CVPR2021

Neural Deformation Graphs for Globally-consistent Non-rigid Reconstruction

Aljaž Božič[1] Pablo Palafox[1] Michael Zollhöfer[2] Justus Thies[1] Angela Dai[1] Matthias Nießner[1]
[1]Technical University of Munich [2]Facebook Reality Labs

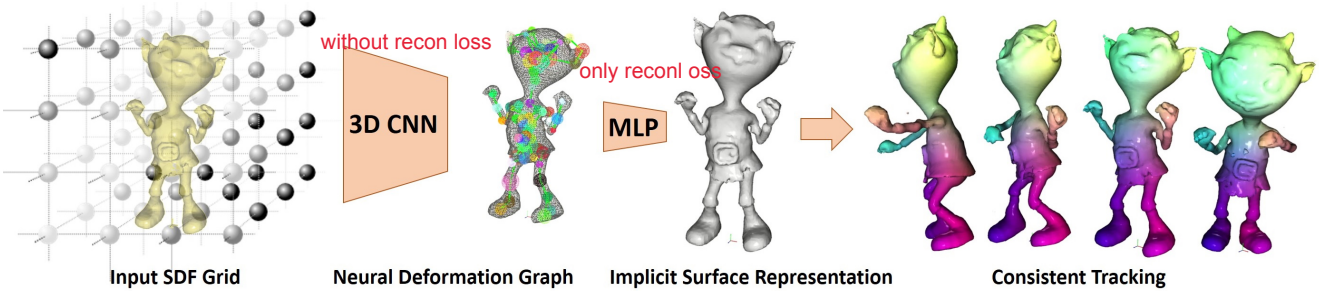

Figure 1: Neural Deformation Graphs: given range input data, represented as a signed distance field, our method predicts globally-consistent deformation graph that is used to reconstruct the non-rigidly deforming surface of an object. The surface of the object is represented as a set of implicit functions centered around the deformation graph nodes. Our global optimization provides consistent surface and deformation prediction, enabling robust tracking of an observed input sequence and even multiple disjoint captures of the same object (as we do not assume sequential input data).

Abstract

We introduce Neural Deformation Graphs for globally-consistent deformation tracking and 3D reconstruction of non-rigid objects. Specifically, we implicitly model a deformation graph via a deep neural network. This neural deformation graph does not rely on any object-specific structure and, thus, can be applied to general non-rigid deformation tracking. Our method globally optimizes this neural graph on a given sequence of depth camera observations of a non-rigidly moving object. Based on explicit viewpoint consistency as well as inter-frame graph and surface consistency constraints, the underlying network is trained in a self-supervised fashion. We additionally optimize for the geometry of the object with an implicit deformable multi-MLP shape representation. Our approach does not assume sequential input data, thus enabling robust tracking of fast motions or even temporally disconnected recordings. Our experiments demonstrate that our Neural Deformation Graphs outperform state-of-the-art non-rigid reconstruction approaches both qualitatively and quantitatively, with 64% improved reconstruction and 62% improved deformation tracking performance.

1. Introduction

Capturing non-rigidly deforming surfaces is essential towards reconstructing and understanding real-world environments, which are often highly dynamic. While impressive advances have been made in reconstructing static 3D scenes [8, 20], dynamic tracking and reconstruction remains very challenging. A plethora of domain-specific dynamic tracking methods has been developed (e.g., human bodies, faces, hands), leveraging strong domain shape and motion priors for robust tracking [4, 26, 29, 38]. However, real-world environments encompass a vast diversity of deformable objects – including people with clothing or animals – making domain specific shape priors often intractable for general deformable reconstruction; in this work, we thus focus on general non-rigid 3D reconstruction without shape or motion priors for general object tracking and reconstruction.

A seminal work in non-rigid 3D reconstruction is DynamicFusion [30], which was the first approach to demonstrate real-time dense reconstruction of dynamic scenes using just a single RGB-D sensor. DynamicFusion showed promissing results towards dynamic reconstruction, but still struggles in many real-world scenarios, which typically include strong deformations and fast frame-to-frame motion, due to its low-level, local correspondence association step.

In particular, the incremental construction of a deformation graph is prone to error aggregation and can lead to tracking failures. Recently, data-driven methods based on deep learning have been introduced [3, 24, 2] that learn priors of non-rigidly deforming objects from dense flow annotations. These approaches leverage a similar incremental deformation graph construction as DynamicFusion, but learn to establish more robust tracking via more sophisticated correspondence optimization based on data-driven priors. However, despite more robust correspondences, these methods still operate on a frame-by-frame basis, thus, aggregate tracking errors and are unable to recover if tracking fails. In order to address these shortcomings without assuming data-driven priors, we propose a globally-consistent neural deformation graph which allows for non-rigid reconstruction from commodity sensor observations, represented as signed distance fields (see Fig. 1). The neural deformation graph gives access to the per frame deformation graph nodes and stores the global graph connectivity. To robustly optimize for consistent deformations over fast motions, we introduce viewpoint consistency (independently for every frame) as well as graph and surface consistency constraints (between pairs of frames). Our viewpoint consistency loss measures the consistency of graph node position predictions w.r.t. rotation augmentation. The graph and surface consistency losses encourage deformations to be modeled in our Neural Deformation Graph such that local graph edge distances are preserved between frames and the deformed surface geometry of a source frame aligns well with the geometry of the target frame. Additionally, our approach does not assume temporally close frames, thus making it easily applicable to low FPS settings or the combination of independently captured depth recordings.

Since there exists no general canonical pose (like a T-pose of a human [26]) that fits all deformable objects, we avoid modeling it explicitly. Instead, we propose to employ a set of implicit functions that are centered around the deformation graph nodes. Specifically, we model local signed distance functions (SDFs) using multi-layer perceptrons (MLPs) that can be deformed to fit any frame, without requiring an explicit canonical pose. The global shape is evaluated by the integration of these local MLP predictions.

To summarize, our technical contributions are:

- a globally-optimized deformation graph that is able to handle deformations present in all frames of an unstructured dataset or a sequence of an object;
- a combination of per-frame viewpoint consistency and frame-to-frame graph and surface consistency for robust tracking of fast deformations;
- an implicit deformable multi-MLP shape representation anchored on the scene-specific deformation graph.

2. Related Work

Our approach is leveraging a low dimensional deformation graph to model the non-rigid deformations of an object, while the actual surface is represented by an implicit function by means of a multi-layer perceptron (MLP). We will discuss the most related approaches in these two fields.

Non-rigid Reconstruction Non-rigid reconstruction is a highly active research field, in particular using commodity RGB-D sensors such as the Kinect. The seminal work DynamicFusion of Newcombe et al. [30] tracks deformable motion and reconstructs the object's shape in an incremental fashion, i.e., frame-by-frame. While this approach relies on local depth correspondences, follow-up methods additionally use sparse SIFT features [19], dense color tracking [16] or dense SDF alignment [34, 35]. These methods show impressive results, but often struggle with fast frame-to-frame motion given their use of hand-crafted correspondences. Bozic et al. [3] introduced an annotated dataset of non-rigid motions that allows to train data-driven non-rigid reconstruction methods with learned correspondences [3, 24, 2]. While learned correspondences improve tracking performance, the approaches are still inherently limited by the employed frame-to-frame tracking paradigm, i.e., tracking errors accumulate over time, and if tracking is lost it is unable to recover. Tracking robustness can also be improved without any learned priors by using multi-view input data [7, 15] (setups with more than 50 cameras) and high-speed cameras [10] (8 cameras at 200 frames per second (FPS)). In contrast to these frame-to-frame tracking approaches, there are methods that focus on global non-rigid optimization [11, 41, 18, 32]; however, these methods either assume ground-truth optical flow [32], or they share the same drawbacks of the aforementioned frame-to-frame tracking approaches [11, 41, 18], and thus have difficulties handling fast deformable motion.

Deformation Graphs State-of-the-art non-rigid reconstruction methods often model deformations with a sparse deformation graph, following the Embedded Deformation [37] formulation. Deformation graphs offer a robust alternative to dense motion estimation with optical flow or scene flow methods, since they can estimate plausible motion even in partially occluded shape parts, when combined with motion regularization such as ARAP [36]. Existing non-rigid reconstruction approaches build the deformation graph incrementally, i.e., frame-by-frame, which can lead to unstable graph configurations in the case of tracking errors. In our approach, we predict a globally consistent deformation graph that can represent motion in all frames of the sequence, while being robust w.r.t. tracking errors present in challenging frames.

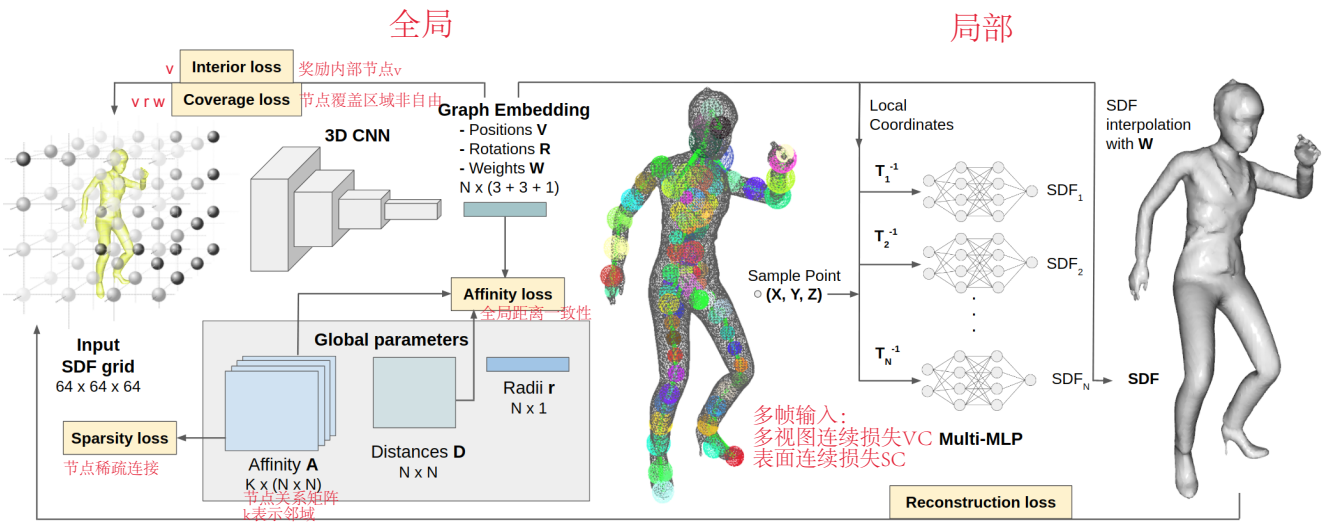

Figure 2: A Neural Deformation Graph encodes a 64^3 SDF grid to a graph embedding with graph node positions $\mathbf{V}$, rotations $\mathbf{R}$ and importance weights $\mathbf{W}$. To compute an SDF value for a sample point $(X, Y, Z) \in \mathbb{R}^3$, the point is transformed to local coordinates around each node, and passed through locally embedded implicit functions that are represented as MLPs; the global SDF value is computed by interpolating the local MLP predictions using the node radii $\mathbf{r}$ and importance weights $\mathbf{W}$. For graph regularization, a set of affinity matrices $\mathbf{A}_i \in \mathbb{R}^{N \times N}$ and a node-to-node distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ are globally optimized.

Sparse motion representations are common for human deformation modeling as well: the human skeletons used in [26, 4] are also instances of deformation graphs. Some works have tried to extend human skeletons to more general objects, but with limited success. In [1], a fixed generic skeleton is fitted to different object meshes, resulting in human-like re-animation of characters, but not general enough to be able to represent all degrees of freedom of general shapes. Fixed hierarchical deformation graphs are used for differentiable non-rigid tracking in [40], but a precomputed graph template is required, with fixed connectivity on different coarseness levels. Thus, it is only applicable to specific object types (e.g., used for hand tracking). Data-driven skeleton prediction has been introduced in [42], but it requires a dataset of manually designed skeletons as supervision, which is hard to obtain for general objects. Our method, instead, estimates both deformation graph nodes and connectivity of general deformable objects in a self-supervised manner.

Implicit Surface Representation Representing surface geometry implicitly with a signed distance field (SDF) has been extensively used in the non-rigid reconstruction community. An efficient algorithm for SDF grid construction from range images has been presented in [8] and extended to support non-rigid deformations in [30]. These methods rely on a discretized 3D grid to store the SDF, which can cause loss of detail, since grid resolution is limited by available memory. A promising direction is to not use discretized grids at all, but instead represent the SDF function continuously using a multi-layer perceptron (MLP), as introduced in [31, 33, 6]. An implicit surface representation is used in [17] for accurate human reconstruction, where the SDF is estimated in a canonical T-pose space. Since there exist no methods for estimation of canonical T-pose spaces for general non-rigid shapes, we instead base our method on the approach of Deng et al. [9]. Assuming ground-truth dense body and skeleton tracking, they represent the human body with multiple MLPs, one for each bone and in its own canonical space, centered around the bone, therefore eliminating the need for a T-pose space. In our general reconstruction approach, we estimate a deformation graph via self-supervision, and append an MLP to each deformation node to represent the surface of the observed object.

While most implicit reconstruction approaches do not produce consistent tracking, methods such as [12, 13, 39] reconstruct objects in a patch-based manner and empirically observe consistency of patches across different deformations. We compare our method, which leverages explicit consistency constraints, to these approaches to evaluate such implicit patch consistency.

3. Method

Given a sequence of signed distance fields observing a non-rigidly deforming surface, our method estimates the dense deformable motion in the sequence as well as reconstructs the geometry of the observed shape. Specifically, we apply self-supervised learning on the sequence that we want to reconstruct. A convolutional neural network that takes a

signed distance field (SDF) as input is trained to predict a consistent deformation graph. We call this neural network *Neural Deformation Graph*, as it implicitly stores the deformation graphs of each frame. Using the predicted graph node positions and orientations, we learn implicit functions to represent the shape of each graph node and, thus, the entire shape of the object. The implicit functions are represented as a multi-layer perceptron (MLP). These MLPs take a 3D point centered around the node position as input and predict its signed distance value, defining the local geometry around the node. Warping all node MLPs to every time step and interpolating their local part reconstruction results in an accurate *implicit deformable shape reconstruction* without the need for an explicit canonical pose. In addition to the sample point locations, the MLPs are conditioned on the predicted graph positions, which enables reconstruction of pose-dependent geometry detail. Using Marching Cubes [27], the geometry can be extracted as a mesh at every time step, with dense correspondences estimated throughout the entire sequence.

3.1. Neural Deformation Graphs

A deformation graph consists of graph nodes and graph edges. We represent the graph nodes $\mathcal{V}$ of each frame of the sequence implicitly by a neural network (Neural Deformation Graph). The graph connectivity is explicitly stored for the entire sequence as an affinity matrix $\mathcal{E} \in \mathbb{R}^{N \times N}$ ($N = |\mathcal{V}|$). A node $v \in \mathcal{V}$ is characterized by its 3D position $\mathbf{v} \in \mathbb{R}^3$, rotation $R \in \mathbb{R}^{3\times3}$, importance weight $w \in \mathbb{R}$ (describing importance of the node, explained below), radius $r \in \mathbb{R}$ (describing the spatial influence of the node), and a local implicit shape function f. We denote the set of node positions $\mathbf{V} = \{\mathbf{v}\}$, rotations $\mathbf{R} = \{R\}$, importance weights $\mathbf{W} = \{w\}$, radii $\mathbf{r} = \{r\}$, and shape functions $\mathbf{f} = \{f\}$, with $\mathcal{V} = (\mathbf{V}, \mathbf{R}, \mathbf{W}, \mathbf{r}, \mathbf{f})$.

To generate the signed distance fields needed for our method, we assume four calibrated cameras. The depth maps at each time step k are back-projected into a common coordinate system and converted into a signed distance field $\mathbf{S}_k$ of dimension 64^3 using static volumetric reconstruction [21]. Note that due to occlusions this representation is partial, thus only an approximate signed distance field is used for our deformation graph prediction. Based on this input, we estimate $(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k)$ using a Neural Deformation Graph (NDG) which is based on a 3D convolutional neural network (see supplemental material for architecture details):

$$(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = \mathrm{NDG}(\mathbf{S}_k).$$

The radii $\mathbf{r}$ of the graph nodes as well as the graph node affinities $\mathcal{E}$ are jointly optimized over the entire input sequence. In addition to the affinity matrix, we also store the average edge lengths (node-to-node distances) $\mathbf{D} \in \mathbb{R}^{N\times N}$, which are used for regularization. For every graph node, we also optimize for a local MLP which is used to represent the surface of the object (see Sec 3.3).

We define a fixed number of graph nodes ($N = 100$) in our experiments; note that this is an upper bound on the effective number of nodes, since the importance weights allow eliminating the effect of redundant nodes, making our method applicable to shapes of different size and structure complexity. To achieve a consistent graph node prediction via self-supervised training, we employ the following constraints for each time-step k.

Graph coverage loss. A deformation graph should cover the entire object to ensure that every deformable part can be represented while simultaneously enforcing that free space is not covered. To this end, we employ a loss that encourages the coverage of the shape by the node centers (w.r.t. their radii). We define the influence of a node (with position $\mathbf{v}$, radius $r > 0$, and importance weight $w > 0$) on a point $\mathbf{x} \in \mathbb{R}^3$ using a *weighted Gaussian function*:

$$\mathcal{G}(\mathbf{x}, \mathbf{v}, r, w) = w \cdot \exp\left(\frac{||\mathbf{x} - \mathbf{v}||_2^2}{r^2}\right)$$

The coverage of a point $\mathbf{x} \in \mathbb{R}^3$ is computed by summing the corresponding contributions of all nodes, and applying a *sigmoid* to encourage a fast transition from covered (where coverage value is 1) to free space (where coverage value should be 0), enabling more accurate surface coverage:

$$\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) = \sigma\left(s\left(\left(\sum_{\mathbf{v},r,w} \mathcal{G}(\mathbf{x}, \mathbf{v}, r, w)\right) - d\right)\right)$$

We empirically set $d = 0.07$ and $s = 100.0$. To compute the coverage loss, we sample points P_{un} uniformly in the shape's bounding box and points P_{ns} near the surface region. Points are assigned coverage value of $c = 0$ if they are visible in at least one of the cameras, otherwise they are assigned $c = 1$. The coverage loss then compares predicted coverage of these point samples with the pre-computed coverage using an ℓ_2 loss:

$$\mathcal{L}_{\text{coverage}} = \lambda_{\text{un}} \sum_{(\mathbf{x},c)\in P_{\text{un}}} ||\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c||_2^2 + \lambda_{\text{ns}} \sum_{(\mathbf{x},c)\in P_{\text{ns}}} ||\mathcal{C}(\mathbf{x}, \mathbf{V}_k, \mathbf{r}, \mathbf{W}_k) - c||_2^2$$

内部 近表面

Node interior loss. In addition to the graph coverage loss, we require the node positions to be predicted inside the shape. If any node's position $\mathbf{v}$ is predicted outside the observed surface $\mathbf{S}_k$, i.e., in the SDF region with positive signed distance value, we penalize it to encourage the node's position to be inside the surface:

$$\mathcal{L}_{\text{interior}} = \sum_{\mathbf{v}\in\mathbf{V}_k} \max\left(\text{interp}(\mathbf{S}_k, \mathbf{v}), 0\right)$$

Here $\mathrm{interp}(\mathbf{S}_k, \mathbf{v})$ is the trilinear interpolation of $\mathbf{S}_k$ at $\mathbf{v}$.

Affinity consistency loss. We also optimize for a global affinity matrix $\mathcal{E} = \{e_{ij} \mid i \in [1, N], j \in [1, N]\}$ representing node-to-node affinities across the entire input sequence. We compute node-to-node Euclidean distances $||\mathbf{v}_i^k - \mathbf{v}_j^k||_2$ at each frame k, and weight them by connectivity weights e_{ij}; this should remain consistent over the whole sequence (relative loss, preserving edge length) and relatively small (absolute loss, preferring close-by connections). To ensure global distance consistency, we additionally optimize over average node-to-node distances d_{ij}, resulting in the affinity loss:

$$\mathcal{L}_{\text{affinity}} = \lambda_{\text{rel}} \sum_{i \neq j} e_{ij} \left| d_{ij}^2 - ||\mathbf{v}_i^k - \mathbf{v}_j^k||_2^2 \right|_1 + \lambda_{\text{abs}} \sum_{i \neq j} e_{ij} ||\mathbf{v}_i^k - \mathbf{v}_j^k||_2^2$$

Sparsity loss. We enforce a sparse connectivity of the graph. Specifically, each node can have up to K neighbors ($K = 2$ in our setting); we use a (soft) loss to encourage these neighbors to be different. To achieve this, we optimize over a set of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_K \in \mathbb{R}^{N \times N}$, and construct $\mathcal{E} \in \mathbb{R}^{N \times N}$ as:

$$\mathcal{E} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{softmax}(\mathbf{A}_i)$$

We use $\mathrm{softmax}$ over the rows of matrix $\mathbf{A}_i$ to guarantee all affinity elements of a node to be positive and add up to 1. To enforce unique graph neighbors, a sparsity loss is employed, encouraging different matrices $\mathbf{A}_i$ to produce different neighbors:

$$\mathcal{L}_{\text{sparsity}} = \sum_{l \neq m} ||\mathrm{softmax}(\mathbf{A}_l) \odot \mathrm{softmax}(\mathbf{A}_m)||_F^2$$

We use $\odot$ to denote the element-wise product.

3.2. Global Deformation Optimization

We compute deformation between any pair of frames by interpolating the nodes' relative motions (translations and rotations), weighted by their influences $\mathcal{G}$. For a source frame s and target frame t, the warping of point $\mathbf{x} \in \mathbb{R}^3$ from frame s to frame t is defined as:

$$\mathcal{W}_{s \to t}(\mathbf{x}) = \sum_{i=1}^{N} \mathcal{G}(\mathbf{x}, \mathbf{v}_i^s, r_i, w_i^s)(\mathbf{R}_i^t (\mathbf{R}_i^s)^{\mathrm{T}} (\mathbf{x} - \mathbf{v}_i^s) + \mathbf{v}_i^t)$$

We denoted parameters at the source frame with $(\cdot)^s$ and at the target frame with $(\cdot)^t$. We use the Embedded Deformation formulation [37] to parameterize frame-to-frame deformation, but instead of fixed-radius skinning we employ node influence $\mathcal{G}$ as the skinning weight, which enables different skinning effects for every node as well as frame-adaptive skinning, i.e., skinning can change depending on the deformation. To ensure globally consistent deformation, we employ a per-frame viewpoint consistency loss and a surface consistency loss.

Viewpoint consistency loss. Since input observations may see very different views, we enforce a viewpoint consistency loss for consistent graph node predictions across varying views. To this end, for each frame k, the rotated 3D input $\mathbf{S}_k$ should produce consistent graph node positions $\mathbf{V}_k$, rotations $\mathbf{R}_k$ and importance weights $\mathbf{W}_k$. In our experiments, we only consider view rotations around the y-axis, since the camera setup is arranged in the x-z plane. In each batch, we sample two random angles α and β for every sample, and compute rotated inputs $\pi_\alpha(\mathbf{S}_k)$ and $\pi_\beta(\mathbf{S}_k)$ by trilinear re-sampling of input SDF grid $\mathbf{S}_k$ using rotated grid indices. Viewpoint consistency is then measured by:

$$\mathcal{L}_{\text{vc}} = ||\pi_\alpha^{-1}\mathrm{NDG}(\pi_\alpha(\mathbf{S}_k)) - \pi_\beta^{-1}\mathrm{NDG}(\pi_\beta(\mathbf{S}_k))||_2^2$$

where the function π_ϕ^{-1} corrects for the input rotation of angle ϕ: $\pi_\phi^{-1}(\mathbf{V}_k, \mathbf{R}_k, \mathbf{W}_k) = (R_\phi^{\mathrm{T}}\mathbf{V}_k, R_\phi^{\mathrm{T}}\mathbf{R}_k, \mathbf{W}_k)$.

Surface consistency loss. Surface points from a source frame s should, after deformation to a target frame t, align well with the target frame's SDF grid $\mathbf{S}_t$. We sample surface points P_{s} in the source frame and warp them to the target frame using the predicted deformation, to trilinearly interpolate the target grid $\mathbf{S}_t$, encouraging surface points to be warped to near zero (surface) SDF values:

$$\mathcal{L}_{\text{sc}} = \sum_{\mathbf{x} \in P_{\text{s}}} (\mathrm{interp}(\mathbf{S}_t, \mathcal{W}_{s \to t}(\mathbf{x})))^2$$

This consistency loss is computed between pairs of samples in the batch, with uniformly sampled batch samples.

3.3. Implicit Surface Reconstruction

We represent the surface of the object as an implicit function. Specifically, each graph node i defines local geometry over the influence of that node, with an implicit function f_i, represented by an MLP. This MLP takes a location in the local space as input and outputs an SDF value. Any point $\mathbf{x} \in \mathbb{R}^3$ in the current frame k can be transformed to the local coordinate system of node i as $\mathcal{W}_{k,i}^{-1}(\mathbf{x}) = \mathbf{P}_{\text{enc}}((\mathbf{R}_i^k)^{\mathrm{T}}(\mathbf{x} - \mathbf{v}_i^k))$. $\mathbf{P}_{\text{enc}} : \mathbb{R}^3 \to \mathbb{R}^F$ denotes positional encoding that transforms 3D local coordinates to a high-dimensional frequency domain (in our case $F = 30$), as presented in [28]. Inspired by Deng et al. [9], we condition each f_i on the predicted input frame's

graph parameters, such that they can encode pose-specific geometry details. We train a linear layer $\Pi_i(\cdot)$ to select a sparse pose code (of dimension $D = 32$) for every f_i from the graph predictions $\mathrm{NDG}(\mathbf{S}_k)$. Given this input of dimension $D + F$, we use 8 linear layers with feature dimension of 32, a leaky ReLU (with negative slope of 0.01) as activation function, and skip connections between the input and the 6th linear layer.

We compute the full surface reconstruction $\mathcal{S}_k$ as an SDF created from interpolating the SDF output values of each local MLP f_i, using the aforementioned skinning weights and transformations to the current frame by the estimated nodes' rotations and translations:

$$\mathcal{S}_k(\mathbf{x}) = \sum_{i=1}^{N} \mathcal{G}(\mathbf{x}, \mathbf{v}_i^k, r_i, w_i^k) f_i(\mathcal{W}_{k,i}^{-1}(\mathbf{x}), \Pi_i(\mathrm{NDG}(\mathbf{S}_k)))$$

This operation is efficiently implement using group convolutions. During training, we use the same point samples P_{un} and P_{ns} as for the graph coverage loss, sampled uniformly and near the surface, but instead of the $0/1$ coverage values we use their approximate SDF values. We then optimize for $\{f_i\}$ using the SDF reconstruction loss:

$$\mathcal{L}_{\mathrm{recon}} = \sum_{(\mathbf{x},\mathrm{sdf}) \in P_{\mathrm{un}} \cup P_{\mathrm{ns}}} |\mathcal{S}_k(\mathbf{x}) - \mathrm{sdf}|_1$$

3.4. Training Details

We use the Adam solver [22] with momentum of 0.9 to optimize the complete loss:

$$\mathcal{L} = \mathcal{L}_{\mathrm{coverage}} + \lambda_{\mathrm{interior}}\mathcal{L}_{\mathrm{interior}} + \mathcal{L}_{\mathrm{affinity}} + \lambda_{\mathrm{sparsity}}\mathcal{L}_{\mathrm{sparsity}} + \lambda_{\mathrm{vc}}\mathcal{L}_{\mathrm{vc}} + \lambda_{\mathrm{sc}}\mathcal{L}_{\mathrm{sc}} + \lambda_{\mathrm{recon}}\mathcal{L}_{\mathrm{recon}}$$

控制变量法 降低损失

Our method is trained in two stages. We initially train the CNN encoder with all losses except the reconstruction loss, and afterwards train the multi-MLP network using only the reconstruction loss, with the CNN encoder frozen.

The CNN encoder is trained for 500k iterations with a learning rate of $5\mathrm{e}^{-5}$ and a batch size of 16; we balance the losses with $\lambda_{\mathrm{un}} = 1.0$, $\lambda_{\mathrm{ns}} = 0.1$, $\lambda_{\mathrm{interior}} = 1.0$, $\lambda_{\mathrm{rel}} = 0.1$, $\lambda_{\mathrm{abs}} = 0.1$, $\lambda_{\mathrm{sparsity}} = 1\mathrm{e}^{-8}$, $\lambda_{\mathrm{vc}} = (10.0, 1.0, 1\mathrm{e}^{-4})$ (for graph node's position, weight and rotation, respectively) and $\lambda_{\mathrm{sc}} = 1\mathrm{e}^{-6}$. Every 50k iterations we increase the loss weights λ_{rel}, λ_{abs}, $\lambda_{\mathrm{sparsity}}$, λ_{sc} by a factor of 10, up to maximum weights $\lambda_{\mathrm{rel}}^{\mathrm{max}} = 10000.0$, $\lambda_{\mathrm{abs}}^{\mathrm{max}} = 1.0$, $\lambda_{\mathrm{sparsity}}^{\mathrm{max}} = 1\mathrm{e}^{-3}$ and $\lambda_{\mathrm{sc}}^{\mathrm{max}} = 1000.0$.

The multi-MLP network is trained for 500k iterations with a learning rate of $5\mathrm{e}^{-4}$ and a batch size of 4, only based on the reconstruction loss with $\lambda_{\mathrm{recon}} = 1.0$.

4. Results

To evaluate our proposed approach, we conducted a series of experiments on real and synthetic recordings where ground truth data is available.

Method	Chamfer	EPE3D
SIF [12]	1.12	8.56
LDIF [13]	2.41	10.40
Multiview DynamicFusion [30]	2.19	3.06
Ours: GRAPH	0.50	8.93
Ours: GRAPH + AO	0.46	8.03
Ours: GRAPH + AO + VC	0.44	4.12
Ours: GRAPH + AO + VC + SC	**0.40**	**1.16**

Table 1: We show quantitative comparisons with state-of-the-art approaches, evaluating geometry using chamfer distance ($\times 10^{-4}$), and deformation using EPE3D ($\times 10^{-2}$). We also include an ablation study of different components of our method: GRAPH = coverage and interior losses, AO = affinity optimization with affinity consistency and sparsity, VC = viewpoint consistency, SC = surface consistency.

Evaluation on Synthetic Data In order to quantitatively and qualitatively evaluate our method, we make use of synthetic human-like and character sequences from the DeformingThings4D dataset [25]. To mimic our real data capture setup, we render 4 fixed depth views for every frame of synthetic animation, and generate SDF grids from these 4 views. Quantitative evaluation is executed on three sequences, including human, character and bear motion, as shown in Fig. 3.

We compare our approach to state-of-the-art network-based reconstruction methods, SIF [12] and LDIF [13], as well as to the non-rigid reconstruction approach DynamicFusion [30], which we extend to the multi-view domain. Both SIF and LDIF execute part-based shape reconstruction, predicting an ellipsoid for every shape part. While the methods do not explicitly focus on deformation tracking, we add dense tracking computation via ellipsoid motion interpolation between two frames, represented by relative translation and rotation. Our multi-view DynamicFusion baseline is a specialized framework for both non-rigid tracking and reconstruction, with coarse-to-fine multi-frame alignment using depth iterative closest point (ICP) correspondences and non-rigid volumetric fusion.

In Tab. 1, we evaluate the deformation estimation and geometry reconstruction performance of these methods in comparison to our approach. The depth data of every sequence is normalized such that the largest bounding box side length is equal to 1.0. All numbers are listed w.r.t. this unit cube, thus, being independent to the scale of the objects. The geometry reconstruction is evaluated using *L2 Chamfer distance*, which computes average squared bi-directional point-to-point distance between reconstructed and ground truth geometry for every time step, thus eval-

深度归一化

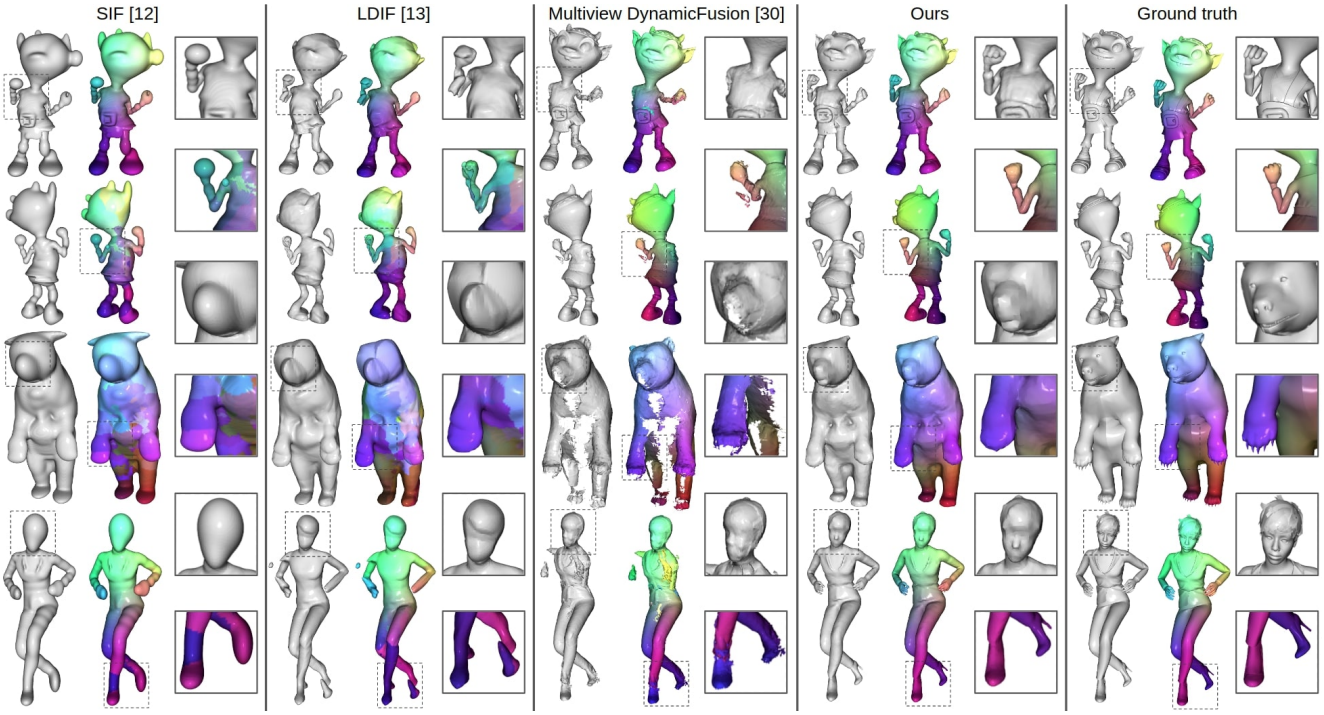

Figure 3: We qualitatively compare our method to the baseline methods on synthetic data. Colors represent corresponding locations in the first frame of the sequence and, thus, giving a notion of the tracking quality and consistency. Our approach outperforms state of the art in both reconstruction and deformation tracking quality.

uating accuracy and completeness of geometry. For deformation evaluation, we uniformly sample 10 keyframes per sequence (with fixed step size), and evaluate dense deformation from each of these keyframes to any other frame by deforming the ground truth points at a keyframe to every other frame. As deformation metric we use the *End-Point-Error (EPE3D)*, measuring average L2 distance between estimated keyframe-to-frame deformations and ground truth deformations.

Among the baselines the best reconstruction performance (lower Chamfer distance) is achieved by SIF [12], while multi-view DynamicFusion [30] obtains better deformation tracking performance (lower EPE3D). However, our approach outperforms all methods on both reconstruction and deformation tracking metrics, achieving 64% better reconstruction and 62% better deformation tracking results. The improvement is also clearly visible in the qualitative comparisons shown in Fig. 3. We visualize tracking performance by a color gradient that is defined in the initial frame of each sequence. Specifically, each point is given a color value w.r.t. its location in the bounding box of the first frame. With perfect tracking and reconstruction, a specific point on the surface will have the same color throughout the sequence. Errors in tracking result in wrong surface colors. The methods SIF [12] and LDIF [13] produce less accurate geometry reconstruction with tracking outliers under larger deformation (e.g., flipped legs in the human sequence). Multi-view DynamicFusion [30] suffers from incomplete geometry because of the incremental graph construction and surface integration. DynamicFusion can also not recover from tracking failures, i.e., shape parts where frame-to-frame tracking failed start to disappear and are integrated at a different location. In contrast, our method is robust in the case of larger deformations and produces complete and accurate geometry reconstruction.

设定颜色
观测变形

Evaluation on Real Data. Our real data capture setup consists of 4 Kinect Azure sensors with hardware synchronization. The cameras are calibrated with a checkerboard using OpenCV [23] and an additional refinement procedure based on ICP [5]. Before recording an actual sequence, we record the background to compute the floor plane using PCA. During capture, we filter out floor points and background points, i.e., all points outside of a cylinder with diameter 1.8 m and height 2.5 m. We use the wide-field-of-view depth capture setting with a resolution of 1024×1024 pixels, at the highest available frame-rate of 15 FPS for this resolution. In Fig. 4, we show a comparison between the multi-view DynamicFusion approach [30] and ours. Our approach achieves considerably more accurate deformation

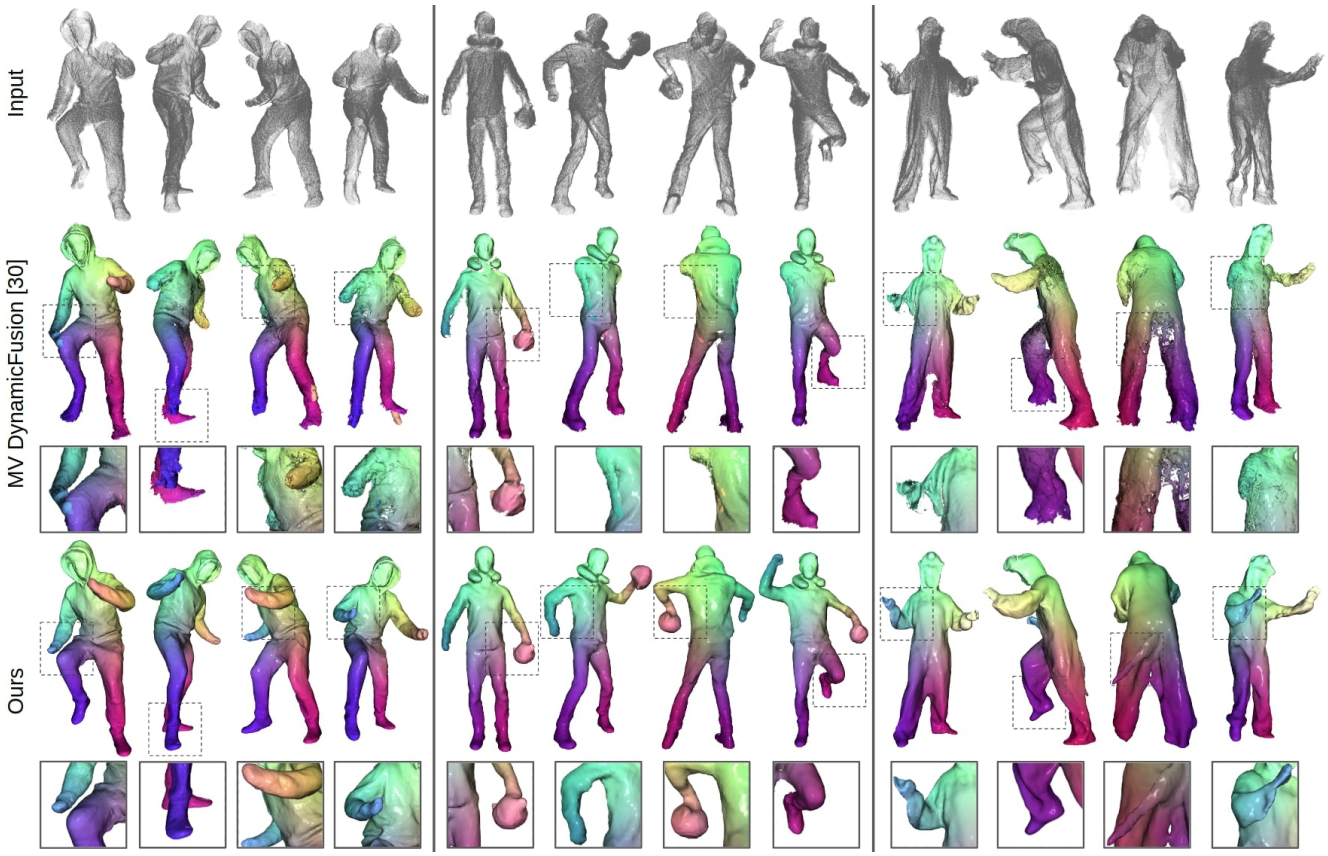

Figure 4: We show qualitative comparisons of our method with multi-view DynamicFusion [41] on real sequences captured by four Kinect Azure sensors. Colors represent corresponding locations in the first frame of the sequence to visualize the tracking quality and consistency.

tracking (color is retrieved from the first frame) while also producing more complete and accurate geometry reconstruction. More qualitative results are shown in the accompanying video.

Ablation study of network components. To evaluate specific parts of our method, we employ an ablation study. Specifically, we analyzed the performance of our method by performing optimization without using certain losses: without affinity related losses (affinity consistency and sparsity losses), viewpoint consistency loss and surface consistency loss. As shown in Tab. 1, using these additional losses vastly improves the method's performance. Especially, it results in a much lower EPE3D error, and, thus, in globally consistent tracking performance.

Limitations. Using our globally consistent Neural Deformation Graph, we show state-of-the-art tracking and reconstruction quality. Currently, our quality is limited by the input, i.e., a 64^3 SDF grid. Sparse 3D convolutions [14] could be applied to cope with higher resolutions. Our approach focuses on the tracking and reconstruction of the geometry, and not the texture. A texture on top of the tracked geometry could be estimated (similar to the color scheme that we show in the results figures) and additional losses based on this texture could be employed. Based on our approach, we believe that there is a potential of several follow-up works with adapted loss formulations and additional features like color reconstruction.

5. Conclusion

We introduced Neural Deformation Graph which allows to reconstruct and track non-rigidly deforming objects in a globally consistent fashion. It is enabled by a neural network that implicitly stores the deformation graph of the object. The network is trained with losses on global consistency, resulting in tracking and reconstruction quality that surpasses the state of the art by more than 60% w.r.t. the respective metrics. We believe that our global optimization of non-rigid motion will be a stepping stone to learn data-driven priors in the future.

Acknowledgments

This work was supported by the ZD.B (Zentrum Digitalisierung.Bayern), a TUM-IAS Rudolf Mößbauer Fellowship, the ERC Starting Grant Scan2CAD (804724), and the German Research Foundation (DFG) Grant Making Machine Learning on Static and Dynamic 3D Data Practical.

Appendix

In this appendix, we provide further details about our proposed neural deformation graphs. Specifically, we describe the network architectures in detail (Sec. A), give additional training information (Sec. B), and present our capture setup used for recording real non-rigid motion sequences using Kinect Azure sensors (Sec. C).

A. Network Architecture

Our method is composed of two learned components; the neural deformation graph based on a 3D CNN (shown in Fig. 5) and a set of multi-layer perceptrons (MLPs) which are used to implicitly represent the surface (see Fig. 8).

Neural Deformation Graph. The neural deformation graph implicitly stores the deformation graphs for each input frame k. The underlying 3D CNN encoder takes a dense 64^3 SDF grid $\mathbf{S}_k$ as input and outputs the $(3+3+1)N$ dimensional vector, representing position, rotation (in axis-angle format) and importance weight for each graph node (N being the number of graph nodes). The input grid is encoded to a spatial dimension of 4^3 and a feature dimension of 64 using 4 blocks of convolutional downsampling with stride of 2, followed by 2 residual units (with ReLU activation unit and batch norm). The downscale operation is detailed in Fig. 6. We use a linear layer to convert the $4^3 \cdot 64$ features to dimension 2048. This feature vector is input to two independent heads, each composed of 3 linear layers with feature dimension of 2048 and a leaky ReLU as activation unit with negative slope of 0.01 (no activation function after the last linear layer). One head predicts graph node rotations $\mathbf{R}_k$ of dimension $3N$, while the other head predicts graph node positions $\mathbf{V}_k$ and weights $\mathbf{W}_k$ (total dimension of $(3+1)N$). We concatenate both predictions, which results in a graph embedding of dimension $7N$. For all our experiments, we use $N = 100$.

Implicit Surface Representation For each graph node i, we train a separate MLP f_i that predicts the signed distance field around the node, as presented in Fig. 8. As explained in the main paper, the complete shape geometry is reconstructed by warping each node's local SDF using estimated node deformations and interpolating the local

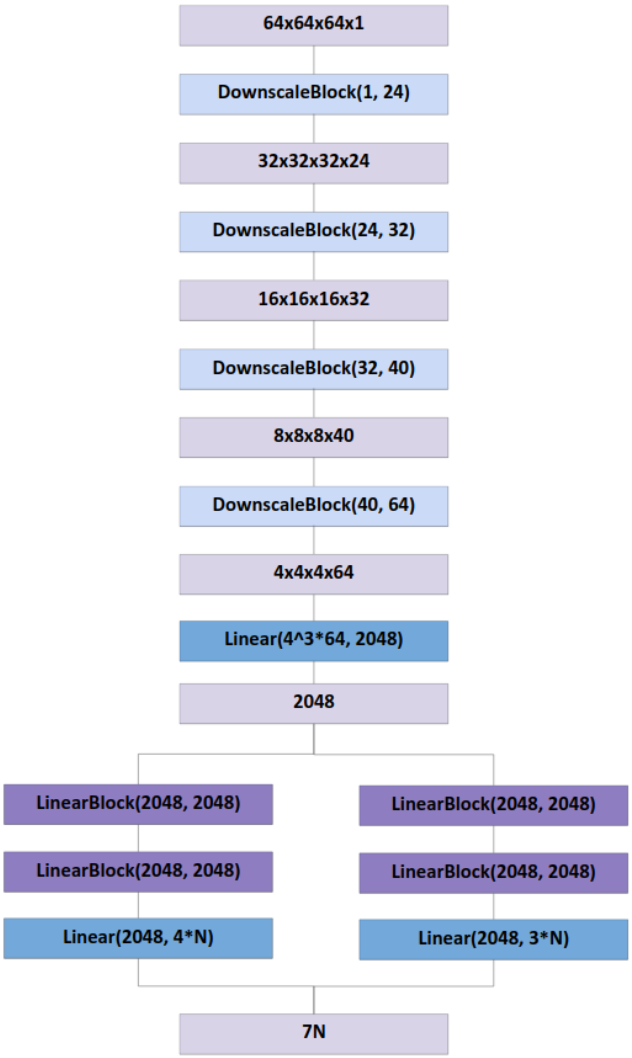

Figure 5: Our neural deformation graph is represented as a 3D CNN which takes as input a 64^3 SDF grid and outputs $7N$ graph node parameters. The architectures of the single computation blocks are detailed in Fig. 6 and Fig. 7.

SDFs using the estimated node interpolation weights (computed from node radii and importance weights). Each MLP takes a 3D position as input, and outputs an SDF value for the given position. The position is represented with positional encoding [28] of dimension $F = 30$. To reconstruct pose-dependent details, the MLP is conditioned on a pose code. This D-dimensional pose code ($D = 32$ in our experiments) is computed by inputting the current deformation graph predictions through a linear projection layer Π_i shown in Fig. 9. Thus, in total, the MLP takes an input of dimension $D + F$, and using 8 linear layers with feature dimension of 32, it outputs the SDF value of the corresponding node. There is a skip connection between the input and

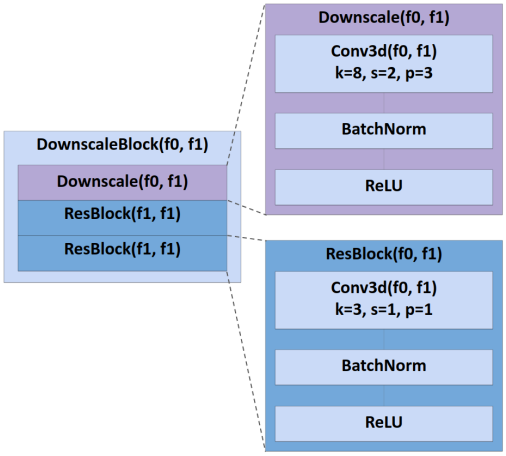

Figure 6: The downscale block used in our 3D CNN architecture (see Fig. 5) reduces the spatial dimension by 2.

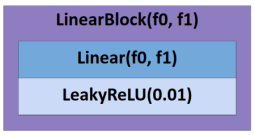

Figure 7: Each linear block of the 3D CNN architecture (see Fig. 5) applies a linear layer, followed by a leaky ReLU with negative slope of 0.01.

Approach	Chamfer
Without pose conditioning	0.46
With pose conditioning	0.40

Table 2: Evaluation of the pose conditioning on reconstructions on synthetic data using Chamfer distance ($\times 10^{-4}$).

the sixth linear layer, and a leaky ReLU with negative slope of 0.01 is applied after each linear layer, as shown in Fig. 7.

In Tab. 2, we evaluate the influence of the pose codes with respect to the reconstruction quality measured using a Chamfer distance. The pose conditioning improves the Chamfer distance by a large margin, which is also visible in the qualitative comparison in Fig. 10.

B. Training Details

Our method's training process is described in Sec. 3.4 of the main paper. In this section, we specify the sampling strategy during training of both the neural deformation graph and the implicit surface representations (see visualization of samples in Fig. 11).

When training the neural deformation graph, we use $|P_{\text{un}}| = 3000$ uniform point samples, $|P_{\text{ns}}| = 3000$ near surface point samples and $|P_{\text{s}}| = 3000$ surface point samples, sampled randomly for each batch from 100k preprocessed point samples. For the graph coverage loss, we apply an additional weight of 10.0 for interior point samples (determined by the SDF sign).

To train our multi-MLP network that implicitly represents the surface, we use $|P_{\text{un}}| = 1500$ uniform point samples and $|P_{\text{ns}}| = 1500$ near surface point samples. Note that this reduced set of samples is applied to satisfy mem-

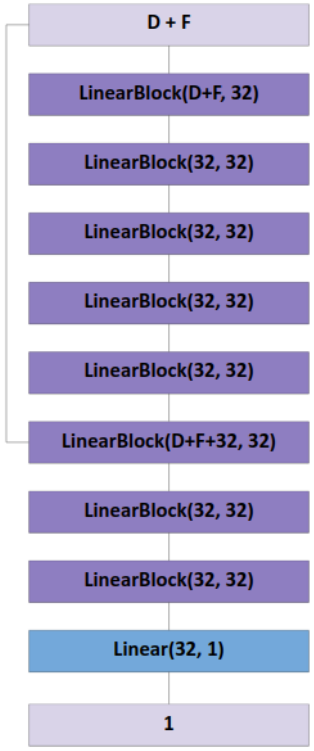

Figure 8: We represent the surface of an object using an implicit function (SDF) which is based on multi-layer perceptrons (MLPs). Each graph node MLP takes as input a $(D + F)$-dimensional vector (consisting of the pose code and the sample position, represented with positional encoding [28] in local coordinates) and outputs the SDF value.

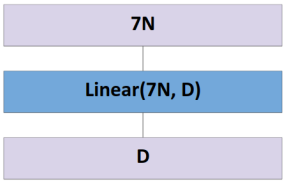

Figure 9: To compute the sparse pose code of the graph, we use a single linear projection layer that converts the graph parameters of dimension $7N$ to a code of dimension D (we use $N = 100$ and $D = 32$). This pose code is input to the local MLP that represents the pose-dependent local surface.

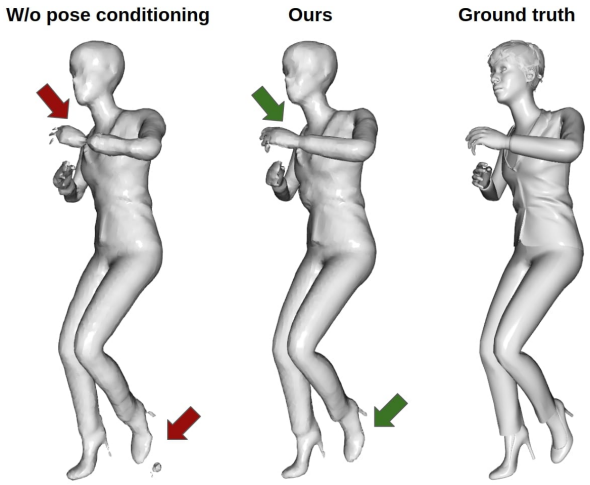

Figure 10: We compare our implicit reconstruction with pose conditioning (middle) to without pose conditioning (left). Pose conditioning clearly improves reconstruction performance in regions of very strong deformation.

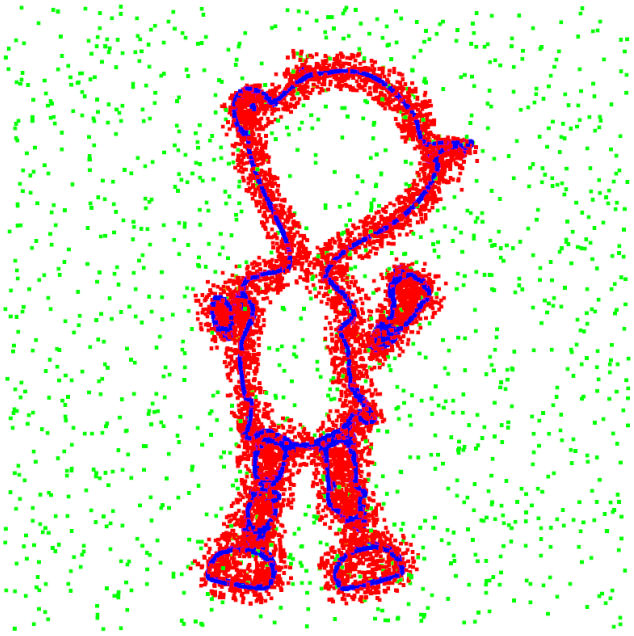

Figure 11: We visualize the uniform samples P_{un} (green), near surface samples P_{ns} (red) and the surface samples P_{s} (blue) for a slice of samples with $z \in [-0.01, 0.01]$ at one frame of the character sequence shown in the Fig. 1 of the main paper.

ory limits of our used GPU (Nvidia Geforce 2080Ti). We use a truncation of 0.1 (in normalized units of the object in the unit cube) for the signed distance field used for the reconstruction loss.

Figure 12: We capture non-rigid motion using 4 Kinect Azure sensors, which are pre-calibrated and hardware-synced to guarantee spatial and temporal coherence of depth captures. We capture depth images at resolution of 1024×1024, using the highest available frame-rate of 15 FPS.

Note, when interpolating the SDF grid for the node interior loss, in the case that a graph node's position is predicted outside the grid, we define the out-of-grid loss by an ℓ_2-distance to the nearest bounding box corner. This encourages graph node positions to be always predicted inside the shape's bounding box.

C. Real Data Capture Setup

In the main paper as well as in the supplemental video, we demonstrate that our method performs well on real data. To capture real-world non-rigid motion sequences, we record with four Kinect Azure sensors as shown in Fig. 12. All four sensors are connected to the same computer via USB-C cables and are hardware-synced using a daisy-chain configuration (connecting the trigger of the master camera with the other 3 cameras in a chain). To avoid interference between depth sensors, we set a delay of 160 microseconds between different depth camera captures.

References

[1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 3

[2] Aljaž Božič, Pablo Palafox, Michael Zollöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking. In *NeurIPS*, 2020. 2

[3] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018. 1, 3

[5] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10:145–155, 01 1992. 7

[6] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. 3

[7] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)*, 34(4):1–13, 2015. 2

[8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 1, 3

[9] Boyang Deng, JP Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. Neural articulated shape approximation. In *The European Conference on Computer Vision (ECCV)*. Springer, August 2020. 3, 5

[10] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2fusion: Real-time volumetric performance capture. *ACM Transactions on Graphics (TOG)*, 36(6):1–16, 2017. 2

[11] Mingsong Dou, Jonathan Taylor, Henry Fuchs, Andrew Fitzgibbon, and Shahram Izadi. 3d scanning deformable objects with a single rgbd sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 493–501, 2015. 2

[12] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Deep structured implicit functions. *arXiv preprint arXiv:1912.06126*, 2019. 3, 6, 7

[13] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 3, 6, 7

[14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018. 8

[15] Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (TOG)*, 38(6):1–19, 2019. 2

[16] Kaiwen Guo, Feng Xu, Tao Yu, Xiaoyang Liu, Qionghai Dai, and Yebin Liu. Real-time geometry, albedo, and motion reconstruction using a single rgb-d camera. *ACM Transactions on Graphics (TOG)*, 36(3):32, 2017. 2

[17] Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2020. 3

[18] Matthias Innmann, Kihwan Kim, Jinwei Gu, Matthias Nießner, Charles Loop, Marc Stamminger, and Jan Kautz. Nrmvs: Non-rigid multi-view stereo. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2754–2763, 2020. 2

[19] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016. 2

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1

[21] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 4

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[23] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 7

[24] Yang Li, Aljaz Bozic, Tianwei Zhang, Yanli Ji, Tatsuya Harada, and Matthias Nießner. Learning to optimize non-rigid tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4910–4918, 2020. 2

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

[26] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 1, 2, 3

[27] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 163–169, New York, NY, USA, 1987. Association for Computing Machinery. 4

[28] 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. *arXiv preprint arXiv:2003.08934*, 2020. 5, 9, 10

[29] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018. 1

[30] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. 1, 2, 3, 6, 7

[31] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 3

[32] Vikramjit Sidhu, Edgar Tretschk, Vladislav Golyanik, Antonio Agudo, and Christian Theobalt. Neural dense non-rigid structure from motion with latent space constraints. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[33] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, pages 1121–1132, 2019. 3

[34] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1386–1395, 2017. 2

[35] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2646–2655, 2018. 2

[36] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. 2

[37] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007. 2, 5

[38] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016. 1

[39] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *European Conference on Computer Vision*, pages 293–309. Springer, 2020. 3

[40] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. *arXiv preprint arXiv:1905.10290*, 2019. 3

[41] Sen Wang, Xinxin Zuo, Chao Du, Runxiao Wang, Jiangbin Zheng, and Ruigang Yang. Dynamic non-rigid objects reconstruction with a single rgb-d sensor. *Sensors*, 18(3):886, 2018. 2, 8

[42] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020. 3