "Dense matching".

Effectiveness of coarse-to-fine scheme. One of our major insights is that performing sequential non-rigid registration with coarse-to-fine scheme allows higher accuracy and better generalizability. To prove the effectiveness of our design, we construct two variants of our method by removing the node-to-frame matching network ϕ and sequential node regression network Φ respectively. As is shown in Tab. 2, both variants show a significant drop in performance.

Analysis on update iteration. For our baseline method, we perform iterative updates with sequential transformer for $M = 6$ iterations. As comparison, we plot the evaluation results using different update iterations in Fig. 4. The results show that our model trained with $M = 6$ performs best at $M = 6$, but can be inferenced at $M = 4$ for better efficiency with minor performance drop.

Analysis on node amount. Node amount significantly affects registration accuracy, since more nodes are capable of modeling more complex motion with more detail. However, prediction large amount of graph nodes is computationally expensive. We show the evaluation results for different node amount in Fig. 4, which shows our choice of $B = 256$ nodes is the best balance between registration accuracy and efficiency.

5. Conclusions

In this work, we performs efficient sequential non-rigid registration by representing temporal correspondences as deformation graphs. We designed a coarse-to-fine matching pipeline to estimate node trajectories of deformation graphs with strong generalization ability. Based on the predicted node positions and radii, we additionally proposed an RBF-based LBS technique for deforming the source object to target point clouds. Experiments demonstrate that our method outperforms existing methods both in accuracy and efficiency across public datasets.

Limitations. Although our method displays great generalizability, it is trained on limited data due to the lack of annotated 4D correspondences, and could benefit from being trained on a larger variety of objects and motions. Its further applications in other areas, such as dynamic scene editing, compression, autonomous driving and robotics, are still yet to be explored.

Acknowledgments. This work was partially supported by the National Key R&D Program of China (No. 2024YFB2809102), NSFC (No. 624B1017, No. 62402427, No. U24B20154), Zhejiang Provincial Natural Science Foundation of China (No. LR25F020003), Zhejiang University Education Foundation Qizhen Scholar Foundation, and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

[1] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. on Graphics (TOG)*, 22(3): 587–594, 2003. 1, 2

[2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: Registering human bodies in motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 6, 8

[3] Aljaz Bozic, Pablo R. Palafox, Michael Zollhöfer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[4] Benedict J. Brown and Szymon Rusinkiewicz. Non-rigid range-scan alignment using thin-plate splines. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 759–765. IEEE Computer Society, 2004. 1

[5] Benedict J. Brown and Szymon Rusinkiewicz. Global non-rigid alignment of 3-d scans. *ACM Trans. on Graphics (TOG)*, 26(3):21–es, 2007. 1

[6] Wei Cao, Chang Luo, Biao Zhang, Matthias Nießner, and Jiapeng Tang. Motion2vecsets: 4d latent vector set diffusion for non-rigid shape reconstruction and tracking. *CoRR*, abs/2401.06614, 2024. 2

[7] Will Chang and Matthias Zwicker. Range scan registration using reduced deformable models. *Computer Graphics Forum*, 28(2):447–456, 2009. 1

[8] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[9] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 3

[10] Peng Huang, Chris Budd, and Adrian Hilton. Global temporal registration of multiple non-rigid surface sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[11] Qixing Huang, Xiangru Huang, Bo Sun, Zaiwei Zhang, Junfeng Jiang, and Chandrajit Bajaj. Arapreg: An as-rigid-as possible regularization loss for learning deformable shape generators. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 3

[12] Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[13] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *CoRR*, abs/2307.07635, 2023. 3

[14] Jiahui Lei and Kostas Daniilidis. Cadex: Learning canonical deformation coordinate space for dynamic surface representation via neural homeomorphism. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3, 7

[15] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum*, 27(5):1421–1430, 2008. 1

[16] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer Graphics Forum*, pages 1421–1430, 2008. 3, 6, 7

[17] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 2, 6, 7

[18] Yang Li and Tatsuya Harada. Lepard: Learning partial point cloud matching in rigid and deformable scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[19] Yang Li and Tatsuya Harada. Non-rigid point cloud registration with neural deformation pyramid. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 6, 7

[20] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. 2021. 2, 6

[21] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[22] Miao Liao, Qing Zhang, Huamin Wang, Ruigang Yang, and Minglun Gong. Modeling deformable objects from a single depth camera. In *IEEE International Conference on Computer Vision (ICCV)*, 2009. 2

[23] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[24] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470. Computer Vision Foundation / IEEE, 2019. 2

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[26] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[27] Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 3, 7

[28] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo

Martin-Brualla. Nerfies: Deformable neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 2

[29] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. 2017. 3

[30] 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 Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 45(8):9806–9821, 2023. 2

[31] Zheng Qin, Hao Yu, Changjian Wang, Yuxing Peng, and Kai Xu. Deep graph-based spatial consistency for robust non-rigid point cloud registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 4

[33] Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. Space-time surface reconstruction using incompressible flow. *ACM Trans. on Graphics (TOG)*, 27(5):110, 2008. 1, 2

[34] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Trans. on Graphics (TOG)*, 26(3):80, 2007. 1, 5

[35] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM Trans. on Graphics (TOG)*, pages 80–es. 2007. 2, 4

[36] Jiapeng Tang, Dan Xu, Kui Jia, and Lei Zhang. Learning parallel dense correspondence from spatio-temporal descriptors for efficient and robust 4d reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[37] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, Martin Bokeloh, Jens Kerber, and Hans-Peter Seidel. Animation cartography - intrinsic reconstruction of shape and motion. *ACM Trans. on Graphics (TOG)*, 31(2):12:1–12:15, 2012. 2

[38] Lingjing Wang, Jianchun Chen, Xiang Li, and Yi Fang. Non-rigid point set registration networks. *CoRR*, abs/1904.01428, 2019. 3

[39] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. *CoRR*, abs/2404.04319, 2024. 3, 4

[40] Hao Yu, Zheng Qin, Ji Hou, Mahdi Saleh, Dongsheng Li, Benjamin Busam, and Slobodan Ilic. Rotation-invariant transformer for point cloud matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2