S.1. Additional Technical Details

S.1.1. Spatio-temporal transformer

Our spatio-temporal transformer consists of several temporal and spatial attention blocks. For temporal attention blocks, we apply self-attention along the temporal dimension. For spatial blocks, we apply self-attention along the node dimension to aggregate information across different nodes. The transformer consists of 12 temporal blocks and 12 spatial blocks, with hidden dimension of 384.

S.1.2. Procrustes analysis for SE(3) estimation

Procrustes analysis is often used to estimate the affine transformation between two groups of data. More specifically, it calculates the direct solution for least-square optimization via singular value decomposition (SVD). Our implementation of the algorithm is shown below Algorithm 1:

Algorithm 1: Procrustes Analysis for Deformation Graph Estimation

```
Input: source node $\mathbf{v}_s^0$, neighbouring graph nodes
       $\{\mathbf{v}_s^k \mid k = 0, \cdots, K-1\}$, corresponding
       frame nodes $\{\mathbf{v}_p^k \mid k = 0, \cdots, K-1\}$
Output: $\mathbf{SE}(3)$ transformation $\mathcal{T}_p$ for $\mathbf{v}_s^0$
$\mathbf{t}_s \leftarrow \frac{1}{K}\Sigma_{k=1}^{K}\mathbf{v}_s^k$
$\mathbf{t}_p \leftarrow \frac{1}{K}\Sigma_{k=1}^{K}\mathbf{v}_p^k$
$\mathbf{X}_s = \{\mathbf{v}_s^k - \mathbf{t}_s \mid k = 0, \cdots, K-1\}$
$\mathbf{X}_p = \{\mathbf{v}_p^k - \mathbf{t}_p \mid k = 0, \cdots, K-1\}$
$\mathbf{U}, \mathbf{S}, \mathbf{V}^T = SVD(\mathbf{X}_s\mathbf{X}_p^T)$
$\mathbf{R} = \mathbf{U}\mathbf{V}^T$
if $\det(\mathbf{R}) = -1$ then
    multiply last row of $\mathbf{R}$ by $-1$
end
return $(\mathbf{R}, \mathbf{X}_p - \mathbf{X}_s)$
```

S.1.3. Selective graph nodes assignment

The details for our graph nodes assignment process for each source point are demonstrated in Alg. 2:

S.1.4. Training losses

The loss for training transformers Φ and Φ_R consists of registration loss, node regression loss and local rigidity constraint:

$$\mathcal{L}_{\text{total}} = \Sigma_{m=1}^{M}\alpha^{M-m}(\mathcal{L}_{\text{reg}}^m + \lambda_{\text{node}}\mathcal{L}_{\text{node}}^m + \lambda_{\text{rigid}}\mathcal{L}_{\text{rigid}}^m), \quad \text{(S1)}$$

where λ_{node} and λ_{rigid} are weight coefficients. In our experiments, we set them to 1.0 and 0.1, respectively.

The registration loss penalizes the L1 distance between predicted and ground truth positions:

$$\mathcal{L}_{\text{reg}}^m = \Sigma_{i=1}^{T_w}\|\hat{\mathbf{X}}_p^{i,m} - \mathbf{X}_p^i\|_1, \quad \text{(S2)}$$

Algorithm 2: Adaptive Nearest-neighbor Sampling

```
Input: source nodes $\mathbf{V}_s$, frame nodes
       $\mathcal{V} = \{\mathbf{V}_i \mid i = 1, \cdots, T\}$, query node $\mathbf{v}_q$,
       rigidity threshold $\epsilon$, minimal neighbor count
       $K_{min}$
Output: valid query node neighbors $\mathbf{V}_{out}$
$K \leftarrow K_{min}$
while true do
    $\mathbf{I} \leftarrow GetNeighborIndices(\mathbf{V}_s, \mathbf{v}_q, K)$
    $\mathbf{V}_{out} \leftarrow FindAssignmentSet(\mathbf{V}_s, \mathcal{V}, \mathbf{I}, \epsilon)$
    $K_{valid} \leftarrow GetCount(\mathbf{V}_{out})$
    if $K_{valid} \geq K_{min}$ then
    else
        $\epsilon \leftarrow \epsilon + 0.1$
        $K \leftarrow Min(K+1, GetCount(\mathbf{V}_s))$
    end
end
return $\mathbf{V}_{out}$
```

where $\hat{\mathbf{X}}_p^{i,m}$ is the predicted point position for the i-th frame at m-th iteration, and $\mathbf{X}_p^i$ denotes the corresponding ground truth. The registration loss give the supervision for the all skinned points, where the gradients indirectly back-propagated to the graph nodes and radii. However, in order to give the direct constrains on graph nodes, the node trajectories loss is defined similarly as:

$$\mathcal{L}_{node}^m = \Sigma_{i=1}^{T_w}\|\hat{\mathbf{V}}_p^{i,m} - \mathbf{V}_p^i\|_1, \quad \text{(S3)}$$

where $\hat{\mathbf{V}}_p^{i,m}$ and $\mathbf{V}_p^i$ denote predicted node graph positions, and their corresponding ground truth values. To further regularize node trajectory prediction and improve spatial awareness while allowing the freedom of large deformations, we utilize a local rigidity loss, which is defined as:

$$\mathcal{L}_{\text{rigid}}^m = \Sigma_{i=1}^{T_w}\|\hat{\mathbf{T}}^{i,m}(\hat{\mathcal{N}}_s(\mathbf{V}_s)) - \hat{\mathcal{N}}_s(\mathbf{V}_p^{i,m})\|_1, \quad \text{(S4)}$$

where $\hat{\mathbf{T}}^{i,m}(\cdot)$ denotes predicted transformations from source nodes to target frames, $\hat{\mathcal{N}}_s(\mathbf{V}_s)$ is the set of neighbors for source nodes, and $\hat{\mathcal{N}}_s(\mathbf{V}_p^{i,m})$ represents the corresponding node predictions.

S.2. Additional Results

S.2.1. More comparisons with registration methods

We show additional comparisons with baselines TCSR [1] and SyNoRim [3] on the D-FAUST depth-projected dataset. It's worth noting that TCSR takes around 8 hours to optimize per sequence, while SyNoRiM performs registration based on input pairs. In contrast, we perform feed-forward registration directly on sequential inputs without any further optimization. To ensure fair comparison, we separately trained 9 models on all testing sequences in D-FAUST for TCSR, with each one being trained for 200,000 iterations. The results in Tab. S1 show that our method achieves state-of-the-art performance across all metrics, while being fast and efficient.

Table S1. **Additional Baselines.** The results are evaluated on depth-projected D-FAUST test set with 50 frames per sequence. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported.

Method	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$
SyNoRiM	0.128	0.003	0.095
TCSR	0.074	0.115	0.518
Ours	**0.013**	**0.264**	**0.877**

S.2.2. Surface reconstruction methods

We conduct additional experiments by comparing our method with dynamic surface reconstruction methods including CaDeX [4], OFlow [8] and LPDC [9]. Quantitative results are shown in Tab. S2. The results show that our method achieves state-of-the-art performance across all metrics.

Table S2. **Comparison with CaDeX, OFlow and LPDC.** The evaluation is done on the test subsets of DT4D-A and D-FAUST. For each of the test sets, we follow OFlow and CaDeX to perform separate tests on unseen motions (UM) and unseen individual (UI) sequences. The evaluated metric is correspondence L2 error, which is consistent with ATE_{3D}.

Method	DT4D-A		D-FAUST	
	UM	UI	UM	UI
OFlow	0.204	0.285	0.094	0.117
LPDC	0.162	0.262	0.080	0.098
CaDeX	0.133	0.239	0.070	0.087
Ours	**0.073**	**0.086**	**0.067**	**0.067**

S.2.3. Comparison using pair-wise input

Although we focus on registering sequential inputs rather than pair-wise registration, we additionally compare our method against pair-wise registration methods [5–7]. Results in Tab. S3 show that our method out-performs pair-wise registration baselines even in their original settings.

S.2.4. Comparison on DT4D-H partial dataset

In addition to DT4D-A dataset which we use for evaluation in the main paper, we conduct experiments on the DT4D-H partial subset as well. Results in Tab. S4 demonstrate our advantage over all baselines.

Table S3. **Pair-wise comparison.** The results are evaluated on depth-projected DT4D-H test set with 2 frames per sequence. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported.

Method	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$
NSFP	0.111	0.040	0.244
NICP	0.082	0.075	0.297
NDP	0.100	0.028	0.280
Ours	**0.058**	**0.158**	**0.556**

Table S4. **Partial DT4D-H.** The results are evaluated on depth-projected DT4D-H test set with 50 frames per sequence. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported.

Method	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$
C-NSFP	0.112	0.012	0.163
C-NICP	0.102	0.037	0.178
C-NDP	0.102	0.011	0.194
Ours	**0.049**	**0.152**	**0.575**

S.2.5. Robustness to noise

Our model is trained on noisy inputs with noise standard deviation $\sigma_{noise} = 10^{-5}$. To evaluate the robustness of our method, we test it on different levels of noisy inputs, as shown in Tab. S5. The results show that our model performs with little accuracy decrease even when under large amount of noise with $\sigma_{noise} = 0.01$. The robustness owes to our iterative spatial-temporal transformer, which filters out the inconsistency in noisy inputs.

Table S5. **Different noise levels.** The results are evaluated on uniform D-FAUST test set with 24 frames per sequence. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported for noised inputs with different standard deviation σ_{noise}. Point input before applying noise is within unit cube.

σ_{noise}	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$
0.0	0.011	0.417	0.916
1.0×10^{-5}	0.011	0.416	0.918
1.0×10^{-4}	0.012	0.414	0.915
1.0×10^{-3}	0.012	0.413	0.915
1.0×10^{-2}	0.012	0.330	0.917
1.0×10^{-1}	0.095	0.001	0.067

S.2.6. Effectiveness of spatio-temporal refinement

The spatio-temporal refinement stage utilizes interleaving spatial and temporal attention, which aggregates structural and dynamic information to ensure consistency among frames and nodes. This is especially crucial for noisy and partial inputs where registration cannot be correctly and consistently performed with only a single frame. To further show the effectiveness of temporal refinement, we replace the temporal attention layers with the same number of spatial attention layers, and evaluate the model's performance after fine-tuning the new variant (Ours-SA). The results shown in Tab. S6 prove that our temporal refinement is crucial for yielding temporally consistent results, which is absent in most prior registration works.

Table S6. **Variations.** The results are evaluated on depth-projected D-FAUST test set with 24 frames per sequence. Average trajectory error ATE_{3D}, inlier proportions $\delta_{0.01}$ and $\delta_{0.05}$ are reported. Ours-SA replaces temporal attention with spatial attention.

Method	$ATE_{3D} \downarrow$	$\delta_{0.01} \uparrow$	$\delta_{0.05} \uparrow$
Ours	0.013	0.264	0.877
Ours-SA	0.029	0.042	0.742

S.2.7. Effectiveness of node selection

Our adaptive node selection method follows SurfelWarp [2] and utilizes the nearest neighbor as reference. We found in our experiments that this assumption achieves robust node selection with little noticeable errors. We show an example of its selective effect in Fig. S1.

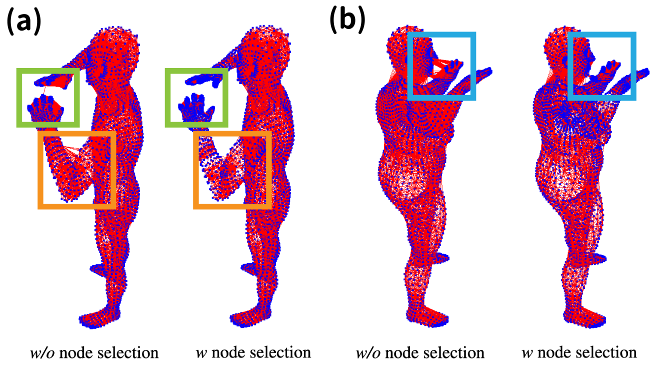

Figure S1. **Node selection.** Two cases demonstrating the effectiveness of our node selection algorithm. Red lines indicate pairs of source points and their corresponding nodes. Without node selection, nodes near source points might be incorrectly assigned.

References

[1] Jan Bednarík, Vladimir G. Kim, Siddhartha Chaudhuri, Shaifali Parashar, Mathieu Salzmann, Pascal Fua, and Noam Aigerman. Temporally-coherent surface reconstruction via metric-consistent atlases. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 1

[2] Wei Gao and Russ Tedrake. Surfelwarp: Efficient non-volumetric single view dynamic reconstruction. In *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. 3

[3] Jiahui Huang, Tolga Birdal, Zan Gojcic, Leonidas J. Guibas, and Shi-Min Hu. Multiway non-rigid point cloud registration via learned functional map synchronization. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 45(2):2038–2053, 2023. 1

[4] 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

[5] 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. 2

[6] 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.

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

[8] 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

[9] 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