Template-free Articulated Neural Point Clouds for Reposable View Synthesis

Lukas Uzolas Elmar Eisemann Petr Kellnhofer
Delft University of Technology
The Netherlands
{l.uzolas, e.eisemann, p.kellnhofer}@tudelft.nl

Abstract

Dynamic Neural Radiance Fields (NeRFs) achieve remarkable visual quality when synthesizing novel views of time-evolving 3D scenes. However, the common reliance on backward deformation fields makes reanimation of the captured object poses challenging. Moreover, the state of the art dynamic models are often limited by low visual fidelity, long reconstruction time or specificity to narrow application domains. In this paper, we present a novel method utilizing a point-based representation and Linear Blend Skinning (LBS) to jointly learn a Dynamic NeRF and an associated skeletal model from even sparse multi-view video. Our forward-warping approach achieves state-of-the-art visual fidelity when synthesizing novel views and poses while significantly reducing the necessary learning time when compared to existing work. We demonstrate the versatility of our representation on a variety of articulated objects from common datasets and obtain reposable 3D reconstructions without the need of object-specific skeletal templates. The project website can be found at https://lukas.uzolas.com/Articulated-Point-NeRF/.

1 Introduction

Synthesizing novel photo-realistic views of captured 3D scenes is important for many domains including virtual/augmented reality, video games or movie productions. In recent years, Neural Radiance Fields (NeRFs) [1] have proved their remarkable capacity to represent complex view-dependent effects, captured in photographs and videos, sparsely sampled from natural light fields [2]. Follow-up works have extended the scope to dynamic scenes [3–10], facilitating rendering of unseen views at different timestamps. Despite progress in reconstruction quality and speed [11], manipulating learned scenes remains a challenge, but would be a highly desirable feature, as it can enable downstream applications, such as the creation of avatars for virtual presence or 3D assets for games and movies.

The key challenge of reposing a NeRF is inverting the backward-warping function that maps individual observations to a shared canonical representation [3]. It is an inversion of traditional kinematic animation, such as Linear Blend Skinning (LBS) [12], where a canonical shape is forward-warped to a desired pose. Such inverse mapping often requires resolving ambiguous situations as it is difficult to guarantee bijectivity (see Fig. 2). A common remedy are parametric templates, typically built for narrow application domains, such as human heads and bodies [13–16], but they are difficult to generalize. Alternatively, object shapes can be retrieved from videos as ensembles of geometric parts, yet existing techniques provide limited image-synthesis fidelity [17–19]. Our work aims to combine these different lines of work and enable joint learning of the NeRF representation and its pose parameterization from sparse or dense multi-view videos. We aim for time-efficient class-agnostic view synthesis of reposable models with high image-synthesis quality without access to a template or pose annotation, which is a combination not currently covered by existing work (see Table 1).

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

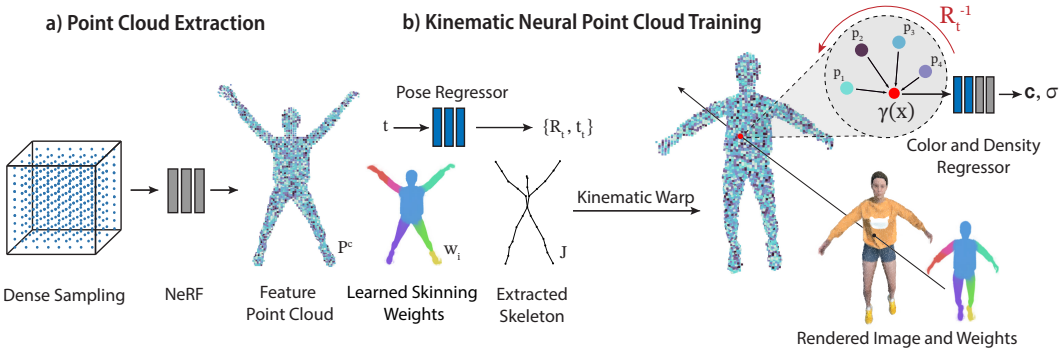


Figure 1: Overview of our method: a) First, we pre-train a NeRF backbone to initialize a feature point cloud P^c for a selected canonical timestamp and to extract an initial skeleton. b) During the main training stage, P^c is forward-warped using LBS consisting of learned time-invariant skinning weights $\hat{w}_i$ and time-dependent pose transformations from an MLP regressor Φ_r. The image is obtained by integration and decoding of features aggregated from points along each camera ray. In summary, we fine-tune the initial neural point features $\mathbf{f_i}$, skinning weights $\hat{w}_i$, joints J, density and color regressor Φ_d and Φ_c of the backbone. We fully train the pose regressor Φ_r and feature point decoder Φ_p from scratch. In test time, we modify the pose transformations to obtain novel poses.

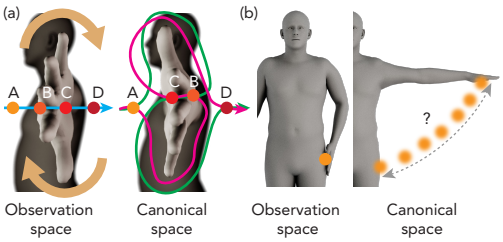


Figure 2: Examples of poses difficult for backward warping. (a) Ill-defined projection from an observation to the canonical space. Both ambiguous solutions (green and magenta) correctly pass through the semantically corresponding surface points B and C. (b) Projection ambiguity for points of contact between two surfaces. Note that in contrast, it is trivial to forward warp the object points from a well-chosen canonical space to any observation space.

Method	Pose	Generic	Fildelity	Training
LASR [17]	No	Yes	Shape	$\leq$ 2 h
ViSER [18]	No	Yes	Shape	2–12 h
BANMO [22]	No	Yes	Low	$\geq$ 12 h
D-NeRF [3]	No	Yes	High	$\geq$ 12 h
TiNeuVox [11]	No	Yes	High	$\leq$ 2 h
CASA [23]	Yes	No	Shape	$\leq$ 2 h
LASSIE [19]	Yes	Yes	Low	$\geq$ 12 h
WIM [20]	Yes	Yes	High	$\geq$ 12 h
Ours	Yes	Yes	High	$\leq$ 2 h

Table 1: A comparison of representative dynamic 3D representations learned from 2D videos. We analyze reposibility, agnosticism to object class, image synthesis fidelity and training time. Papers that demonstrate reposing are shown below the bar. "*Shape*" denotes shape-only reconstruction without texture details.

Recent work by Noguchi et al. [20] addresses a similar problem. However, we exploit the structure-free nature of point-based NeRF representations [21], enabling direct forward warping of the canonical object to any desired pose, while maintaining the capacity and flexibility of NeRF-like rendering to capture highly detailed image features. Together with our automatically extracted and jointly-optimized LBS skeletal pose parameterization, our work allows for faster training and achieves state-of-the-art visual quality in a much shorter time. Finally, we demonstrate how to easily repose learned representations of varied objects by directly editing joint angles of the forward LBS model.

To summarize, we make the following contributions: 1) We propose a novel method for learning articulated neural representations of dynamic objects using forward projection of neural point clouds. 2) We demonstrate state-of-the-art novel view synthesis for a variety of multi-view videos with training times lower than comparable methods. 3) We jointly learn a skeletal model without domain-specific priors or user annotations and demonstrate reposing of the reconstructed 3D objects.

2 Related Work

Neural Radiance Fields and Parameterizations Implicit neural networks have recently emerged as a powerful tool for building differentiable representations of 3D scenes [24–47] and for learning view-dependent *Neural Radiance Fields* (NeRFs) [1, 2, 48–57] enabling photo-realistic novel view synthesis. Their high computational cost has been dramatically reduced by spatial decomposition [58–60] and hybrid representations [61–65]. Recently, neural point clouds have combined expressive neural functions with the flexibility of explicit geometry [21, 66–69]. We also use neural point clouds, but unlike previous work, we jointly learn a forward kinematic model along with the associated soft part segmentation and, thus, convert fast voxel-based representations to reposable NeRFs.

Dynamic Neural Radiance Fields Equivalent to 2D video, dynamic NeRFs capture temporal changes in scenes by encoding each frame separately [70], expanding the radiance field into the temporal domain [7, 8, 71, 72], or time-dependent interpolation in a compact latent representation [6, 9, 10]. Alternatively, a single canonical NeRF can be animated by backward warping from the canonical space to individual time-varying observation spaces [3–6, 9, 11]. However, such warps rely on the bijectivity of the mapping which is difficult to guarantee for all observed poses (see Fig. 2). Forward mapping only requires a single well-posed canonical representation, which we exploit.

Object Reposing Directly reposing high-dimensional deformation fields of dynamic NeRFs is impractical. Instead, parametric templates can sparsely represent prominent deformation modes for faces [13, 14], bodies [15, 16], hands [73], and also non-human objects, such as animals [74]. Together with skeleton-based LBS, they enable articulated neural representations of human heads [69, 75–81] or bodies [82–97], as well as modeling distributions in generative models [98–101]. However, the assumption of piece-wise rigid motion and the availability of a skeletal template restrict the applicable object classes. The former issue has been addressed by physically inspired but computationally expensive deformation models [102]. To remedy the latter issue, the skeletal model can be retrieved from a database [23], adapted from a generic graph [19, 103], fitted as a template to 2D observations [104–108] or fully learned during a 3D shape recovery [20, 109]. An external cage can also be used for re-animation of static NeRFs [110]. Alternatively, a surface embedding can be learned without a skeleton if reposing is not required [17, 18, 22].

Similarly to Noguchi et al. [20], we jointly learn a 3D object representation, a skeletal model, and observed poses. However, we utilize a point-based NeRF representation to benefit from its computational efficiency, ease of geometric transformation, and capacity to learn complex light fields. We share our point-based approach with a contemporaneous work *MovingParts* [111] which, however, relies on inverting the backward flow. We demonstrate the reconstruction of large transformations in the *Robots* dataset [20], not demonstrated in *MovingParts*. Furthermore, our approach does not enforce binary segmentation and, hence, can represent non-rigid part transitions (see Fig. 6 right).

3 Preliminaries

Our method builds upon available NeRF reconstruction methods as a backbone for learning a reposable dynamic representation. In principle, any NeRF-like representation is a suitable initial point for our training. In practice, we use TiNeuVox [11] in all experiments, as it enables fast reconstruction with both sparse and dense multi-view supervision in dynamic scenes. Next, we describe the core concepts of NeRF [1] and TiNeuVox [11] to provide context for our method in Sec. 4.

Time-Aware Neural Voxels NeRF [1] maps a point $\mathbf{p} = (x, y, z)$ and view direction $\mathbf{d} = (\theta, \phi)$ to color $\mathbf{c} = (r, g, b)$ and density σ. The mapping function often takes form of a Multi-Layer-Perceptron (MLP) Φ, such that $(\mathbf{c}, \sigma) = \Phi(\mathbf{p}, \mathbf{d})$. The color of an image pixel $C(\mathbf{r})$ can be obtained via volume rendering along the ray $r(t) = \mathbf{o} + t\mathbf{d}$.

TiNeuVox [11] expands NeRF to dynamic scenes and significantly increases training speed by utilizing an explicit volume representation. To this extent, each point $\mathbf{p}$ at time t in observation space is backward-warped to canonical space as

$$\mathbf{p}^c = \Phi_b(\hat{\mathbf{p}}, \hat{\mathbf{t}}), \text{ where } \hat{\mathbf{p}} = \gamma(\mathbf{p}), \ \hat{\mathbf{t}} = \Phi_t(\gamma(t)), \tag{1}$$

Here, γ is the positional encoding [1], Φ_t a small color-decoding MLP, Φ_b a backward-warping MLP, and $\hat{\mathbf{p}}$ and $\hat{\mathbf{t}}$ the encoded point and time respectively. The canonical point $\mathbf{p}^c$ is used to

sample a feature vector $\mathbf{v}_m \in \mathcal{R}^V$ from the explicit feature volume $\mathbf{V} \in \mathcal{R}^{X \times Y \times Z \times V}$ by means of trilinear interpolation $interp$ with varying stride s_m at each scale: $\mathbf{v}_m = \gamma(interp(\mathbf{p}^\mathbf{c}, \mathbf{V}, s_m))$. The feature vectors across all scales are concatenated as $\mathbf{v} = \mathbf{v}_1 \oplus ... \oplus \mathbf{v}_m \oplus ... \oplus \mathbf{v}_M$, where $\oplus$ is the concatenation operator. Finally, the view-dependent color and density are regressed by additional MLPs Φ_f, Φ_d and Φ_c:

$$\mathbf{f} = \Phi_f(\mathbf{v}, \hat{\mathbf{p}}, \hat{\mathbf{t}}), \text{ where } \sigma = \Phi_d(\mathbf{f}), \ \mathbf{c} = \Phi_c(\mathbf{f}, \mathbf{d}). \tag{2}$$

The volume rendering equation [112] integrates density/color of points $\mathbf{p}_i$ sampled along the ray r:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{ where } T_i = exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)), \tag{3}$$

Here, δ_i is the distance between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The feature volume $\mathbf{V}$ and all MLPs are optimized end-to-end and supervised by the Mean Squared Error (MSE) and the corresponding ground-truth pixel color $C(\mathbf{r})$: $\mathcal{L}_{photo} = ||\hat{C}(\mathbf{r}) - C(\mathbf{r})||_2^2$.

Note that because Φ_f (Eq. 2) is conditioned by the time $\hat{\mathbf{t}}$, space and time are entangled and the feature volume V does not represent a single static canonical shape. Therefore, we cannot directly learn a forward warping function to invert Φ_b as proposed by Chen et al. [102]. In the next section, we show that this is not necessary and we can learn a forward kinematic model directly without relying on specific properties of the backbone, which will allow us to replace it in the future.

4 Method

While dynamic NeRF models such as TiNeuVox [11] can reproduce motion in a dynamic scene, they do not enable reposing. Furthermore, it is impractical to post hoc reparametrize their motion representation due to the ambiguities of backward flow inversion (see Fig. 2).

We instead propose a completely new representation based on a neural point cloud and an explicit kinematic model for forward warping, and we use the backbone only as an initialization for our training procedure (see Fig. 1). First, we extract a feature point cloud from a selected canonical frame of the backbone Sec. 4.1. Second, we describe the underlying kinematic model of our 3D representation and its initialization Sec. 4.2. Consecutively, we introduce how the 3D point cloud is warped from a canonical space to an observation space and rendered Sec. 4.3. Lastly, we describe our losses Sec. 4.4.

4.1 Canonical Feature Point Cloud

We first pre-train a backbone NeRF model (TiNeuVox [11]) using its default parameters. To initialize the feature point cloud, we sample the canonical density function of TiNeuVox $\sigma = \Phi_d(\mathbf{f})$ (Eq. 2) on a uniform coordinate grid and discard empty samples through thresholding. The grid resolution is adaptively chosen, such that $|P| \approx 10000$, similarly to other explicit NeRFs [11, 64, 113]. After thresholding, we obtain points $P = \{\mathbf{p}_i | i = 1, ..., N\}$. Furthermore, we extract a feature vector $\mathbf{f}_i$ for each point $\mathbf{p}_i$ (see Eq. 2).

4.2 Kinematic Model

We forward-warp our feature point cloud from canonical space to observation space using an LBS kinematic model to match the training images. Here, we describe how we initialize, use, and simplify our kinematic model.

Skeleton Initialization We do not rely on a class-specific template to initialize our kinematic skeleton. Instead, we extract an approximate initial skeleton tree by Medial Axis Transform (MAT) and refine it during training.

Let $M = \{\mathbf{p}_m | m = 1, ..., M\}$ be the set of medial axis points extracted by applying a MAT on P^c. We choose the root of our kinematic model $\mathbf{p}_{root}$ as the medial axis point that is closest to all other medial-axis points, i.e., $\mathbf{p}_{root} := \arg\min_i \sum_j ||\mathbf{p}_i - \mathbf{p}_j||_2, \forall \mathbf{p}_i \in M, i \neq j$.

Next, we leverage dense sampling-grid neighborhoods and define a graph G_M connecting neighboring points in M. Points disconnected from $\mathbf{p}_{root}$ are removed. We then use a heuristic to select sparse joints (nodes) as a subset of G_M and define the bones (edges) based on their connectivity. To this extent, we traverse G_M starting from $\mathbf{p}_{root}$ in a breadth-first manner and mark $\mathbf{p}_m$ as a joint $\mathbf{j}_b$ if its geodesic distance from the preceding joint exceeds a threshold $B_{length} = 10$. As a result, we obtain a set of time-invariant canonical joint positions $J = \{\mathbf{j}_b\}$ which we further optimize during training.

While this skeleton is usually over-segmented, we show in our experiments that this does not hamper the training. We propose a pruning strategy to enable easier pose manipulation afterwards (see Fig. 7).

Blend Skinning Weights For each point $\mathbf{p}_i$ we initialize its raw skinning weight vector $\hat{\mathbf{w}}_i$ by an exponential decay function of the distance $dist$ to each bone line b_j such that $\hat{w}_{i,j} = 1/e^{dist(\mathbf{p}_i^c, b_j)}$. Before skinning, we obtain the final blend skinning weight vector $\mathbf{w}_i$ through scaling by a global learnable parameter α and applying a softmax: $w_{i,j} = e^{\hat{w}_{i,j}/\alpha} / \sum_k e^{\hat{w}_{i,k}/\alpha}$. During the training, we optimize the initial $\hat{\mathbf{w}}_i$ as well as α. Accounting for the per-point weights, we define our full canonical feature point cloud as $P^c = \{(\mathbf{p}_i^c, \mathbf{f}_i, \hat{\mathbf{w}}_i) | i = 1, ..., N\}$.

Point Warping We forward-warp the canonical point cloud P^c to an observation space of timestamp t via LBS [12]. The local transformation matrix $\hat{T}_b^t$ of each bone b is defined by a rotation R_b^t around its parent joint $\mathbf{j}_b$. Consequently, each point $\mathbf{p}_i^c$ is transformed by a linear combination of bone transformations as:

$$\mathbf{p}_i^t = \bar{T}_i^t p_i^c = \sum_{b=1}^{|B|} w_{i,b} T_b^t \mathbf{p}_i^c, \text{ with } T_b^t = T_{p_b}^t \hat{T}_b^t \text{ and } \hat{T}_b^t = \begin{bmatrix} R_b^t & \mathbf{j}_b + R_b^t \mathbf{j}_b^{-1} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{4}$$

where T_b^t is defined recursively by its parent bone p_b. $T_{p_b}^t$ of the skeleton root is identity.

$B = \{(\mathbf{j}_p, \mathbf{j}_c) \mid \forall (p, c) \in H\}$

$H := graph$

We express rotations $R_b^t \in SO(3)$ using the Rodrigues' formula, where $\hat{\mathbf{r}} = \mathbf{r}/||\mathbf{r}||$ is the axis of rotation. However, we learn the rotation angle θ directly as an additional parameters because we find it leads to a better pose regression than using $\theta = ||\mathbf{r}||$. The time-dependent rotations $\mathbf{r}_b, \theta_b$ for each bone b are regressed by an MLP: $\Phi_r(\gamma(t)) = \mathbf{r}_1^t, \theta_1^t, ..., \mathbf{r}_b^t, \theta_b^t, ..., \mathbf{r}_B^t, \theta_B, \mathbf{r}_r^t, \theta_r^t, \mathbf{t}_r^t$, where $\mathbf{r}_r^t$, θ_r^t and $\mathbf{t}_r^t$ are the time-dependent root rotation and translation.

Skeleton simplification The initial skeleton's over-segmentation does not hamper pose reconstruction during training, but we optionally simplify the skeleton after training to ease pose editing (see Fig. 7). We prune or merge bones based on the size of rotation angles θ_b^t produced by the transformation regressor Φ_r. Joints, which do not exhibit a rotational change from the rest pose above the threshold of t_r deg in more than 5% of the observed timestamps, are marked static. We then merge the bones of such joints and their corresponding weights $\hat{\mathbf{w}}$. See the supplement for details.

4.3 Dynamic Point Rendering

We adopt the point cloud rendering from [21] but extend it to explicitly model rotational invariance of the radiance field. For each sampling point x, we find up to $N = 8$ nearest observation feature points $\mathbf{p}_i^t$ within a radius of 0.01 and roto-translate them into their canonical frames. This enables the feature embedding MLP Φ_p to learn spatial relations in a pose-invariant coordinate frame as:

$$\mathbf{f}_{i,x}^t = \Phi_p(\mathbf{f}_i, x_{\mathbf{p}_i}^t), \text{ with } x_{\mathbf{p}_i}^t = \gamma(R_i^{t^{-1}}(x - \mathbf{p}_i^t)), \tag{5}$$

where, R_i^t is the rotation component $\bar{T}_i^t$ (Eq. 4) for point $\mathbf{p}_i^t$ and $\gamma(.)$ is the positional encoding [1]. The neighboring embeddings $\mathbf{f}_{i,x}^t$ are aggregated by inverse distance weighting, which produces the final feature input for Φ_d and Φ_c (see Eq. 2) and the consequent volume rendering (Eq. 3):

$$\mathbf{f}_x^t = \sum_i^N \frac{d_i}{\sum_j^{|B|} d_j} f_{i,x}^t, \text{with } d_i = ||p_i^t - x||^{-1}; \sigma = \Phi_d(\mathbf{f}_x^t);\ \mathbf{c} = \Phi_c(\mathbf{f}_x^t, \mathbf{d}). \tag{6}$$

Figure 3: Qualitative comparison displaying two held-out views-frames of scenes from the *Robots* rendered by WIM [20] and our method after 2 and 10 hours of training, and the PSNR scores.

4.4 Losses

Next to the photometric loss $\mathcal{L}_{photo}$ (Sec. 3), we utilize a 2D chamfer loss to penalize the difference between the point cloud projected into a training view $I(P^t)$ and the corresponding 2D ground truth object mask M^t: $\mathcal{L}_{mask} := \mathcal{L}_{champ}(I(P^t), M^t)$. The chamfer loss is defined as:

$$\mathcal{L}_{champ}(P^1, P^2) = \frac{1}{|P^1|}\sum_{i}^{|P^1|} \min_j ||\mathbf{p}_i^1 - \mathbf{p}_j^2||_2^2 + \frac{1}{|P^2|}\sum_{j}^{|P^2|} \min_i ||\mathbf{p}_i^1 - \mathbf{p}_j^2||_2^2. \tag{7}$$

To prevent the skeleton from diverging too much from the medial axis M, we further minimize the chamfer loss between M and the joints J (see Sec. 4.2): $\mathcal{L}_{skel} := \mathcal{L}_{champ}(M, J)$. In addition, we minimize the transformation angles and the root translation: $\mathcal{L}_{tranf} = (\sum_i |\theta_i^t|) + |\mathbf{t}_r^t|$. Local rigidity is further enforced upon points after deformation via as-rigid-as-possible regularization:

$$\mathcal{L}_{arap} = \sum_{i}^{|P^t|} \sum_{j}^{N(p_i)} |||p_i^c - p_j^c||_2^2 - ||p_i^t - p_j^t||_2^2|. \tag{8}$$

Lastly, we apply two regularizers on the blend skinning weights. To encourage smoothness, we penalize divergence of skinning weights in the rendering neighborhood N: $\mathcal{L}_{smooth} = \sum_i^{|P^t|} \sum_{j \in N(p_i)} |w_i - w_j|$, and, to encourage sparsity, we apply:

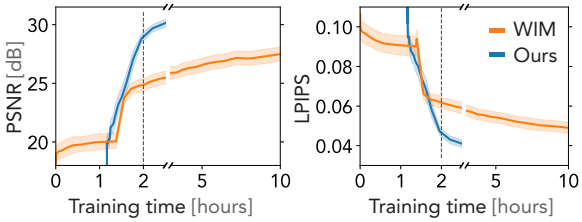
$$\mathcal{L}_{sparse} = -\sum_{i}^{|P^c|}\sum_{j}^{B} w_{i,j}\log(w_{i,j}) + (1 - w_{i,j})\log(1 - w_{i,j}). \tag{9}$$

In total, our training loss is $\mathcal{L} = \omega_0\mathcal{L}_{photo} + \omega_1\mathcal{L}_{mask} + \omega_2\mathcal{L}_{skel} + \omega_3\mathcal{L}_{tranf} + \omega_4\mathcal{L}_{smooth} + \omega_5\mathcal{L}_{sparse} + \omega_6\mathcal{L}_{ARAP}$ where $\omega = \{200, 0.02, 1, 0.1, 10, 0.2, 0.005\}$ in our experiments.

5 Experiments

Here, we evaluate our work, which obtains state-of-the-art view-synthesis quality with a lower training cost than other articulated methods. We also demonstrate class-agnostic reposing capability and we evaluate contribution of method components in an ablation study. Video examples can be seen on the project website.

Datasets We chose three multi-view video datasets that are commonly used for the evaluation of dynamic multi-view synthesis methods and pose articulation. First, the *Robots* dataset [20] features multi-view synthetic videos of 8 topologically varied robots, making it well suited for testing pose articulation performance (see Fig. 4). We use 18 views for training and 2 for evaluation. Second, the



Time	Method	PSNR↑	SSIM↑	LPIPS↓
2 h	WIM [20]	25.05	0.952	0.059
	Ours	**28.81**	**0.968**	**0.047**
10 h	WIM [20]	27.61	0.964	0.046
	Ours	**30.19**	**0.973**	**0.041**

Figure 4: Quality of unseen view synthesis during training with 95% confidence intervals in the *Robots* dataset [20]. The initial plateau of WIM [20] matches the 10k warm-up steps used by the authors before training with all data. Our onset time corresponds to the 70 minutes required for pre-training of the backbone. Training of our method was terminated after 2.5 hours.

Method	Reposable	PSNR↑	SSIM↑	LPIPS↓	Training Time
D-NeRF [3]	No	30.50	0.95	0.07	20 hours
TiNeuVox-B [11]	No	32.67	0.97	0.04	28 mins
Hexplane [72]	No	31.04	0.97	0.04	11.5 mins
K-Plane hybrid [71]	No	31.61	0.97	-	52min
WIM* [20]	**Yes**	23.81	0.91	0.10	11 hours
Ours*	**Yes**	29.10	0.94	0.06	2.5 hours

Table 2: Quality of unseen view synthesis for the *Blender* dataset [3]. Results of D-NeRF and TiNeuVox-B reprinted from Fang et al. [11]. *Without the "Bouncing balls" scene.

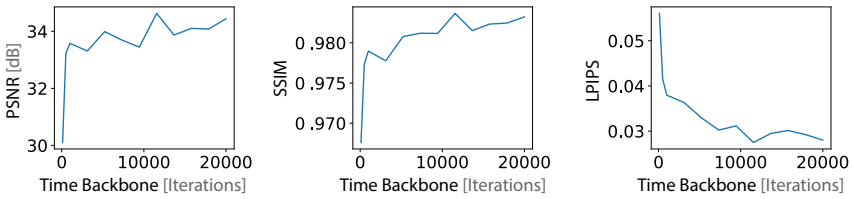


Figure 5: Effect of the backbone initialization pre-training steps on the final result of our method when trained on the *Jumping Jacks* scene from the *Blender* dataset. Our final choice of 20k iterations corresponds to approximately 70 minutes of real time.

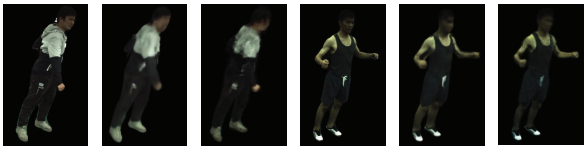
Blender dataset [3] is a sparse multi-view synthetic dataset with 5 humanoid and 2 other articulated objects[1]. Its visual quality benchmarks are used to test the fidelity of image detail reconstruction (see Fig. 6 right). We use the original training-test split. Third, *ZJU-MoCap* dataset [82] is a common test suite for dynamic human reconstruction and we evaluate 5 of its multi-view sequences with the same 6 training views as used in *Watch-It-Move* [20]. Finally, we evaluate image quality using peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [114], and learned perceptual image patch similarity (LPIPS) [115] image metrics.

Implementation details We pre-train the TiNeuVox [11] backbone using the authors' implementation and an additional distortion loss [116], as implemented in [117]. Our method is implemented in Pytorch and we train each scene using the Adam optimizer for 160k (*Blender* and *Robots*) or 320k (*ZJU-MoCap*) iterations with a batch size of 8192 rays, sampled randomly from multiple views and a single timestamp. We choose the canonical timestamp by visual inspection, and gradually increase the number of observed timestamps during training. We adjust ray sampling and scheduling for the *ZJU-MoCap* dataset (see the Supplement). All experiments were done on a single Nvidia GPU RTX 3090Ti. See the supplementary materials for details. For more details, see the Appendix.

Baselines We compare our method to state-of-the-art non-articulated and articulated methods for high-fidelity multi-view video synthesis (see Table 1 for an overview). *D-NeRF* [3] extends NeRF to the temporal domain by backward warping a static canonical NeRF. *TiNeuVox* [11] improves

[1]We do not use the multi-component *Bouncing balls* scene.

Figure 6: While our model well reproduces details and non-rigidity of the *Dinosaur* (right), it can fail for complex motions combining long chains of rotations with texture cue ambiguity (left).



Method	PSNR↑	SSIM↑	LPIPS↓
WIM [20]	31.08	0.963	**0.053**
Ours	29.60	0.958	0.063
Ours^{pose}	**32.05**	**0.967**	0.056
Ours^{SMPL}	32.01	0.967	0.056

Figure 10: Comparison in the *ZJU-MoCap* dataset. Ours: Our full method. Ours^{pose}: Ours with an additional pose-conditioned feature embedding network Φ_p. Ours^{SMPL}: Ours^{pose} with ground truth SMPL [15] skeleton.

performance of *D-NeRF* using voxel grids. *Hexplane* [72] and *K-Planes* [71] decompose to the space-time volume to several hyper-planes. Finally, *Watch-It-Move* [20] (WIM) jointly learns a surface representation and LBS model for articulation. Note, that because we aim at time-efficient learning, training of WIM was stopped after 11 hours (i.e., 80k of the original 400k optimization steps).

Novel view synthesis We provide results for the *Robots* view synthesis without the skeleton simplification; quantitatively in Fig. 4 and qualitatively in Fig. 3. More visual examples are available in the supplement. After the initial pre-training, our method quickly surpasses the image quality of WIM [20]. The mean image scores obtained for our method after 2 hours of training (incl. pre-training) are higher than those that WIM achieves after 10 hours. We attribute this to NeRF's capability to approximate even complex shapes, which are difficult to fit using the signed distance function utilized by WIM. Nevertheless, for visually simple objects with long nested pose transformation chains, this capacity might encourage false explanations of the articulation and cause artifacts, as visible in Fig. 6 left. However, this is not a common issue. We illustrate it in the *Blender*, where WIM struggles to represent fine details (see Fig. 6 right), while our method achieves image scores close to those of non-reposable baselines (see Table 2 and the supplement).

Furthermore, Fig. 5 shows that our image quality is not highly dependent on the pre-training phase. We observe that mere 100 iterations of *TiNeuVox* pre-training provide an initialization sufficient for achieving PSNR scores over 30 dB by our method. However, while fine geometric details are still recovered, the coarse initial shape limits the skeleton complexity and, therefore, we opt for 20k pre-training steps in all other experiments.

ZJU-MoCap In Fig. 10, we compare our method to WIM in the *ZJU-MoCap* dataset. We observe that both methods are able to recover the 3D shape and the skeletal motion despite the difficulty of an accurate fine texture detail reconstruction. This can partly be attributed to imperfections in camera calibrations (see Supplement F of [96]) and partly by soft deformations of clothes which are modeled neither method. Notably, with a small modification our method learns to partially compensate for this. In Ours^{pose}, we condition the feature embedding network Φ_p by the skeletal poses jointly learned from scratch by our method. This improves the image quality by modeling the residual deformations. See the Supplement for details. Finally, Ours^{SMPL} shows that replacing our skeleton in Ours^{pose} with a ground truth from an annotated SMPL template [15] does not affect the performance. This validates our skeleton initialization based on Medial Axis Transform.

Reposing In Fig. 7, we visualize the learned blend-skinning weights $\hat{\mathbf{w}}_i$ and skeletons with and without the skeleton simplification. The algorithm is able to substantially reduce the skeleton

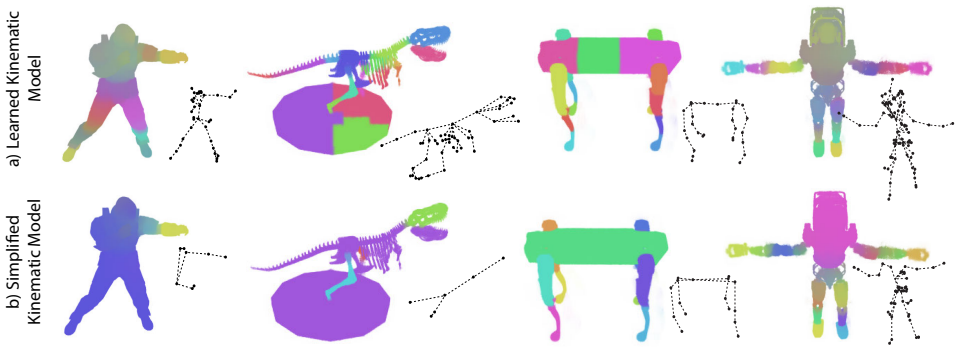


Figure 7: Learned LBS weights and skeleton: a) After training. b) After additional post-processing (weight merging and skeleton pruning, see Sec. 4.2).

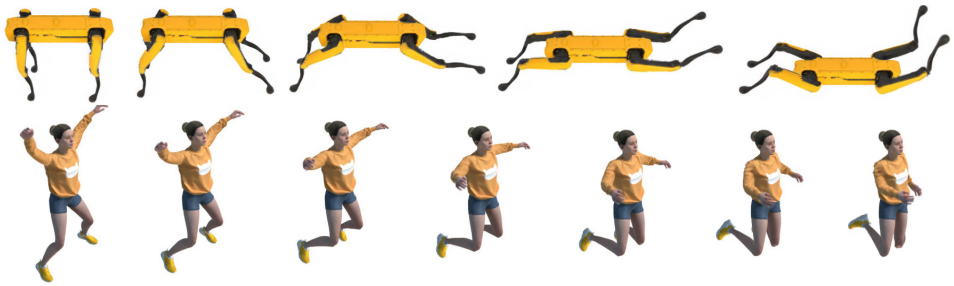


Figure 8: Reposing using the simplified skeleton. Interpolation from canonical to novel pose.

Figure 9: Ablation examples. a) $\mathcal{L}_{ARAP}$ enforces rigidity after pruning (see upper pipes), b) $\mathcal{L}_{smooth}$ results in better part-segmentation, c) $\mathcal{L}_{skel}$ enforces the joints to stay inside the shape.

complexity, while largely preserving semantic articulations observed in the training data. However, it is not able to remove all unnecessary skeletal branches when complex geometry is present (see Fig. 7 right). In Fig. 8, we show that such post-processed skeletons allow for animating of novel poses by smoothly interpolating between user-defined poses. More animations can be found in the Supplement.

Loss ablation Our experiments suggest the regularization does not improve the image quality, but it improves the quality of the kinematic model, which is important for our main goal of reposing (see Fig. 9). Specifically, $\mathcal{L}_{ARAP}$ avoids non-rigid distortions (a), $\mathcal{L}_{smooth}$ leads to a more semantically meaningful part segmentation (b), $\mathcal{L}_{skel}$ encourages functional placement of the skeleton joints inside the object volume even for thin parts (c), and $\mathcal{L}_{tranf}$ leads to a reduction of necessary object parts after simplification (62 components are removed instead of 49 for *Atlas*). More details are provided in the Supplement.

Backbone The *TiNeuVox* backbone allows us to transfer parameters learned during the pre-training and initialize the features $\mathbf{f}_i$ and the regressors $\mathbf{\Phi}_d$ and $\mathbf{\Phi}_c$. Interestingly, our experiment shows that such off-the-shelf features $\mathbf{f}_i$ often do not need any further fine-tuning (see Table 4). Despite this, our method is agnostic to the backbone choice by design and the end-to-end training procedure of the entire model is essential for this. We demonstrate it by training with a random initialization. This way, only the point positions $\mathbf{p}_i^c$ obtainable by any 3D reconstruction method are needed. Table 3

	PSNR↑	SSIM↑	LPIPS↓
Full Method	30.19	0.973	0.041
Random Init.	29.97	0.971	0.045

Table 3: Results with a *TiNeuVox* initialization and with a random initialization of the features $\mathbf{f}_i$ and the regressors $\mathbf{\Phi}_d$ and $\mathbf{\Phi}_c$ in the *Robots* dataset.

	PSNR↑	SSIM↑	LPIPS↓
Full Method	29.10	0.94	0.06
No Fine-tuning	29.16	0.94	0.05

Table 4: Results with and without fine-tuning of the feature points $\mathbf{f}_i$ in the *Blender* dataset.

shows that this leads to only a negligible drop in reconstruction quality in the *Robots* dataset. Here, the weight of $\mathcal{L}_{skel}$ ($w_2 = 1$) was adjusted to prevent drift of the skeleton.

6 Discussion

Limitations and Future Work We demonstrate fast learning of articulated NeRFs for state-of-the-art view synthesis and straightforward skeletal reposing for objects with piece-wise rigid pose transformations. While the LBS allows for fitting partially non-rigid deformations (see Fig. 6 right), representing fully non-rigid, topologically varying, and multi-component objects would benefit from a higher-dimensional parameterization. Chen et al. [102] aim in that direction, but an intuitive and cost-effective reposing still favors skeleton-based techniques such as ours.

The performance of our method depends on the quality of the backbone reconstruction. While TiNeuVox [11] supports reconstruction from even sparse data, a different backbone could offer a more robust starting geometry and improve the skeleton initialization. Although our approach works well even for highly structured shapes and thin structures (see Fig. 6 right), we observe incorrect joint placement for objects with long chains of highly nonlinear geometrical transformations (see Fig. 6 left). Moreover, our approach successfully reduces the number of extraneous bones but it is not able to completely eliminate all superfluous skeletal elements (see Fig. 7). Our method is restricted to the kinematic motion space exhibited in the training sequences. While extrapolation is possible to some extent, the proposed method is not able to generalize to arbitrary unseen poses. Finally, we focus on image synthesis for user-defined poses rather than a direct fitting of unseen skeletal poses from images or motion capture. An inverse fitting of skeletal poses to novel 2D observations or transfer of poses from one object to another remain opportunities for future research.

Conclusion We presented a method for fast learning of articulated models for high-quality novel view synthesis of captured 3D objects. Our forward-warped neural point clouds avoid the pitfalls of backward warping and elegantly integrate a skeleton-based LBS. As a result, our method merges straightforward reposing even of strongly animated inputs, as present in the *Robots* dataset, with high visual fidelity of NeRF rendering. Our work is a significant step towards low-cost acquisition of animatable 3D objects for games, movies, and education.

Ethical Considerations Our method renders novel views and poses of 3D objects including human bodies. However, we do not focus on this class and we note that many human-specific methods exist (see Sec. 2). Nevertheless, we acknowledge that our method could potentially be used to produce fake images of people and that research is needed to understand the risks and their mitigation.

References

[1] 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] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022.

[3] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[4] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021.

[5] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

[6] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[7] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021.

[8] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *CVPR*, 2021.

[9] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph. (SIGGRAPH Asia)*, 40(6), 2021.

[10] Tianye Li, Mira Slavcheva, Michael Zollhöfer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5521–5531, June 2022.

[11] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[12] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, 2000.

[13] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *ACM SIGGRAPH*, page 187–194, 1999.

[14] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Trans. Graph. (SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017.

[15] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (SIGGRAPH Asia)*, 34(6):248:1–248:16, 2015.

[16] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, pages 10975–10985, 2019.

[17] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15980–15989, 2021.

[18] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Ce Liu, and Deva Ramanan. Viser: Video-specific surface embeddings for articulated 3d shape reconstruction. *Advances in Neural Information Processing Systems*, 34:19326–19338, 2021.

[19] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *arXiv preprint arXiv:2207.03434*, 2022.

[20] Atsuhiro Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3687, 2022.

[21] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022.

[22] Gengshan Yang, Minh Vo, Neverova Natalia, Deva Ramanan, Vedaldi Andrea, and Joo Hanbyul. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022.

[23] Yuefan Wu*, Zeyuan Chen*, Shaowei Liu, Zhongzheng Ren, and Shenlong Wang. CASA: Category-agnostic skeletal animal reconstruction. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[24] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 2018.

[25] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] Lars 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)*, 2019.

[27] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[28] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[29] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[30] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning (ICML)*, 2020.

[31] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3D scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[32] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation. *arXiv preprint arXiv:2009.09808*, 2020.

[33] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In *European Conference on Computer Vision (ECCV)*, 2020.

[34] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[35] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[37] Julien N.P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. ACORN: Adaptive coordinate networks for neural representation. *ACM Transactions on Graphics (SIGGRAPH)*, 2021.

[38] Petr Kellnhofer, Lars Jebe, Andrew Jones, Ryan Spicer, Kari Pulli, and Gordon Wetzstein. Neural lumigraph rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[41] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[42] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. DeepVoxels: Learning persistent 3D feature embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[43] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[44] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3D supervision. *arXiv preprint arXiv:1911.00767*, 2019.

[45] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (SIGGRAPH)*, 2019.

[46] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. DIST: Rendering deep implicit signed distance function with differentiable sphere tracing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[48] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[49] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural reflectance and visibility fields for relighting and view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[50] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[51] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021.

[52] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[53] David B Lindell, Julien NP Martel, and Gordon Wetzstein. AutoInt: Automatic integration for fast neural volume rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[54] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. *arXiv preprint arXiv:2104.00670*, 2021.

[55] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[56] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021.

[57] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. *Eurographics Association*, 2020.

[58] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[59] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021.

[60] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022.

[61] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[62] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[63] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv:2201.05989*, January 2022.

[64] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[65] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[66] Julian Ost, Issam Laradji, Alejandro Newell, Yuval Bahat, and Felix Heide. Neural point light fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18419–18429, 2022.

[67] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: Particle based encoding for online neural radiance fields in dynamic scenes. *arXiv preprint arXiv:2211.04041*, 2022.

[68] Ruofan Liang, Jiahao Zhang, Haoda Li, Chen Yang, and Nandita Vijaykumar. Spidr: Sdf-based neural point fields for illumination and deformation. *arXiv preprint arXiv:2210.08398*, 2022.

[69] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J Black, and Otmar Hilliges. Pointavatar: Deformable point-based head avatars from videos. *arXiv preprint arXiv:2212.08377*, 2022.

[70] Shengze Wang, Alexey Supikov, Joshua Ratcliff, Henry Fuchs, and Ronald Azuma. Inv: Towards streaming incremental neural videos. *arXiv preprint arXiv:2302.01532*, 2023.

[71] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[72] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.

[73] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Trans. Graph. (SIGGRAPH Asia)*, 36(6), November 2017.

[74] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *CVPR*, pages 6365–6373, 2017.

[75] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *CVPR*, 2021.

[76] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhofer. Learning compositional radiance fields of dynamic human heads. In *CVPR*, pages 5704–5713, 2021.

[77] Yudong Guo, Keyu Chen, Sen Liang, Yongjin Liu, Hujun Bao, and Juyong Zhang. Ad-nerf: Audio driven neural radiance fields for talking head synthesis. In *ICCV*, 2021.

[78] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M Avatar: Implicit morphable head avatars from videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[79] Kacper Kania, Kwang Moo Yi, Marek Kowalski, Tomasz Trzciński, and Andrea Tagliasacchi. CoNeRF: Controllable Neural Radiance Fields. In *CVPR*, 2022.

[80] Yiyu Zhuang, Hao Zhu, Xusen Sun, and Xun Cao. Mofanerf: Morphable facial neural radiance field. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pages 268–285. Springer, 2022.

[81] ShahRukh Athar, Zexiang Xu, Kalyan Sunkavalli, Eli Shechtman, and Zhixin Shu. Rignerf: Fully controllable neural 3d portraits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20364–20373, 2022.

[82] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *CVPR*, 2021.

[83] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021.

[84] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In *NeurIPS*, 2021.

[85] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *NeurIPS*, 2021.

[86] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *ICCV*, 2021.

[87] Youngjoong Kwon, Dahun Kim, Duygu Ceylan, and Henry Fuchs. Neural human performer: Learning generalizable radiance fields for human performance rendering. In *NeurIPS*, 2021.

[88] Haotong Lin, Sida Peng, Zhen Xu, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields with learned depth-guided sampling. In *arXiv*, 2021.

[89] Tao Hu, Tao Yu, Zerong Zheng, He Zhang, Yebin Liu, and Matthias Zwicker. Hvtr: Hybrid volumetric-textural rendering for human avatars. In *arXiv*, 2021.

[90] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Trans. Graph.(ACM SIGGRAPH Asia)*, 2021.

[91] Tianhan Xu, Yasuhiro Fujita, and Eiichi Matsumoto. Surface-aligned neural radiance fields for controllable 3d human synthesis. In *CVPR*, 2022.

[92] Fuqiang Zhao, Wei Yang, Jiakai Zhang, Pei Lin, Yingliang Zhang, Jingyi Yu, and Lan Xu. Humannerf: Efficiently generated human radiance field from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7743–7753, 2022.

[93] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *CVPR*, 2022.

[94] Boyi Jiang, Yang Hong, Hujun Bao, and Juyong Zhang. Selfrecon: Self reconstruction your digital avatar from monocular video. In *CVPR*, 2022.

[95] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11594–11604, 2021.

[96] Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhöfer, Jürgen Gall, Angjoo Kanazawa, and Christoph Lassner. Tava: Template-free animatable volumetric actors. In *European Conference on Computer Vision*, pages 419–436. Springer, 2022.

[97] Yihao Zhi, Shenhan Qian, Xinhao Yan, and Shenghua Gao. Dual-space nerf: Learning animatable avatars and scene lighting in separate spaces. In *2022 International Conference on 3D Vision (3DV)*, pages 1–10. IEEE, 2022.

[98] Yang Hong, Bo Peng, Haiyao Xiao, Ligang Liu, and Juyong Zhang. Headnerf: A real-time nerf-based parametric head model. In *CVPR*, 2022.

[99] Alexander W Bergman, P Kellnhofer, Wang Yifan, Eric R Chan, David B Lindell, and Gordon Wetzstein. Generative neural articulated radiance fields. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022): NeurIPS Proceedings*. 2022.

[100] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Unsupervised learning of efficient geometry-aware neural articulated representations. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*, pages 597–614. Springer, 2022.

[101] Suyi Jiang, Haoran Jiang, Ziyu Wang, Haimin Luo, Wenzheng Chen, and Lan Xu. Humangen: Generating human radiance fields with explicit priors. *arXiv preprint arXiv:2212.05321*, 2022.

[102] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837, 2022.

[103] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. *arXiv preprint arXiv:2211.12497*, 2022.

[104] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7122–7131, 2018.

[105] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2252–2261, 2019.

[106] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael J Black. Learning to regress 3d face shape and expression from an image without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7763–7772, 2019.

[107] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[108] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020.

[109] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Hi-lassie: High-fidelity articulated shape and skeleton discovery from sparse image ensemble. *arXiv preprint arXiv:2212.11042*, 2022.

[110] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 159–175. Springer, 2022.

[111] Kaizhi Yang, Xiaoshuai Zhang, Zhiao Huang, Xuejin Chen, Zexiang Xu, and Hao Su. Movingparts: Motion-based 3d part discovery in dynamic radiance field. *arXiv preprint arXiv:2303.05703*, 2023.

[112] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[113] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022.

[114] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[115] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[116] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[117] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *arXiv preprint arXiv:2206.05085*, 2022.

[118] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023.

[119] Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. *Advances in Neural Information Processing Systems*, 33, 2020.

[120] Ta-Chih Lee, Rangasami L Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: graphical models and image processing*, 56(6):462–478, 1994.

[121] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2: e453, 2014.

Supplement

A Extra Results *Blender* dataset

Table 5, and Fig. 11 show the quantitative and qualitative per-scene results respectively in the *Blender* dataset [3]. While the non-reposable methods often achieve excellent results, our method is a clear improvement with respect to the other reposable method, WIM [20], while simultaneously reducing training time[2].

	Jumping Jacks			**Mutant**			**Hook**			**T-Rex**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
D-Nerf [3]	32.80	0.98	0.03	<u>31.29</u>	0.97	0.02	29.25	0.96	0.11	31.75	0.97	0.03
TiNeuVox [11]	34.23	0.98	0.03	**33.61**	0.98	0.03	**31.45**	0.97	0.05	<u>32.70</u>	0.98	0.03
HexPlane [72]	31.65	0.97	0.04	33.79	0.98	0.03	28.71	0.96	0.05	30.67	0.98	0.03
Tensor4D [118]	<u>34.43</u>	0.98	0.03	×	×	×	×	×	×	×	×	×
WIM [20]	29.77	0.97	0.04	25.80	0.95	0.06	25.33	0.94	0.06	26.19	0.94	0.08
Ours	**34.50**	0.98	0.03	28.56	0.96	0.03	<u>30.24</u>	0.97	0.05	**32.85**	0.98	0.02
	Stand Up			**Hell Warrior**			**Lego**			**Bouncing Ball**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
D-Nerf[3]	<u>32.79</u>	0.98	0.02	25.02	0.95	0.06	<u>21.64</u>	0.83	0.16	38.93	0.98	0.10
TiNeuVox [11]	**35.43**	0.99	0.02	**28.17**	0.97	0.07	**25.02**	0.92	0.07	**40.73**	0.99	0.04
HexPlane [72]	34.36	0.98	0.02	24.24	0.94	0.07	25.22	0.94	0.04	39.69	0.99	0.03
Tensor4D [118]	36.32	0.98	0.02	×	×	×	26.71	0.95	0.003	×	×	×
WIM [20]	27.46	0.96	0.04	16.71	0.87	0.14	15.41	0.73	0.25	×	×	×
Ours	31.93	0.97	0.02	<u>27.53</u>	0.96	0.06	17.91	0.76	0.14	×	×	×

Table 5: Quantitative Results Per-Scene on *Blender* dataset. We only highlight best and second-best values for PSNR as the precision reported of SSIM and LPIPS in [11] is not enough for fair comparison in many cases.

B Extra Results *Robots* dataset

Table 6, and Fig. 12 show the quantitative and qualitative per-scene results respectively for the *Robots* dataset [20]. Consistent with the findings in the main paper, we see that WIM produces overly smooth results while our method can capture more details in a shorter amount of training time. However, it struggles to accurately recover the true poses for long kinematic chains (Table 6, *Iiwa* and *Pandas*).

[2]Training times are shown in the main paper.

	Atlas			**Baxter**			**Cassie**			**Iiwa**		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
WIM[20]	23.99	0.94	0.07	22.70	0.95	0.06	30.20	0.97	0.04	**31.58**	**0.98**	**0.03**
Ours	**28.71**	**0.97**	**0.04**	**28.60**	**0.97**	**0.04**	**31.84**	**0.98**	**0.04**	29.74	0.98	0.04
	Nao			**Pandas**			**Spot**					
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR ↑	SSIM ↑	LPIPS↓	PSNR ↑	SSIM↑	LPIPS↓			
WIM[20]	26.85	0.95	0.06	**32.31**	**0.98**	**0.03**	26.30	0.97	0.04			
Ours	**29.50**	**0.96**	**0.05**	30.56	0.97	0.05	**32.40**	**0.98**	**0.02**			

Table 6: Quantitative Results Per-Scene in the *Robots* dataset.

Figure 11: Additional qualitative per-scene results of our method for the *Blender* dataset. We compare WIM [100], with our method and the ground truth. The displayed results correspond to the final state of the training procedures as described in the paper.

Figure 12: Additional qualitative per-scene results of our method in the *Robots* dataset. We compare WIM [100], with our method and the ground truth. The displayed results correspond to the final state of the training procedures as described in the paper.

C Implementation Details

C.1 Our method

For training, we utilize PyTorch and the Adam optimizer. We pre-train TiNeuVox for 20k iterations on the synthetic *Blender* data set, and 40k iterations on the WIM data set, and add distortion loss regularization [116] as implemented in [117]. All hyperparameters are adopted from [11].

The neural point clouds are trained for 160k iterations. In each iteration, we sample 8192 rays randomly from multiple views at time step t. On each ray, we sample multiple points p_i as a function of the voxel size of the pre-trained NeRF volume, as specified in [11]. For each sampling point, we query a maximum of 8 neighborhood points $N(p_i)$ within a radius of 0.01 using the KeOps framework [119].

For the kinematic skeleton, we use a bone length of $B_{length} = 10$ for all experiments and a point cloud density threshold of 0.05. We apply the MAT on the binary volume that we retrieve after thresholding the density volume. Assuming a single object, we remove small holes from the volume and extract the medial axis on the biggest blob via the method proposed in [120] and use the scikit-image library [121].

For the mask loss, we project the warped point cloud to five views for the *Robots* dataset and to a single view for *Blender*, as only a single view per timestamp is available. Furthermore, we subsample the point cloud and ground truth mask to $3\,000$ points.

We fine-tune neural point features $\mathbf{f_i}$, skinning weights $\hat{\mathbf{w}}_i$, joints J, density and color regressor Φ_d and Φ_c and train the pose regressor Φ_r and feature point decoder Φ_p from scratch. The MLP of Φ_r has 4 layers of size 128, except for the first layer which has a size of $128 + dim(\gamma(\mathbf{x})) = 191$, where γ is the positional encoding. For all MLPs and the weights, we use a learning rate of 0.0001, except for Φ_r which uses a learning rate of 0.001. We further set the learning rate of α and J to 0.00001. We further utilize learning decay, as in [11, 64] which decays the learning rate by 0.1 after $80\,000$ iterations. We train on a single Nvidia GPU RTX 3090.

C.2 Validation

We used the author's code[3] to reproduce the results of Watch-It-Move [20].

For the *Robots* datasets we adapted the author's configuration files. First, we modified the selection of training and testing views to match the split used in our experiments. That way we held out images from the 10th and 20th camera for testing and provided the remaining 18 cameras for training. Second, to enable early pose fitting and a meaningful computation of progressive learning metrics (see Fig. 4 in the paper), we disabled the gradual scheduler of the training frames. We kept the author's initialization phase with only the first 10 timestamps accessible for the first 10 000 iterations but we provided the entire training set afterwards. This corresponds to the moment past the 1-hour mark in the paper Fig. 4 where the metrics start to improve as the model gets a chance to fit poses of the entire sequence. Without this modification, the network would need another 70 000 steps to access this information. Note that we have not observed a meaningful difference in the quality of the final trained model as a result of this modification. Finally, we reduced the batch size from 16 to 8 views to fit into the 24 GB VRAM of our Nvidia RTX 3090 GPU.

We applied the same training strategy and settings when training with the *Blender* dataset and used the author's pre-trained snapshots for the *ZJU-MoCap* dataset.

D Details of Skeleton Simplification

The skeleton simplification is an optional post-processing step that allows to simplify our kinematic model (see Sec. 4.2 in the paper). While our model supports reposing without this step, we suggest that a smaller number of skeletal parameters makes the process easier for the user/animator.

A key part of this procedure is selection of skeleton joints that are redundant and can be removed. To this goal, we identify *static joints* as those that do not exhibit a rotational deviation with respect to the

[3] https://github.com/NVlabs/watch-it-move

rest pose and relative to its parent joint above a specified threshold in more than 5% of the observed timestamps. Given the low computational cost of this procedure, a user can quickly experiment with ideal selection of this threshold after the training procedure is completed.

After the static joints are identified, they can be removed and the skinning weights corresponding to their children and parent bones can be merged. Here, the parent bone refers to the skeletal tree edge starting in the given joint and pointing towards the root while the children bones are the edges in the leaf direction. The merge only happens for two specific configurations: First, for a static joint, we merge the skinning weights of the children with their parent. Second, we merge the weights of sibling bones if their motion does not differ based on the 5%-heuristic.

To simplify the kinematic skeleton, we further prune bones where possible. A kinematic chain that has multiple consecutive static joints connected by bones will be reduced to the longest possible bone, removing unnecessary joints. However, a joint that functions as a center of rotation for non-static children joints is never pruned. Lastly, static end effector joints are removed. An example of the procedure can be seen Fig. 13.

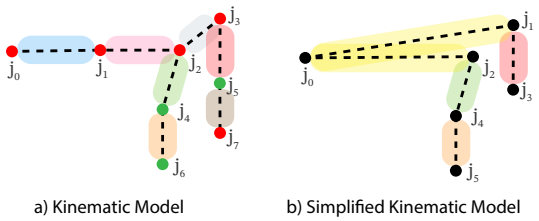


Figure 13: Visualization of kinematic model simplification. a) The trained kinematic mode: The nodes visualize whether the joint is static (red) or not (green). Each bone is associated with a blend-skinning weight (rounded colored rectangles), and j_0 is the root node. b) The simplified kinematic model: Based on the static joints, bones have been pruned and weights have been merged where possible. Weights of bones (j_0, j_1), (j_1, j_2), and (j_2, j_3) have been merged into the root weight (yellow), as all of the joints were marked as static. Furthermore, joints j_1 and j_7 could be removed without harming the underlying kinematic model because they do not have an effect on point cloud deformation. The root node is never pruned.

E Extra Results Ablations

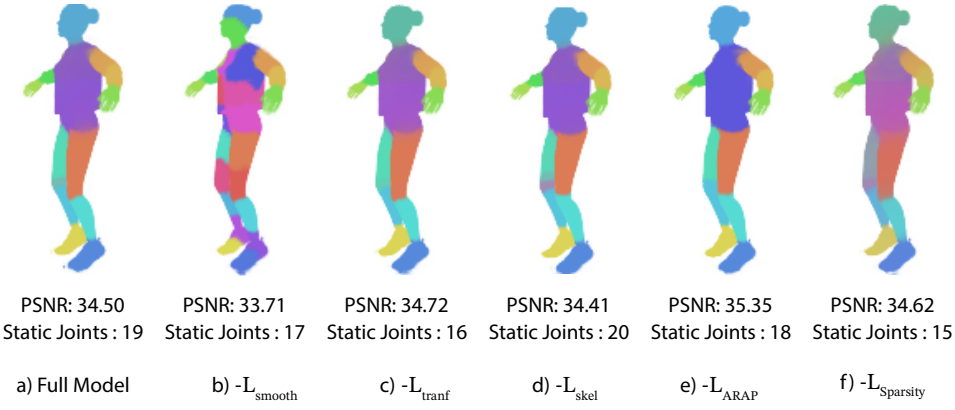


Figure 14: Additional ablation results on *Jumping Jacks* scene from the *Blender* dataset. The skinning weights and PSNR values are displayed prior to the simplification step. The static joint counts refer to the number of joints that can be removed in our simplification step to ease the reposing task. See Sec. D. b) - f) show the show the results without the denoted regularization term.

Extra ablation results for the *Jumping Jacks* scene from the *Blender* dataset can be seen in Fig. 14. We observe that without $\mathcal{L}_{smooth}$ (Fig. 14b), the skinning weights are less consolidated and wrong

Symbol	Description	Trainable	Initialization
$\mathbf{p}_i$	3D Point from canonical point cloud	Fixed	Pre-trained model
$\mathbf{f}_i$	Feature vector associated each $\mathbf{p}_i$	Fine-tuned	Pre-trained model
$\hat{\mathbf{w}}_i$	Blend skinning vector **before** softmax	Fine-tuned	Inverse bone-to-point distance
α	global scalar which scales $\hat{\mathbf{w}}_i$	Trained from scratch	0.1
Φ_c	Color regressor	Fine-tuned	Pre-trained model
Φ_d	Density regressor	Fine-tuned	Pre-trained model
Φ_p	Feature decoder	Trained from scratch	Random
Φ_r	Pose regressor	Trained from scratch	Random
Φ_{pe}	Pose embedding network	Trained from scratch	Random

Table 7: Overview of our notation.

skinning weights are produced (compared to the full model in Fig. 14a). This limits the image quality represented by the PSNR score. Next, without $\mathcal{L}_{tranf}$ (Fig. 14c) joint rotations are less sparse and, therefore, we detect fewer static joints for removal during the skeleton simplification (see Sec. D). The presented static joint counts were measured for a simplification threshold of 20 degrees). Next, $\mathcal{L}_{skel}$ (Fig. 14d) does not have a major effect in this scene. However, for scenes with thin structures, omission of $\mathcal{L}_{skel}$ allows the skeleton to drift outside of the geometry. This results in bad reconstruction (see Fig. 8c in the paper). Next, utility of $\mathcal{L}_{ARAP}$ depends on the object class. For strictly articulated shapes (see the robot in Fig. 8a in the paper), it enforces part rigidity and avoids unrealistic deformations. However, its contribution is less obvious for partially soft shapes such as humans (see Fig. 14e). Despite this, we used the same loss weights for all our results without notable issues. Finally, omission of $\mathcal{L}_{sparse}$ decreases separation of the part labels (note the reduced label color saturation in Fig. 14f). This suggests that this term reduces entanglement between points and joints which indirectly leads to a higher static joint count for skeleton simplification.

F Notation

An overview of our notation and trainable parameters can be found in Table 7.

G Training Schedule for the *ZJU-MoCap* dataset

We make two adjustments while training in the *ZJU-MoCap* dataset that both aim to compensate for a bias towards observation of early timestamps due to our incremental training data scheduler. We deem this important since the real captured human subjects do not exhibit all motion modalities uniformly through the entire sequences. This is different from the synthetic *Robots* and *Blender* datasets where motion is simplistic, exaggerated and evenly distributed throughout the sequences.

First, we sample the training timestamps with importance sampling to increase probability of later timestamps and compensate for their shorter overall accessibility during the course of training. We define an importance of a timestamp as the inverse of the total sample count so far. Second, we only enable the sparsity regularization after 160K training iterations when the scheduler converges and all timestamps become accessible. This avoids loss of kinematic joints by early merging when not all motion modalities were observed yet. Neither change leads to a computational overhead.

H Ours^{pose}: An extension for local deformations

Our full model is based on skeletal articulation with Linear Blend Skinning and as such it can well model large locally rigid deformations. While well suited for many synthetic objects, this alone does not accurately model local deformations of fabrics in the clothes of humans subjects in the *ZJU-MoCap* dataset. Despite this, our full method can perform meaningful articulation of the human subjects as shown in the main paper.

To further improve reconstruction of small surface details, we propose to extend our full method and additionally condition the feature regressor Φ_p by a pose embedding $pe^t \in \mathbb{R}^{64}$ such that $\mathbf{f}^t_{i,x} = \Phi_p(\mathbf{f}_i, \underline{pe^t}, x^t_{\mathbf{p}_i})$ (see Eq. 5 from the main paper). Here, pe^t is regressed by another MLP (4 layers, $0.5 \times |\gamma(\mathbf{J}^c)|$ hidden dimensions) from the canonical joint positions $\mathbf{J^c}$ and the joint positions

$\mathbf{J^t}$ observed at time t as $pe^t = \Phi_{pe}(\gamma(\mathbf{J}^c - \mathbf{J}^t))$. This allows the method to learn additional local geometry deformations and color changes specific to the current pose (see Fig. 15). Note, that the poses are not an input to our method as they are already jointly learned in our original full model. We observe, that the learned deformations are locally constrained by the fixed neighborhood search radius in Eq. 6. Furthermore, we detach the gradient flow from $\mathbf{J^t}$ such that the skeletal poses are not optimized to fit the local deformations. Please refer to the main paper for a comparison to our unmodified full method and WIM [20].

Figure 15: Example of clothes deformations when conditioning Φ_p on the pose embedding in Ours^{pose}.

I Additional Animations

We demonstrate animation capabilities of our method by creating two animation sequences based on manually defined joint rotations. First, we create a walking sequence of the *Spot* robot from the *Robots* dataset in Fig. 16. Second, we create an over-extended jaw opening animation for the *Trex* from the *Blender* dataset in Fig. 16. Lastly, we animate a scene from *ZJU-MoCap* from two different views Fig. 18.

Figure 16: Manually defined walking animation for the *Spot* from the *Robots* dataset.

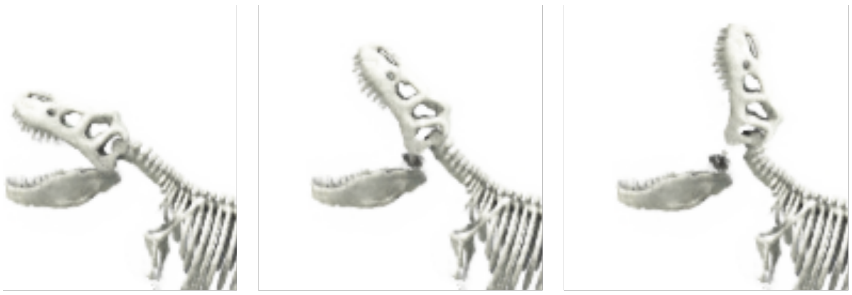


Figure 17: Manually defined jaw over-extension animation for the *T-Rex* from the *Blender* dataset.

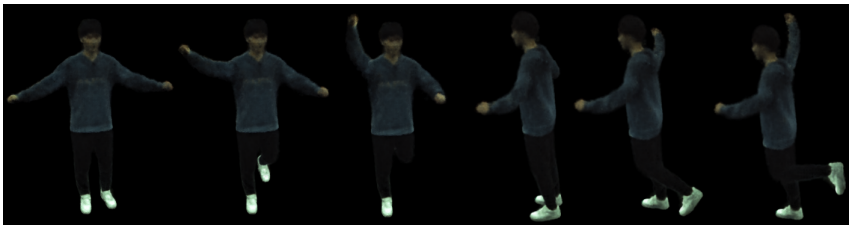


Figure 18: Manually defined animation for scene *384* from the *ZJU-MoCap* dataset from two different views.