Unbiased 4D: Monocular 4D Reconstruction with a Neural Deformation Model

Erik C.M. Johnson[1, 2] Marc Habermann[1] Soshi Shimada[1] Vladislav Golyanik[1] Christian Theobalt[1]
[1]Max Planck Institute for Informatics, SIC [2]Saarland University, SIC

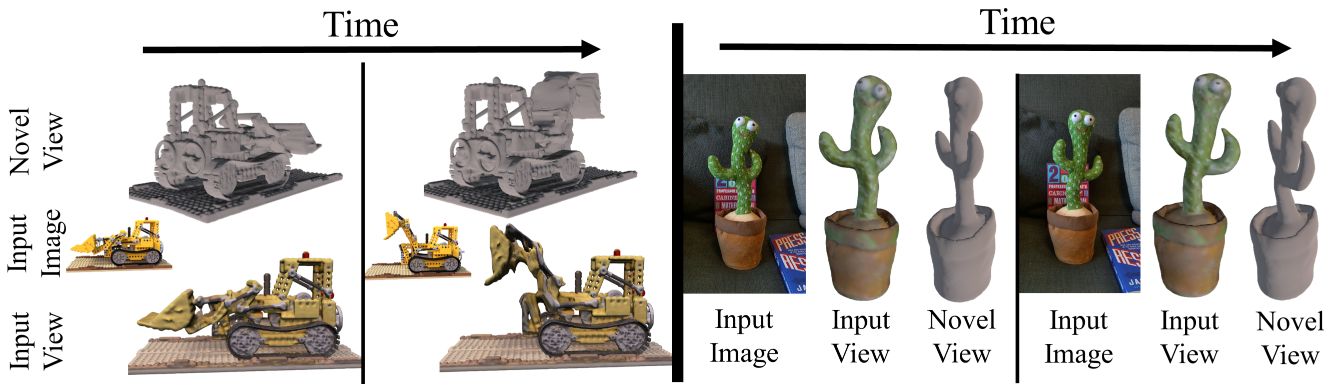


Figure 1: **We present a new method for 4D reconstruction of dynamic scenes using a single RGB video.** In contrast to previous work, our approach can handle small- and large-scale deformations of arbitrary objects due to our separation of non-rigid deformations using a canonical space, our unbiased volume rendering formulation, and a novel scene flow loss.

Abstract

Capturing general deforming scenes is crucial for many computer graphics and vision applications, and it is especially challenging when only a monocular RGB video of the scene is available. Competing methods assume dense point tracks, 3D templates, large-scale training datasets, or only capture small-scale deformations. In contrast to those, our method, Ub4D, makes none of these assumptions while outperforming the previous state of the art in challenging scenarios. Our technique includes two new—in the context of non-rigid 3D reconstruction—components, i.e., 1) A coordinate-based and implicit neural representation for non-rigid scenes, which enables an unbiased reconstruction of dynamic scenes, and 2) A novel dynamic scene flow loss, which enables the reconstruction of larger deformations. Results on our new dataset, which will be made publicly available, demonstrate the clear improvement over the state of the art in terms of surface reconstruction accuracy and robustness to large deformations. Visit the project page `https://4dqv.mpi-inf.mpg.de/Ub4D/`.

1. Introduction

Reconstructing the deforming 3D geometry of an object from image data is a long-standing and important problem in computer vision with many applications in the movie and game industries, as well as VR and AR. Especially interesting and the subject of this work is the 4D reconstruction from a single RGB video, as this is the most intuitive and user-friendly capture setup. Over the last decade, many monocular 4D reconstruction approaches have been proposed; they can be categorized into dense non-rigid structure from motion (NRSfM) methods, shape-from-template (SfT) approaches, and neural template-free approaches.

NRSfM methods [5, 46, 10, 1, 28, 17, 26, 43] usually assume dense and coherent 2D point tracks connecting the frames of the video. While accurate results can be obtained, it is usually hard to satisfy this assumption in real-world captures, limiting the use case in practice. SfT methods [39, 25, 58, 42, 9] assume an object template is given. While this provides a strong prior for this highly ill-posed task, initial reconstruction errors in the template can lead to tracking errors. Importantly, topological changes cannot be captured by such methods. Last, template-free learning-based approaches have shown compelling results for category-specific (*e.g.,* humans [38, 37]) and general scenes with small deformations [47]. However, generalization beyond categories and the reconstruction of large scale deformations remains a challenge.

To this end, we propose Unbiased 4D (Ub4D), *i.e.,* a novel method for the 4D reconstruction of a deforming object given a single RGB video of the object and, optionally, a rough proxy geometry; see Figure 1. Using a signed distance field (SDF) network, we represent the object of interest as an implicit SDF in canonical space. In order to obtain the deformed per-frame geometry, we propose a bending network, which deforms the current frame into a shared canonical space. To supervise the SDF and bending network, we impose a volume rendering loss extending prior work on unbiased volume rendering [51] to dynamic scenes. In particular, we compare the rendered images and object segmentation masks with the ground truth images and masks. This formulation alone still struggles with larger scene deformations. Thus, in addition, we propose a scene flow loss, which attaches free space to a rough mesh or sparse 3D proxy in order to guide the scene deformations predicted by the bending network. In summary, our primary **technical contributions** are as follows:

- Ub4D, *i.e.,* a new approach for dense 4D surface reconstruction from monocular image sequences based on an implicit surface representation and a dynamic bending network.
- A new scene flow loss leveraging coarse geometric proxies (dense and sparse), which further increases the robustness to large-scale scene deformations.
- Extending the unbiased formulation of volume rendering [51] to general deforming scenes.

We also introduce a new benchmark dataset for general and large-scale deforming scenes and demonstrate that our method outperforms the previous state of the art in terms of accuracy and robustness to large scale scene deformations. The code and the new dataset will be made publicly available for future research.

2. Related Work

Several method classes for 3D reconstruction of non-rigidly deforming surfaces and shapes from monocular image sequences are known in the literature. They differ in the assumptions they make about the available priors and types of motions and deformations. In this section, we review the methods most closely related to our approach.

Non-Rigid Structure from Motion (NRSfM). NRSfM operates on point tracks over the input monocular views [5, 46]. It then factorizes them into camera poses and deformable (per-frame) geometry of observed surfaces. Assuming that accurate point tracks can be obtained is a restrictive assumption in practice. If points of the input views are tracked densely, NRSfM can then even be used to obtain dense surfaces [10, 1, 28, 11]. Both neural NRSfM methods for the sparse [17, 26, 49] and dense [43] cases were recently proposed in the literature. Deep NRSfM [17, 26, 49, 43] is related to NRSfM in that it lifts 2D input points in 3D and does not rely on 3D supervision. Ub4D is similar to NRSfM in that 1) it has the least number of assumptions (no training datasets, no 3D priors) and 2) requires camera or object movement while recording the scene. It differs from NRSfM in that it operates directly on images with no need for 2D correspondences.

Shape from Template (SfT). This class of techniques assumes a 3D shape prior called a *template*. SfT is then posed as the problem of tracking and deforming the template so that the new states plausibly reproject to the input images [39, 25, 58, 12]. While some approaches have demonstrated accurate results, even for larger deformations, they come at the cost of being category-specific (*e.g.,* they only work for humans [13, 4]). Further, the assumption of a known 3D template is limiting when dealing with unknown objects. Moreover, obtaining the template usually requires a separate step, which can be difficult. ϕ-SfT [14] explains 2D observations through physics-based simulation of the deformation process. In contrast to them, we do not model physics laws explicitly. Moreover, we target a different class of non-rigid objects (thin surfaces [14] *vs* articulated objects). Deep SfT or direct surface regression methods assume multiple states available for training [42, 9]. Our approach differs from SfT in that it only requires 2D images as input. Nonetheless, it can benefit from a subset of frames observing static scene states to initialise the canonical volume. Note that—in contrast to SfT techniques—observing a scene under rigidity assumption [58] or having a template in advance from elsewhere is not a strict requirement for us.

Monocular 3D Mesh Reconstruction. 3D mesh reconstruction methods deform an initial mesh to match image observations [50, 15, 18, 53]. They are exclusively neural techniques, usually trained using 2D data only, *i.e.,* raw image collections. One of their limitations is that large sets of images are not available for all object categories (*e.g.,* consider rarely observed biological species). Moreover, the methods, which do not require 2D image priors, might capture coarse articulations but fail to reconstruct fine surface details [15, 53]. Starting from a sphere mesh is a restricting assumption. Even though many watertight meshes are, in theory, topologically equivalent to a sphere, a practical attempt to guide sphere deformations by image cues can converge to local minima.

Free Viewpoint Video and Neural Surface Extraction. Coordinate-based volumetric neural representations learned from 2D observations, such as NeRF [24], can be used to render high-quality novel views of rigid [57, 7, 20] and non-rigid scenes [47, 34, 21, 33, 29, 35, 19, 52]. While they have shown impressive results, the volumetric representation they use lacks surface constraints so that it is difficult to extract high-quality surfaces from the learned representation. At the same time, some works [27, 51, 55, 56] propose to represent 3D scenes as a neural SDF and use volume rendering to learn the representation. We are inspired by the recent progress in neural volumetric representations learned without 3D supervision. Even though our goal is not novel view rendering and editing, we show that a NeRF-inspired component can be useful for monocular non-rigid 3D reconstruction. Moreover, surface extraction methods [27, 51, 55, 56] have focused on rigid objects so far. Thus, we demonstrate that the problem we are interested in, *i.e.,* monocular *non-rigid* 3D reconstruction, significantly benefits from advances in another, distantly related research direction [44].

3. Method

The goal of Ub4D is to reconstruct the dense and deforming surface of an object from a single RGB video. Therefore, our method takes as input the monocular image sequence $\{I_i, S_i : i \in [1, N_f]\}$ of the segmented object consisting of N_f RGB images I_i and respective segmentation masks S_i. We assume the extrinsic and intrinsic camera parameters are known. Optionally, corresponding per-frame coarse geometric proxies with N_v vertices can be provided $\{M_i : i \in [1, N_f]\}$, where $M_i = \{\boldsymbol{v}_i^{(k)} : k \in [1, N_v]\}$ and $\boldsymbol{v}_i^{(k)}$ denotes vertex k of the mesh in frame i. Note that we only use corresponding vertices, i.e. no connectivity information, which allows the use of sparse point sets (e.g. skeleton) as the geometric proxy. Given these inputs, Ub4D outputs an explicit geometry for every frame.

A diagram of our method, along with its inputs and outputs, is shown in Figure 2. We first describe our model for non-rigid deformations in Section 3.1. Then we present a rendering method for supervision with 2D images and object segmentations (Section 3.2). Next, we formulate a novel loss on the scene flow using a geometric proxy provided as input to our method (Section 3.3). Finally, we explain how an explicit geometry can be obtained from our implicit scene representation (Section 3.4).

3.1. Non-Rigidity Model

We model temporal non-rigid deformations as a vector field projecting points from the frame space into a canonical space. One can conceptualize this by considering it as a bending of the straight rays originating from the camera. Given a straight ray with an origin $\boldsymbol{o} \in \mathbb{R}^3$ and a viewing direction $\boldsymbol{d} \in \mathbb{R}^3$ as $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$, we bend this ray with a frame-specific bending network $\boldsymbol{b}_i : \mathbb{R}^3 \to \mathbb{R}^3$ as $\tilde{\boldsymbol{r}}_i(t) = \boldsymbol{r}(t) + \boldsymbol{b}_i(\boldsymbol{r}(t))$ where i denotes the frame. This bent ray is a directed parametric path in $\mathbb{R}^3$ like the straight ray, but, where the derivative of the straight ray is constant (*i.e.* $\frac{\mathrm{d}\boldsymbol{r}(t)}{\mathrm{d}t} = \boldsymbol{d}$), the bent ray has an instantaneous direction at each point along it of $\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} = \boldsymbol{d} + \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}\boldsymbol{d}$. We desire that this bending network transforms points from frame space into a single canonical representation of the object shared by all frames of the input.

While this discussion presents the bending network as a per-frame vector field throughout $\mathbb{R}^3$, it is implemented using a per-frame latent code $\boldsymbol{l}_i \in \mathbb{R}^{64}$. This latent code is given as input along with a point in space to a Multi-Layer Perceptron (MLP) $\boldsymbol{b} : (\mathbb{R}^3, \mathbb{R}^{64}) \to \mathbb{R}^3$ and the latent codes are optimized during training. $\boldsymbol{l}_i$ passed to the bending network is the *only* frame-specific element in our method and no other network receives it. One can see this as a factorization between the temporal and spatial domains where our method forces time to be entirely modeled by the latent code and the bending network.

This is similar to the non-rigidity model employed in NR-NeRF [47]. However, we propose a different regularization to enable the modeling of larger deformations, which also removes the need to learn a rigidity score throughout the scene. Whereas NR-NeRF [47] penalizes the bending network output for its absolute length, we instead enforce that the deformation of the current frame is similar to that of the neighboring frames. This assumes that neighboring frames represent similar object states, which is a more reasonable assumption for dynamic scenes compared to the absolute amount of deformation. More specifically, for N_s samples along a straight ray $\boldsymbol{r}$, we penalize the bending network as:

$$L_{\text{NBR}} = \frac{1}{N_s} \sum_{z=1}^{N_s} \sum_{j \in \mathcal{N}(i)} \omega_i^{(z)} \, ||\boldsymbol{b}_i(\boldsymbol{r}(t^{(z)})) - \boldsymbol{b}_j(\boldsymbol{r}(t^{(z)}))||_2^2, \quad (1)$$

where $\omega_i^{(z)}$ is the visibility weight at sample z along the bent ray (see Section 3.2) and $\mathcal{N}(i)$ are the neighbours of frame i. We also penalize the divergence of the bending network as:

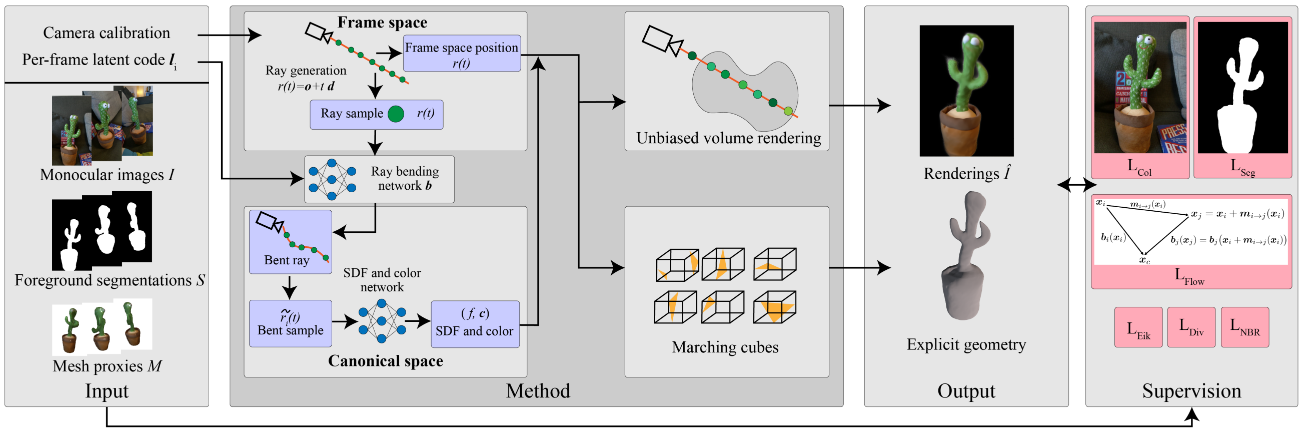
$$L_{\text{DIV}} = \frac{1}{N_s} \sum_{z=1}^{N_s} \omega_i^{(z)} \, |\nabla \cdot \boldsymbol{b}_i(\boldsymbol{r}(t^{(z)}))|^2, \quad (2)$$

where we use the unbiased, approximated divergence as presented in Tretschk *et al.* [47].

3.2. Rendering Method

Recent studies on *static* scene reconstruction have demonstrated that volume rendering enables more stable training compared to surface rendering [56, 51, 55]. Therefore, we extend the volume rendering method proposed in NeuS [51] to *dynamic* scenes. Let $f : \mathbb{R}^3 \to \mathbb{R}$ be a Signed Distance Field (SDF) modeled by an MLP that takes as input sampled points $\tilde{\boldsymbol{r}}_i(t)$ on the bent ray. Then, NeuS [51] shows that we can compute the opaque density as:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \max\left\{ \frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}, 0 \right\}, \quad (3)$$



Camera calibration
Per-frame latent code l_i
Monocular images I
Foreground segmentations S
Mesh proxies M
Input
Frame space
Ray generation
$r(t)=o+t\,d$
Frame space position $r(t)$
Ray sample $r(t)$
Ray bending network b
Bent ray
SDF and color network
$\tilde{r}(t)$ Bent sample
(f, c) SDF and color
Canonical space
Method
Unbiased volume rendering
Marching cubes
Renderings $\hat{I}$
Explicit geometry
Output
$\mathrm{L}_{\mathrm{Col}}$
$\mathrm{L}_{\mathrm{Seg}}$
x_i
$m_{i\to j}(x_i)$
$x_j = x_i + m_{i\to j}(x_i)$
$b_i(x_i)$
$b_j(x_j) = b_j(x_i + m_{i\to j}(x_i))$
x_c
$\mathrm{L}_{\mathrm{Flow}}$
$\mathrm{L}_{\mathrm{Eik}}$
$\mathrm{L}_{\mathrm{Div}}$
$\mathrm{L}_{\mathrm{NBR}}$
Supervision

Figure 2: Our Ub4D approach takes a sequence of images and respective foreground segmentations recorded with a single calibrated RGB camera as input. In addition, each frame is also equipped with a learnable latent code. Given this, our method learns a canonical and colored SDF representing the static scene. Our bending network then maps the frame space to canonical space and volume rendering and marching cubes can produce per-frame renderings and geometries, respectively. We then weakly supervise our scene representation with image-based losses as well as spatio-temporal priors including our novel scene flow loss.

where Φ_s is the logistic Cumulative Distribution Function (CDF) with standard deviation s^{-1}. This is in contrast to the original formulation, which operates on unbent ray sample points rather then bent ones. To calculate the color of each camera ray, we employ a hierarchical sampling procedure with N_s samples in total (samples in coarse and fine stages) along the bent ray $\{\tilde{\boldsymbol{r}}_i(t^{(z)}) : z \in \mathbb{Z}, z \in [1, N_s]\}$ where $t^{(z)} < t^{(z+1)}, \forall z$. Then, the color of the ray can be computed as:

$$\hat{I}(\tilde{\boldsymbol{r}}_i) = \sum_{z=1}^{N_s-1} \omega_i^{(z)} \, \boldsymbol{c}\Big(\tilde{\boldsymbol{r}}_i\big(t^{(z)}\big), \tilde{\boldsymbol{r}}_i\big(t^{(z+1)}\big) - \tilde{\boldsymbol{r}}_i\big(t^{(z)}\big)\Big), \tag{4}$$

where $\boldsymbol{c}(\cdot)$ is a color function modeled by an MLP, which takes as input the point position $\tilde{\boldsymbol{r}}_i(t)$ and the viewing direction of the ray at that point, which is approximated with a forward difference. The weight $\omega_i^{(z)}$ is occlusion-aware and *unbiased* with respect to the object's surface [51], which is formulated based on the opaque density $\rho_i(\tilde{\boldsymbol{r}}_i(t))$ from Equation (3) as follows:

$$\omega_i^{(z)} = T_i^{(z)} \alpha_i^{(z)}, \text{ where} \tag{5}$$

$$T_i^{(z)} = \prod_{\zeta=1}^{z-1} (1 - \alpha_i^{(\zeta)}), \text{ with} \tag{6}$$

$$\alpha_i^{(z)} = \max\left\{ \frac{\Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z)})\big)\Big) - \Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z+1)})\big)\Big)}{\Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z)})\big)\Big)}, 0 \right\}, \tag{7}$$

$$\text{and } \alpha_i^{(z)} = 1 - \exp\left(- \int_{t^{(z)}}^{t^{(z+1)}} \rho(t) \mathrm{d}t \right). \tag{8}$$

Importantly, the discrete opacity $\alpha_i^{(z)}$ derivation from NeuS [51] still applies in the case of a bent ray as replacing the constant viewing direction with $\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}$ does not affect the analysis.

In addition to the color, we can determine if a ray intersects the object by computing the sum of the weights:

$$\hat{S}(\tilde{\boldsymbol{r}}_i) = \sum_{z=1}^{N_s-1} \omega_i^{(z)}, \tag{9}$$

where $\hat{S}$ approaches 1 for a ray intersecting the object and otherwise $\hat{S}$ approaches 0.

Supervision. We supervise the dynamic scene representation by computing the L1 distance between the color of each bent ray $\tilde{\boldsymbol{r}}_i^{(p)}$ and the corresponding ground-truth color $I_i^{(p)}$ of pixel p:

$$L_{\mathrm{COL}} = \frac{1}{N_p} \sum_{p=1}^{N_p} \left| \hat{I}\big(\tilde{\boldsymbol{r}}_i^{(p)}\big) - I_i^{(p)} \right|, \tag{10}$$

where N_p is the number of pixels sampled from frame i. To more explicitly ensure that our approach solely focuses on reconstructing the foreground object, we also define a segmentation loss L_{SEG} as the binary cross entropy between the estimated segmentation $\hat{S}\big(\tilde{\boldsymbol{r}}_i^{(p)}\big)$ and the ground truth

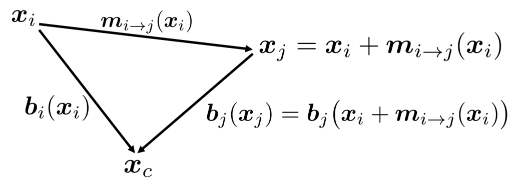

$\boldsymbol{x}_i$
$\boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)$
$\boldsymbol{x}_j = \boldsymbol{x}_i + \boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)$
$\boldsymbol{b}_i(\boldsymbol{x}_i)$
$\boldsymbol{b}_j(\boldsymbol{x}_j) = \boldsymbol{b}_j\big(\boldsymbol{x}_i + \boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)\big)$
$\boldsymbol{x}_c$

Figure 3: Graphical depiction of the relationship between the scene flow from frame i to j (*i.e.,* $\boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)$) and the bending network projecting both points to the same canonical position $\boldsymbol{x}_c$.

object segmentation $S_i^{(p)}$. Finally, we enforce f to be an SDF with the Eikonal loss defined as follows:

$$L_{\text{EIK}} = \frac{1}{N_p N_s} \sum_{p=1}^{N_p} \sum_{z=1}^{N_s} (|\nabla f(\tilde{\boldsymbol{r}}_i^{(p)}(t^{(z)}))| - 1)^2. \tag{11}$$

3.3. Scene Flow Loss

So far, very large scene deformations remain a challenge for Ub4D since it can create erroneous multiple geometries in the canonical space to best explain the monocular observations. This is particularly noticeable for scenes containing large translations (see Figure 5b). To resolve this, we accept an additional input in the form of a coarse and coherent per-frame geometric proxy. From these coarse 3D correspondences, we can compute an estimate of the scene flow, which can then be used to regularize the bending network. This greatly reduces the effect of duplicate geometries in the canonical space.

Consider a function $\boldsymbol{m}_{i\to j} : \mathbb{R}^3 \to \mathbb{R}^3$ that returns the scene flow estimate at a point from frame i to j. The scene flow allows us to transform points from a frame i into any other frame j as $\boldsymbol{x}_j = \boldsymbol{x}_i + \boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)$. Given that the bending network projects a point $\boldsymbol{x}_i$ in frame space into canonical space resulting in the point $\boldsymbol{x}_c$, it follows:

$$\begin{aligned} \boldsymbol{x}_c &= \boldsymbol{x}_i + \boldsymbol{b}_i(\boldsymbol{x}_i) = \boldsymbol{x}_j + \boldsymbol{b}_j(\boldsymbol{x}_j) = \\ &= \boldsymbol{x}_i + \boldsymbol{m}_{i\to j}(\boldsymbol{x}_i) + \boldsymbol{b}_j\big(\boldsymbol{x}_i + \boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)\big). \end{aligned} \tag{12}$$

Intuitively, this means that a point in frame i and its corresponding point in frame j determined through the scene flow $\boldsymbol{m}_{i\to j}(\boldsymbol{x}_i)$ should be mapped to the same point $\boldsymbol{x}_c$ in canonical space by the bending network (see Figure 3). We can then formulate it as a loss for a set $\mathcal{X}$ of sampled points:

$$L_{\text{FLO}} = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} ||\boldsymbol{m}_{i\to j}(\boldsymbol{x}) + \boldsymbol{b}_j\big(\boldsymbol{x} + \boldsymbol{m}_{i\to j}(\boldsymbol{x})\big) - \boldsymbol{b}_i(\boldsymbol{x})||_2^2. \tag{13}$$

The scene flow from the geometric proxies can only be directly computed on the surface. However, our implicit surface representation can potentially be evaluated at any point in 3D space. Thus, we extrapolate this scene flow to any point in $\mathbb{R}^3$ with a convex combination over the vertices using a kernel function depending on the distance to the vertices inspired by the spatial weighting approach in bilateral filtering [45]:

$$\boldsymbol{m}'_{i\to j}(\boldsymbol{x}) = \frac{\sum_{k=1}^{N_v} w_{\lambda_1}\big(||\boldsymbol{x} - \boldsymbol{v}_i^{(k)}||_2\big)\,\big(\boldsymbol{v}_j^{(k)} - \boldsymbol{v}_i^{(k)}\big)}{\sum_{k=1}^{N_v} w_{\lambda_1}\big(||\boldsymbol{x} - \boldsymbol{v}_i^{(k)}||_2\big)}, \tag{14}$$

where $w_{\lambda_1}(x) = e^{-\lambda_1 x^2}$ is a kernel function with λ_1 as a scale parameter affecting the weighting of vertex flow estimates. Additionally, we add an attenuation term, so that the scene flow falls off as the distance to the nearest vertex increases:

$$\boldsymbol{m}_{i\to j}(\boldsymbol{x}) = w_{\lambda_2}\Big(\min_{k=1}^{N_v} ||\boldsymbol{x} - \boldsymbol{v}_i^{(k)}||_2\Big)\,\boldsymbol{m}'_{i\to j}(\boldsymbol{x}) \tag{15}$$

where $w_{\lambda_2}(x) = e^{-\lambda_2 x^2}$ is a kernel function with λ_2 as a scale parameter defining the extent of the kernel.

3.4. Surface Extraction

To convert our deforming and implicit scene representation into an explicit geometry, we use the Marching Cubes algorithm [22]. For points sampled in frame i, we transform them from the frame space into the canonical space, *i.e.,* $\boldsymbol{x}_c = \boldsymbol{x}_i + \boldsymbol{b}_i(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ is a point sampled for marching cubes and $\boldsymbol{x}_c$ is the canonical space point at which we then evaluate the SDF. We restrict the selection of frame march points to the camera frustum of the given frame since any space not seen in that frame is unconstrained by our reconstruction losses and may contain aberrant geometry.

4. Results

In the following, we visually and quantitatively compare our method to previous works on monocular 4D scene reconstruction (Sec. 4.1). Next, we validate the design choice of using an SDF network rather than a density network (Sec. 4.2). Finally, we ablate other important design choices over several baselines (Sec. 4.3) and show more qualitative results on real world data (Sec. 4.4). All experiments were performed using a single NVIDIA Quadro RTX 8000 with 48 GB RAM. For more qualitative results, we refer to the supplement and the video.

Dataset. We aim at reconstructing the *full* deforming geometry and, thus, the monocular capture requires sufficient camera motion around the dynamic object to observe every part at least once. However, we found that existing datasets either capture static scenes with a circulating camera path around the object or dynamic scenes with very limited camera motion. Therefore, we create our own dataset of dynamic objects with sufficient camera motion, which contains synthetic scenes for quantitative evaluations and real scenes for qualitative results.

For the synthetic evaluation, we create two scenes in Blender [8] showing a deforming cactus, referred to as *Cactus*, and a moving human, referred to as *RootTrans*. Each of the scenes has an image resolution of 1024×1024 and is 150 frames long. We define a moving camera viewing the dynamic object and provide the camera parameters as input to our method. To generate the proxy geometries, which are required for our proposed scene flow loss, we leverage a human capture method [31] for the *RootTrans* sequence (further details are included in our supplementary material). For the *Cactus* sequence, we use a highly downsampled version of the ground-truth geometry as a coarse proxy. A visualization of the proxies is shown in Figure 4.

For the evaluation of our method on real data, we capture two sequences: one of a moving human, called the *Humanoid* sequence, and one of a deforming cactus toy, called *RealCactus*. We capture these sequences at resolutions of 960×1280 and 1080×1920, respectively, with a mobile phone camera. Again each sequence contains around 150 frames. To obtain the camera parameters, we use the rigid Structure from Motion (SfM) software COLMAP [40, 41]. As with the *RootTrans* synthetic sequence, we generate proxy geometries for the *Humanoid* sequence using the same human capture method [31]. However, unlike the *RootTrans* sequence, we only input the sparse *skeleton* (i.e. 12 vertices) as the proxy, rather than the full posed SMPL-X [31] model. This demonstrates that our proxy need not include any information about the location of the surface. The *RealCactus* sequence does not have proxy geometry as it includes an initially rigid subsequence. For the foreground masks, we manually labeled a few frames and then trained a segmentation network, based on the UNet architecture [36], on those labeled frames, which then provides masks for all frames in a semi-automated fashion.

For additional evaluation of our method, we leverage the *Lego* object [1] made available by Mildenhall *et al.* [24], which we animate over time by lifting the boom and tilting the bucket (see Figure 5a), to obtain a dynamic scene. Further, we defined a monocular camera path for 150 frames, rendered monocular images and masks at a resolution of 800×800, and extracted the known camera extrinsics and intrinsics. This scene includes an initially rigid subsequence and does not require proxy geometry for our method.

Evaluation Metrics. To quantitatively evaluate our method and compare it to the state-of-the-art methods, we compute the Chamfer distance (CD) and Hausdorff distance (HD) between our result and the ground-truth geometry. Since we evaluate on synthetic scenes without a meaningful physical scale, we report the absolute numbers without any physical unit. The baseline methods either assume a fixed camera [58] or predict the camera [53, 43] and, in both cases, we apply ICP [3] to rigidly align their meshes with the ground truth in order to compare to our method, which assumes camera motion is known.

[1]Released under CC-BY-3.0 and modifications are made. Originally created by Blend Swap user Heinzelnisse.

4.1. Quantitative Comparison

We compare our method to N-NRSfM [43], DDD [58], LASR [53], and ViSER [54]. N-NRSfM is a Non-Rigid Structure-from-Motion (NRSfM) method, which uses an auto-decoder to deform a mean shape based on a learned per-frame latent representation. DDD is template-based; it deforms the template to minimize an energy formulation. For DDD, we provide the first frame's ground-truth mesh as a template. Both LASR and ViSER do not require a template and recover a rigged mesh that is animated over the image sequence. Several other 4D reconstruction techniques with source code available online, such as Shimada *et al.* [42] or Ngo *et al.* [25], do not work under our assumptions; so, we do not include them. For each related method, we follow the original papers to find the best possible hyperparameters.

The quantitative results on the synthetic sequences are reported in Table 1a and a qualitative comparison is shown in Figure 4. Ub4D outperforms the state-of-the-arts both quantitatively and qualitatively. We found that prior work struggles with large scale deformations resulting in tracking errors [58], has a limited resolution [53], [54], or only reconstructs the frontal geometry [43] while ours accurately captures the large deformations of the entire geometry. Also note that although we rigidly align the results for other methods with the ground truth, our method still achieves the most accurate results. For completeness, we also report our results after ICP, which is even more accurate.

4.2. Comparison to Volume-based Representations

Like some previous works [51, 56], our method leverages an SDF representation to model the surface of the object. An alternative choice would be predicting volume density with a network [47, 24]. While a volume density representation has proven to be a good choice for novel view synthesis, naïvely extracting a surface from such a density representations results in a noisy and inaccurate geometry as demonstrated in Figure 5a, where we compare to the state-of-the-art method [47], called NR-NeRF, for novel view synthesis of dynamic scenes. It suffers from noisy surfaces and can result in significant inaccuracies for deforming parts of the object since these regions have relatively low densities. In contrast, an SDF representation removes the need to determine a threshold when extracting the explicit geometry and must add a zero crossing, *i.e.,* a surface, in order to satisfy the reconstruction losses. Further, this example shows the limited ability of NR-NeRF to model large deformations as they penalize the absolute

(a)

Scene	*Method*	*CD (HD)* (↓)
Cactus	LASR [53]	20.23
	ViSER [54]	14.34
	N-NRSfM [43]	102.00 (5.74)
	DDD [58]	34.71
	Ub4D (ours)	**3.06** (2.42)
	Ub4D after ICP	2.71 (2.24)
RootTrans	LASR [53]	0.39
	ViSER [54]	0.37
	N-NRSfM [43]	0.38 (0.09)
	DDD [58]	0.26
	Ub4D (ours)	**0.23** (0.14)
	Ub4D after ICP	0.03 (0.02)

(b)

Scene	*Comparison*	*CD* (↓)
Cactus	w/o L_{FLO}	8.32 †
	w/o L_{EIK}	5.47
	w/o L_{FLO}, L_{NBR}, L_{DIV}	5.34 †
	Ub4D (Ours)	**3.06**
RootTrans	w/o L_{FLO}	60.25
	w/o L_{EIK}	0.29
	w/o L_{FLO}, L_{NBR}, L_{DIV}	3.83 †
	Ub4D (Ours)	**0.23**

Table 1: a **Quantitative comparison to previous work.** We report the Chamfer distance (CD) between the ground truth object geometry and the respective reconstructions averaged over the sequence. Since N-NRSfM [43] provides only a planar surface (not a watertight mesh), we additionally report the Hausdorff distance (HD) averaged over the sequence for it and our method. Note that we quantitatively outperform the previous work. b **Quantitative ablation study.** We report the Chamfer distance (CD) between the ground-truth scene geometry and the respective reconstructions averaged over the sequence. "†" denotes frames that do not produce any geometry (due to frustum culling). Note that our full method provides the best result for both scenes.

offset length, which our neighbouring frame regularization allows us to handle.

4.3. Ablation Study

We validate our design decisions through an ablation study on the *Cactus* and *RootTrans* sequences and report the metrics in Table 1b. Our full supervision consists of six loss terms: L_{COL}, L_{SEG}, L_{EIK}, L_{FLO}, L_{NBR} and L_{DIV}. We compare the full method to removing the terms: 1) L_{FLO}, which is our novel flow loss, 2) L_{EIK}, which directly regularizes the SDF and color network and indirectly regularizes the bending network and 3) L_{FLO}, L_{NBR}, and L_{DIV}, which are all direct bending network regularizers. Most importantly, the full combination of losses provides the best result validating the contribution of each term.

Concerning 1), our flow loss especially helps for the large root translation and arm motion of the *RootTrans* sequence. Without using this loss, multiple different geometries are synthesized, which fit the reconstruction losses. Then, the bending network can "switch" between the different copies throughout the sequence. This results in overfitting to the camera pose and exploits monocular depth ambiguities to generate geometry that is not seen in other views. Figure 5b shows this overfitting to the camera pose with multiple distinct geometries being used over the sequence to satisfy the reconstruction losses.

Regarding 2), we found that not using L_{EIK} leads to overall noisier surfaces and thus the quality is reduced. Finally concerning 3), without any explicit regularization of the bending network, the deformations can be almost arbitrary again leading to overfitting to individual frames by violating 3D consistency resulting in a reduced accuracy. We even observed that the network was not able to produce any geometry for some frames, which validates the necessity of explicit regularization of the bending network.

4.4. Qualitative Results on Real Word Scenes

We next demonstrate that Ub4D works well on real-world scenes. Figure 6 visualizes our *RealCactus* sequence depicting a dancing cactus and the *Humanoid* sequence where a person is moving their arms and legs. Although in both cases the dynamic scenes contain large deformations, our method robustly and accurately reconstructs individual frame geometries, which also contain medium frequency details. We refer to our supplement for further visualizations.

5. Discussion and Possible Extensions

Ub4D significantly outperforms all competing methods in our evaluations, both numerically and qualitatively. In fact, we found that none of the existing methods can deal with captures that include object motion *and* severe camera motions while our method leverages such recording conditions to its benefit, inspired by classical non-rigid structure from motion algorithms.

However, in the case of severe scene deformations we rely on a geometric proxy. On the one hand, this is the cost for significantly improved results. On the other hand,

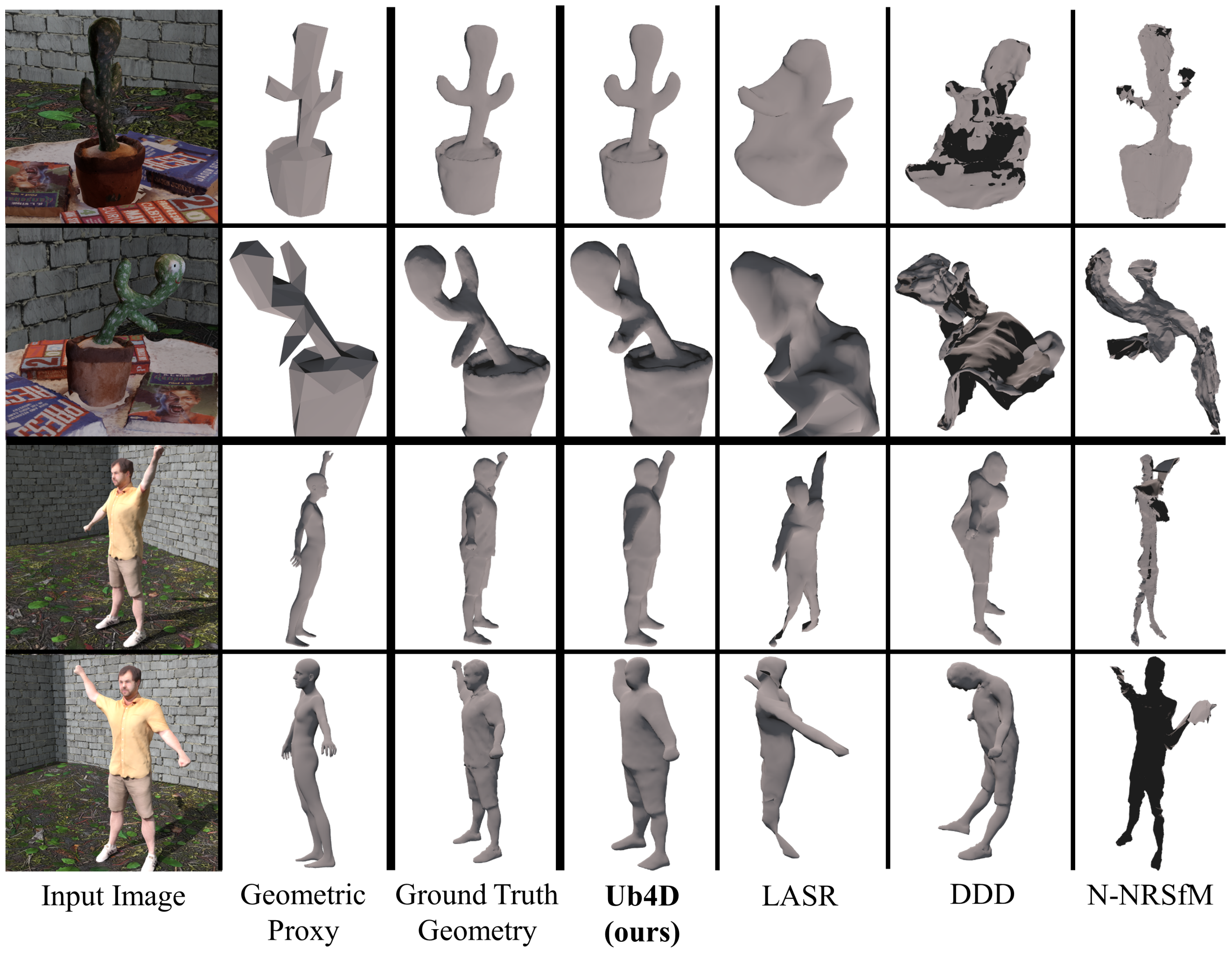
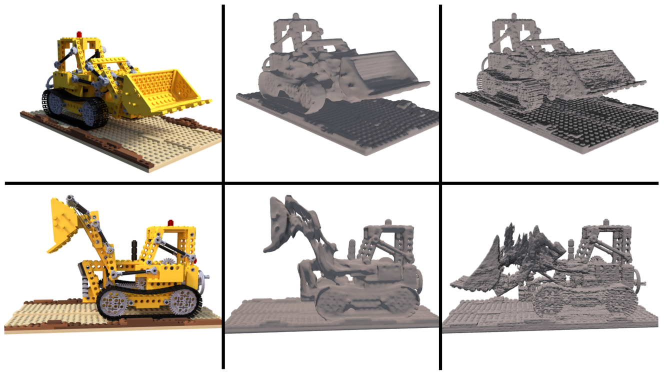


Figure 4: Qualitative comparison for select frames of our synthetic sequences rendered from novel views. Note that competing methods struggle with reconstructing the dense and deforming surface, while our method captures the large scale deformations as well as medium scale details.

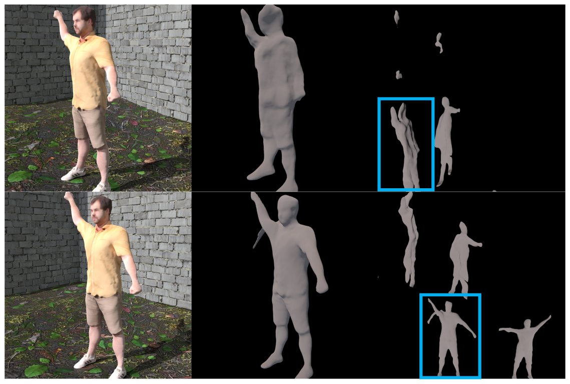
our scene flow loss is versatile in the sense that this proxy can be either a mesh or even just a few points, as long as the proxy roughly describes the deformation of the scene (see Fig. 6). Future work involves exploring this direction even further with the main question being: How sparse can the proxy be and could it even be a 2D entity in the image plane? Along these lines, we see multiple avenues for future research, including tracking a generic proxy along with learning the SDF and using 2D image features for initialising a sparse proxy.

6. Conclusion

We presented, Ub4D, a method of a new class for 3D reconstruction of deformable scenes from a single RGB camera. It represents the scene as a learned static canonical volume with an implicit surface. A bending network warping the frames into this canonical volume accounts for the scene deformation. Our scene flow loss improves the reconstruction accuracy and robustness in the case of large deformations. The qualitative and quantitative comparisons to different method types show that our approach is a clear step towards dense and deformable tracking of general and largely deforming scenes solely using a single RGB camera. We hope to see more work in this direction in the future.



(a)



(b)

Figure 5: (a): Comparison of Ub4D using an SDF scene representation (without scene flow loss) to a density-based scene representation, called NR-NeRF [47]. To generate surfaces for NR-NeRF, we apply marching cubes [22] with a threshold of 50. The density-based representation leads to an overall noisier surface compared to our approach. We also penalize bending using neighbouring frame offsets allowing Ub4D to accurately reconstruct large deformations. (b): Qualitative ablation of the scene flow loss (L_{FLO}) on the *RootTrans* sequence. Right column shows the scene from a static novel view with the geometry in the camera frustum highlighted with a blue box. Note that without the proposed scene flow loss using proxy geometry, Ub4D can produce multiple distinct copies of the character at different scales by exploiting the monocular depth ambiguity.

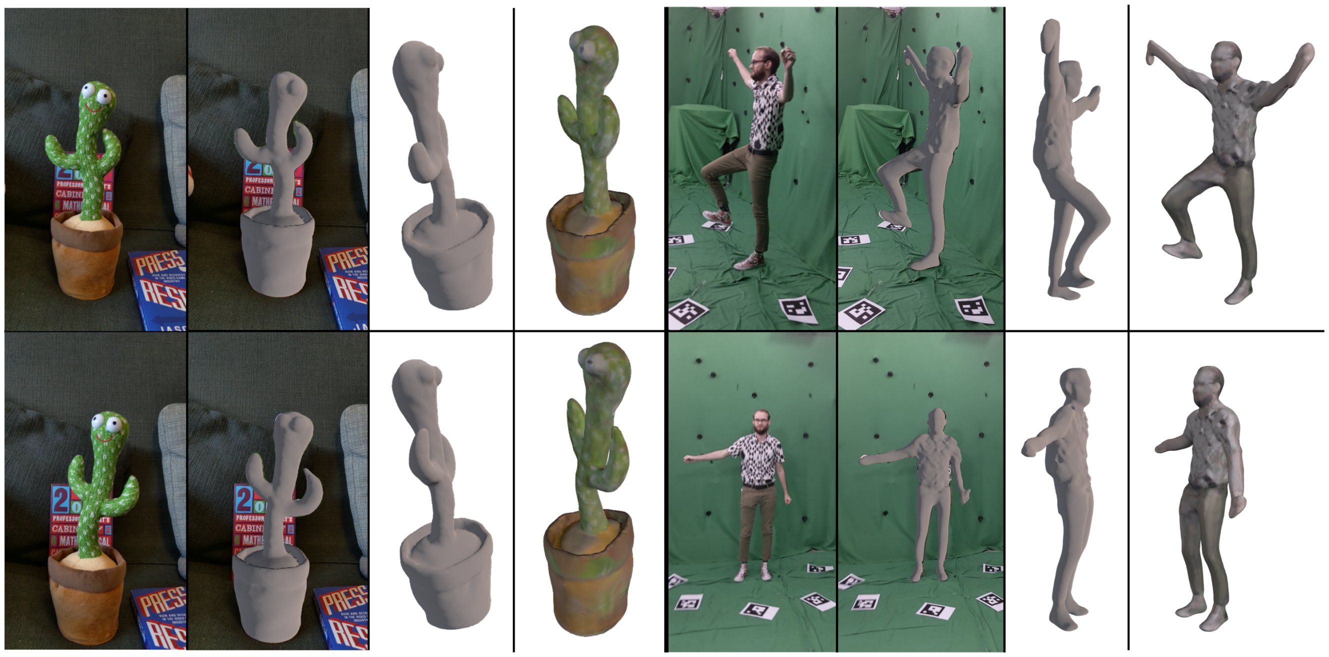


Figure 6: Qualitative results of our method for a non-humanoid object, *RealCactus*, left, and a human character, *Humanoid*, right. The scene flow loss is only used for the human character and only uses a sparse skeleton geometric proxy (12 vertices). This shows that the geometric proxy need not be dense or provide any information about the surface. Note that the recovered geometry nicely overlays onto the input image but also looks plausible from a novel 3D viewpoint.

References

[1] Mohammad D. Ansari, Vladislav Golyanik, and Didier Stricker. Scalable dense monocular surface reconstruction. In *International Conference on 3D Vision (3DV)*, 2017.

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

[3] Paul J. Besl and Neil D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: Control Paradigms and Data Structures*, 1992.

[4] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European conference on computer vision (ECCV)*, 2016.

[5] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Computer Vision and Pattern Recognition (CVPR)*, 2000.

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[7] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2018.

[9] David Fuentes-Jimenez, Daniel Pizarro, David Casillas-Perez, Toby Collins, and Adrien Bartoli. Texture-generic deep shape-from-template. *IEEE Access*, 9:75211–75230, 2021.

[10] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[11] Vladislav Golyanik, André Jonas, Didier Stricker, and Christian Theobalt. Intrinsic Dynamic Shape Prior for Dense Non-Rigid Structure from Motion. In *International Conference on 3D Vision (3DV)*, 2020.

[12] Marc Habermann, Weipeng Xu, Helge Rhodin, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Nrst: Non-rigid surface tracking from monocular video. In *German Conference on Pattern Recognition (GCPR)*, 2018.

[13] Marc Habermann, Weipeng Xu, Michael Zollhöfer, Gerard Pons-Moll, and Christian Theobalt. Livecap: Real-time human performance capture from monocular video. *ACM Transactions on Graphics (TOG)*, 38(2):14:1–14:17, 2019.

[14] Navami Kairanda, Edgar Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. ϕ-sft: Shape-from-template with a physics-based deformation model. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

[15] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision (ECCV)*, 2018.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations, ICLR*, 2015.

[17] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. In *International Conference on Computer Vision (ICCV)*, 2019.

[18] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] 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.

[21] 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. In *ACM Transactions on Graphics (TOG)*, 2021.

[22] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH*, 21(4):163–169, 1987.

[23] MATLAB. *R2020a*. The MathWorks Inc., Natick, Massachusetts, 2010.

[24] 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.

[25] Dat Tien Ngo, Sanghyuk Park, Anne Jorstad, Alberto Crivellaro, Chang D. Yoo, and Pascal Fua. Dense image registration and deformable surface reconstruction in presence of occlusions and minimal texture. In *International Conference on Computer Vision (ICCV)*, 2015.

[26] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *International Conference on Computer Vision (ICCV)*, 2019.

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

[28] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Isometric non-rigid shape-from-motion with riemannian geometry solved in linear time. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(10):2442–2454, 2018.

[29] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *International Conference on Computer Vision (ICCV)*, 2021.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[31] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[32] Karl Pearson. LIII. on lines and planes of closest fit to systems of points in space. In *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1901.

[33] 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 *International Conference on Computer Vision (ICCV)*, 2021.

[34] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[35] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[37] 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 *International Conference on Computer Vision (ICCV)*, 2019.

[38] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[39] Mathieu Salzmann, Julien Pilet, Slobodan Ilic, and Pascal Fua. Surface deformation models for nonrigid 3d shape recovery. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(8):1481–1487, 2007.

[40] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[42] Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. Ismo-gan: Adversarial learning for monocular non-rigid 3d reconstruction. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[43] 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.

[44] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edith Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik. Advances in Neural Rendering. *Computer Graphics Forum (EG STAR 2022)*, 2022.

[45] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer vision (ICCV)*, 1998.

[46] Lorenzo Torresani, Aaron Hertzmann, and Chris Bregler. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(5):878–892, 2008.

[47] 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 *International Conference on Computer Vision (ICCV)*, 2021.

[48] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research (JMLR)*, 2008.

[49] Chaoyang Wang and Simon Lucey. Paul: Procrustean autoencoder for unsupervised lifting. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[50] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *European Conference on Computer Vision (ECCV)*, 2018.

[51] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[52] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[54] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[55] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[56] 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.

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

[58] Rui Yu, Chris Russell, Neill DF Campbell, and Lourdes Agapito. Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video. In *International Conference on Computer Vision (ICCV)*, 2015.

Unbiased 4D: Monocular 4D Reconstruction with a Neural Deformation Model —Supplementary Material—

Erik C.M. Johnson[1, 2] Marc Habermann[1] Soshi Shimada[1] Vladislav Golyanik[1] Christian Theobalt[1]
[1]Max Planck Institute for Informatics, SIC [2]Saarland University, SIC

In this supplementary material, we present additional results in Section A and implementation details in Section B. Section C provides a derivation of the discrete opacity equation for bent rays and Section D shows the unbiased nature of our rendering method. Additional details on the experiments performed are given in Section E, including how we compute the geometric proxies for our human character scenes in Section E.5. Section F investigates the necessary accuracy and resolution of the geometric proxies. Section G analyzes the learned latent codes and demonstrates novel geometry synthesis. Finally, we discuss some additional limitations in Section H.

A. Additional Results

Figures 7 and 8 show additional results for our synthetic sequences: *i.e., Cactus* and *RootTrans*, respectively. We include a qualitative visualization of the output's Chamfer distance to the ground truth where dark blue is zero and red is above the mid-point between the mean and the maximum, both taken over the entire sequence. Note that overall our reconstructed surface has a very low error and only a small region was not precisely reconstructed. Similar to many monocular reconstruction methods, the majority of our higher error regions are due to the monocular depth ambiguity (rows 1 and 3 in Figure 7; and row 4 in Figure 8). For further results, we also direct the reader to the video.

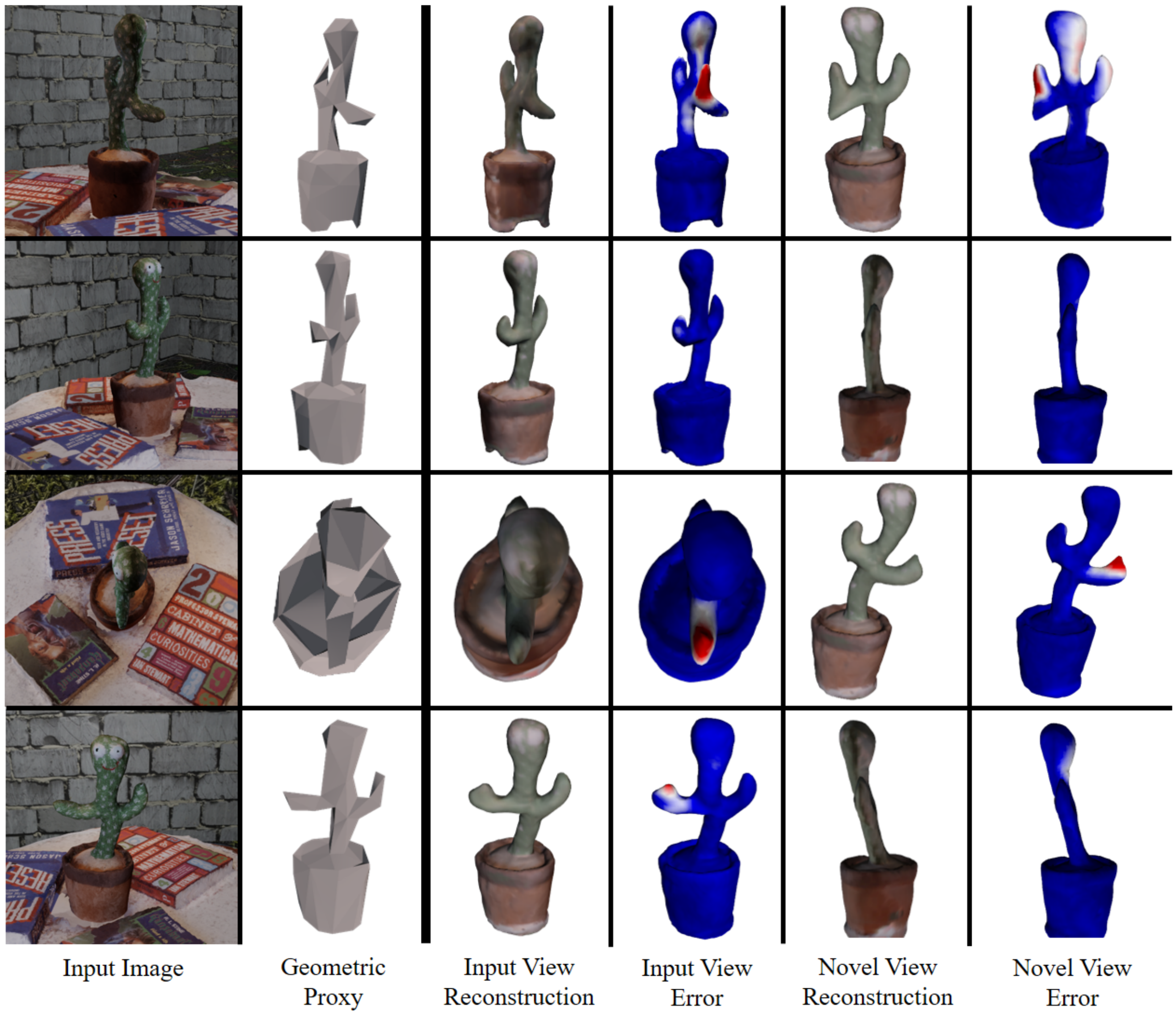


Figure 7: Additional results for our *Cactus* sequence. We include error coloring where blue is low error and red is high error, relative to the entire sequence.

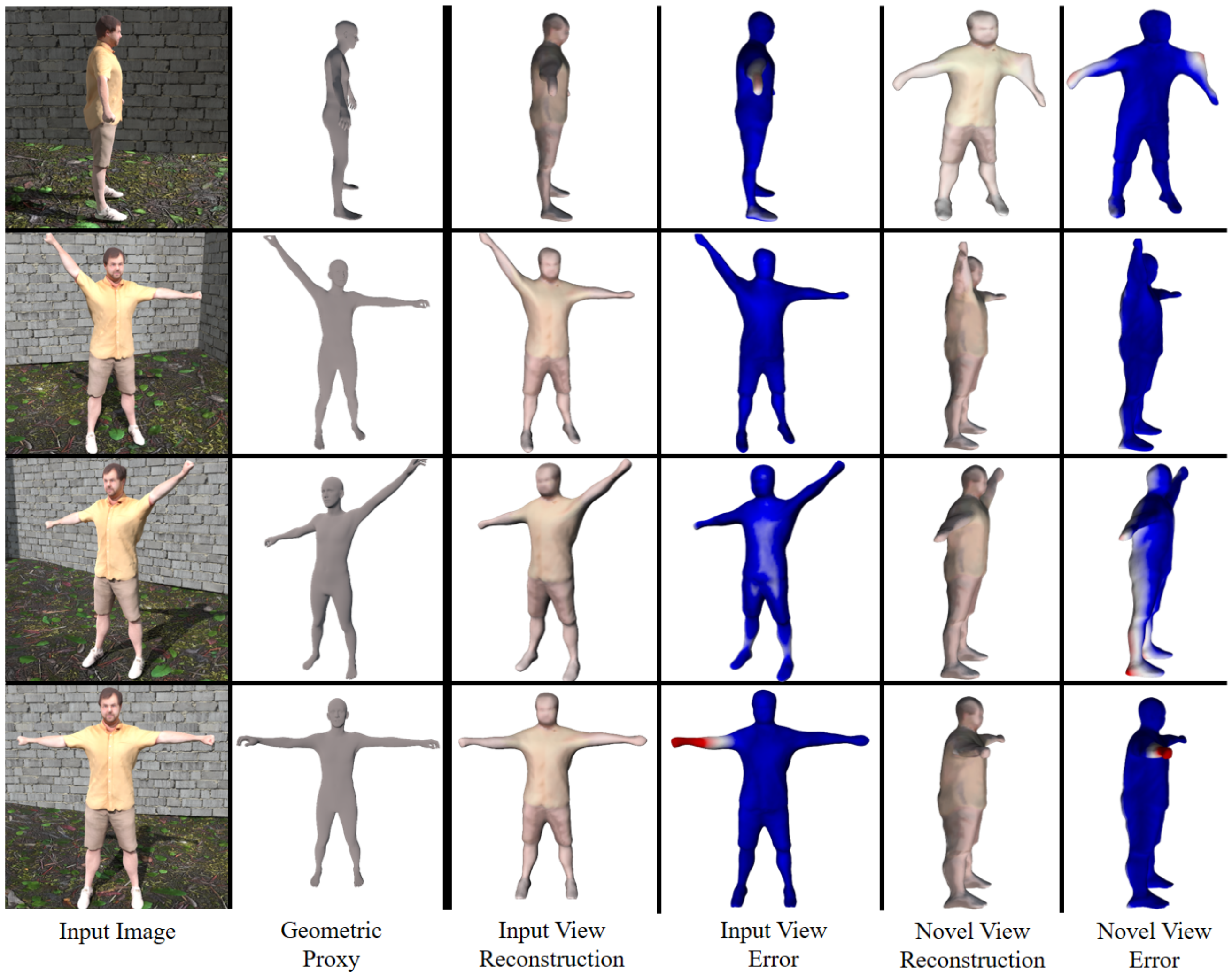


Figure 8: Additional results for our *RootTrans* sequence. We include error coloring where blue is low error and red is high error, relative to the entire sequence.

B. Implementation Details

We base our implementation on the codebase of Wang et al. [51], which is implemented in PyTorch [30]. Our method consists of three MLP networks: *Bending*, *SDF*, and *Rendering*. We provide a diagram showing the networks in Figure 9 and give additional details in Table 2. We also configure the starting weights of *SDF* using the geometric initialization method of Atzmon and Lipman [2].

At the start of training we follow Tretschk et al. [47] and initialize the latent codes with zeros. For each iteration during training, we select 512 pixels uniformly over the image for which to fire rays. We sample 64 positions along each straight ray, jitter these samples, and then importance sample 64 additional positions based on the SDF values.

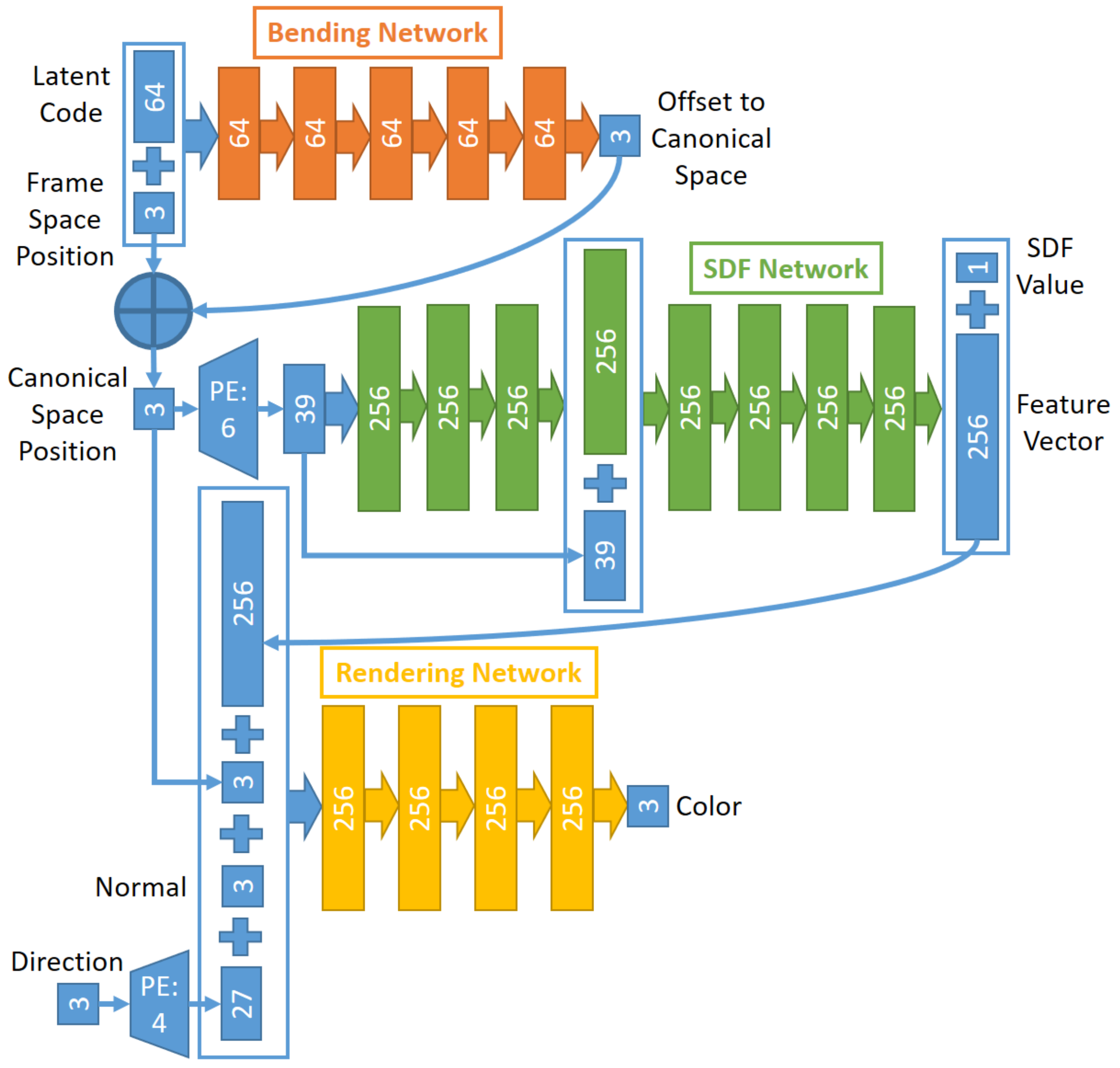


Figure 9: Network diagram of our method. PE denotes Positional Encoding [24] with the given number of additional frequencies. The addition node performs element-wise addition while the plus blocks represent vector concatenation.

Network	*Activation*	*Weight Normalization*
Bending	ReLU	No
SDF	SoftPlus ($\beta = 100$)	Yes
Rendering	ReLU	Yes

Table 2: Network parameters used in our implementation.

C. Derivation of the Discrete Opacity $\alpha_i^{(z)}$ Equation for Bent Rays

In this section we show that the derivation of discrete opacity $\alpha_i^{(z)}$ (Section 3.2) follows from volume rendering principles and the definition of the opaque density $\rho_i(\tilde{\boldsymbol{r}}(t))$ (Section 3.2, Equation 3). This analysis is similar to that in the Appendix A of Wang et al. [51], where we extend their derivation to any smooth parametric path.

Before beginning, we remind the reader that we render along bent rays, which are parametric paths in $\mathbb{R}^3$:

$$\tilde{\boldsymbol{r}}_i(t) = \boldsymbol{r}(t) + \boldsymbol{b}_i\big(\boldsymbol{r}(t)\big), \;\; \boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}. \tag{16}$$

We direct the reader to Section 3.1 for the definition of these terms. At each point on the bent ray there is an instantaneous viewing direction $\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}$ which can be computed analytically as:

$$\tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} = \tfrac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{r}(t)\big] + \tfrac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{b}_i(\boldsymbol{r}(t))\big] = \boldsymbol{d} + \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}\tfrac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{r}(t)\big] = \boldsymbol{d} + \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}\boldsymbol{d}, \tag{17}$$

where $\frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}$ is the Jacobian of the bending network w.r.t. its input $\boldsymbol{r}(t)$, a point along the straight ray. Note that, in our case, the Jacobian exists everywhere since the bending network is an MLP; thus our bent ray is a smooth parametric path.

In Section 3.2, we define the opaque density as:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \max\left\{\frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}, 0\right\}, \tag{18}$$

where Φ_s is the CDF of the logistic distribution. In order to proceed, we must expand the numerator through the chain rule:

$$\tfrac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big) = \phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\tfrac{\mathrm{d}}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t)\big) = \phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\big], \tag{19}$$

where ϕ_s is the Probability Density Function (PDF) of the logistic distribution. There is no need to expand the instantaneous viewing direction now that we have demonstrated its smoothness.

Placing (19) into (18) gives:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \max\left\{-\frac{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\big]}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}, 0\right\}. \tag{20}$$

There are two regions of interest identified in Appendix A of NeuS [51]: a ray entering geometry and a ray exiting geometry.

We first present the case where a ray is entering the geometry as depicted in Figure 10. Since we know that in this case:

$$\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} < 0 \tag{21}$$

and that both ϕ_s and Φ_s are non-negative, we can drop the maximum in the opaque density and return the numerator to its more condensed form:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \frac{-\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\Big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\Big]}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)} = \frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}. \tag{22}$$

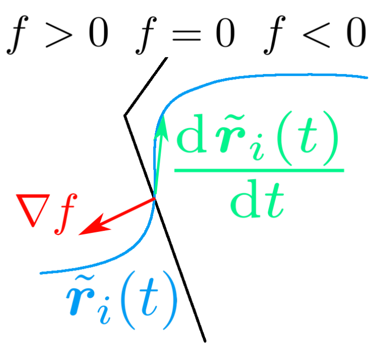

$f > 0$
$f = 0$
$f < 0$
$\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}$
∇f
$\tilde{\boldsymbol{r}}_i(t)$

Figure 10: Graphical depiction of a bent ray (traveling left to right) entering an SDF surface. Note that the instantaneous viewing direction and gradient of the SDF must have a negative dot product for the bent ray to be entering the geometry.

The remaining derivation for the discrete opacity $\alpha_i^{(z)}$ follows exactly as in Appendix A of NeuS [51]:

$$\begin{aligned}\alpha_i^{(z)} &= 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} \rho(t)\mathrm{d}t\right) = 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} \frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}\mathrm{d}t\right) = \\ &= 1 - \exp\left(\ln\left[\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z+1)}))\big)\right] - \ln\left[\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z)}))\big)\right]\right) = 1 - \frac{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z+1)}))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z)}))\big)} = \\ &= \frac{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z)}))\big) - \Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z+1)}))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^{(z)}))\big)}.\end{aligned} \tag{23}$$

Note that (23) is non-negative ($\because\ f(\tilde{\boldsymbol{r}}_i(t^{(z)}) > f(\tilde{\boldsymbol{r}}_i(t^{(z+1)})$ and Φ_s is non-negative and monotonically increasing) and as such is equivalent to a maximum with zero.

The second case to consider is where a ray is exiting the geometry as depicted in Figure 11. Given that

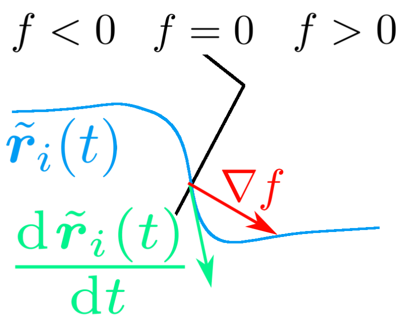
$$\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} > 0 \tag{24}$$

and that both ϕ_s and Φ_s are non-negative, (18) gives that $\rho_i(\tilde{\boldsymbol{r}}_i(t)) = 0$. Thus the discrete opacity $\alpha_i^{(z)}$ is:

$$\alpha_i^{(z)} = 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} \rho(t)\,\mathrm{d}t\right) = 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} 0\,\mathrm{d}t\right) = 0. \tag{25}$$

Since (23) will be non-positive when exiting the geometry ($\because\ f(\tilde{\boldsymbol{r}}_i(t^{(z+1)}) > f(\tilde{\boldsymbol{r}}_i(t^{(z)})$ and Φ_s is non-negative and monotonically increasing), we can write the derived equation for $\alpha_i^{(z)}$ satisfying both cases as:

$$\alpha_i^{(z)} = \max\left\{\frac{\Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z)})\big)\Big) - \Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z+1)})\big)\Big)}{\Phi_s\Big(f\big(\tilde{\boldsymbol{r}}_i(t^{(z)})\big)\Big)}, 0\right\}. \tag{26}$$



$f < 0$
$f = 0$
$f > 0$
$\tilde{\boldsymbol{r}}_i(t)$
∇f
$\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}$

Figure 11: Graphical depiction of a bent ray (traveling left to right) exiting an SDF surface. Note that the instantaneous viewing direction and gradient of the SDF must have a positive dot product for the bent ray to be exiting the geometry.

D. Unbiased Nature of our Rendering Method

In this section, we show that our rendering method is unbiased with respect to the surface of the object, i.e. $f\big(\tilde{\boldsymbol{r}}_i(t)\big) = 0$, given that s becomes sufficiently small. This demonstration follows a similar progression to that in the Appendix B of Wang et al. [51]. We assume two theoretical properties: that f is an SDF and that $\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}$ is never the zero vector. Both of these properties are enforced by penalizers in our method (i.e. the Eikonal and divergence regularizers respectively), but are not strictly guaranteed. Specifically, the $\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} \neq \mathbf{0}$ property follows from a divergence-free bending network since this prevents the compression of space necessary to give a stationary ray.

From Figure 10, it can be seen that for a smooth parametric path to intersect the surface, there *must* be a finite region $t \in (t_l, t_r)$ such that $\nabla f\big(\tilde{\boldsymbol{r}}_i(t)\big) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} < 0$. We can re-write the weight as:

$$\begin{aligned}
\omega(\tilde{\boldsymbol{r}}_i, t) &= T(\tilde{\boldsymbol{r}}_i, t)\,\rho(\tilde{\boldsymbol{r}}_i(t)) = \exp\left(-\int_0^t \rho(\tilde{\boldsymbol{r}}_i(\tau))\,\mathrm{d}\tau\right)\rho(\tilde{\boldsymbol{r}}_i(t)) = \\
&= \exp\left(-\int_0^{t_l} \rho(\tilde{\boldsymbol{r}}_i(\tau))\,\mathrm{d}\tau\right)\exp\left(-\int_{t_l}^{t} \rho(\tilde{\boldsymbol{r}}_i(\tau))\,\mathrm{d}\tau\right)\rho(\tilde{\boldsymbol{r}}_i(t)) = \\
&= T(\tilde{\boldsymbol{r}}_i, t_l)\exp\Big(\ln\big[\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\big] - \ln\big[\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t_l))\big)\big]\Big)\,\rho(\tilde{\boldsymbol{r}}_i(t)) = \\
&= T(\tilde{\boldsymbol{r}}_i, t_l)\frac{\cancel{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t_l))\big)}\frac{\Big[-\nabla f(\tilde{\boldsymbol{r}}_i(t))\cdot\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\Big]\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\cancel{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}}, \\
\therefore \omega(\tilde{\boldsymbol{r}}_i, t) &= \underbrace{\frac{T(\tilde{\boldsymbol{r}}_i, t_l)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t_l))\big)}}_{\text{constant}}\underbrace{\Big[-\nabla f(\tilde{\boldsymbol{r}}_i(t))\cdot\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\Big]\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}_{F(t)}\,.
\end{aligned} \tag{27}$$

Then we can establish that for:

$$F(t) = \underbrace{\Big[-\nabla f(\tilde{\boldsymbol{r}}_i(t))\cdot\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\Big]}_{G(t)}\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\,, \tag{28}$$

$\exists s > 0$ such that $F(t)$ is maximized by $f(\tilde{\boldsymbol{r}}_i(t^*)) = 0$, $t^* \in (t_l, t_r)$. Consider another value $t^\dagger \in (t_l, t_r)$, $t^\dagger \neq t^*$ where $G(t^\dagger) = 1$ is maximum and $G(t^*) = \epsilon$ is minimum for some necessarily non-zero value ϵ. This corresponds to the worst case for the unbiasedness since $0 < G(t) \leq 1, \forall t \in (t_l, t_r)$. Then:

$$G(t^*)\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^*))\big) \overset{?}{>} G(t^\dagger)\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big), \tag{29}$$

$$\frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} \overset{?}{>} \frac{G(t^\dagger)}{G(t^*)} = \frac{1}{\epsilon}\,. \tag{30}$$

Taking the limit of the left-hand side of (30) as s approaches 0 and using the definition of the logistic PDF ϕ_s:

$$\lim_{s\to 0} \frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} = \lim_{s\to 0} \frac{\exp\left(\frac{f(\tilde{\boldsymbol{r}}_i(t^\dagger))}{s}\right)}{4\left(1+\exp\left(-\frac{f(\tilde{\boldsymbol{r}}_i(t^\dagger))}{s}\right)\right)^2} = \infty \, . \tag{31}$$

Thus, for every possible ϵ, $\exists s > 0$ such that:

$$\frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} > \frac{1}{\epsilon} \, , \tag{32}$$

which implies $F(t^*) > F(t^\dagger)$, $\forall t^\dagger \in (t_l, t_r)$, $t^\dagger \neq t^*$. $\square$

E. Experimental Details

We present hyperparameters and additional details for the experiments with respect to our method and previous works. Sections E.1, E.2, E.3, and E.4 give the experimental details for Ub4D, LASR, DDD, and N-NRSfM, respectively. As a reminder, the datasets used are summarized in Table 3.

Name	*Creation*	*Frames*	*Resolution*	*Geometric Proxies?*	*GT?*
Cactus	Blender [8]	150	1024×1024	Yes (decimated GT)	Yes
RootTrans	Blender [8]	150	1024×1024	Yes (SMPL [31])	Yes
Lego	Blender [8]	150	800×800	No	No
Humanoid	Real World	171	960×1280	Yes (SMPL [31])	No
RealCactus	Real World	150	1080×1920	No	No

Table 3: Summary of the datasets introduced in this work. GT indicates access to ground truth geometry in the form of per-frame meshes. Synthetic scenes are above the dashed line, real-world captures below.

E.1. Ub4D

The hyperparameter settings for the experiments presented are contained in Table 4. Our loss weights are all relative to the colour weight:

$$L = L_{\text{COL}} + \omega_{\text{SEG}} L_{\text{SEG}} + \omega_{\text{EIK}} L_{\text{EIK}} + \omega_{\text{NBR}} L_{\text{NBR}} + \omega_{\text{DIV}} L_{\text{DIV}} + \omega_{\text{FLO}} L_{\text{FLO}} . \tag{33}$$

Additionally, for the *RealCactus* experiment we use constant ω_{NBR} and ω_{DIV} weights, rather than the $\frac{1}{100}$ factor exponentially increasing schedule of Tretschk et al. [47].

We give the time to apply marching cubes over the scene, which is independent of scene, in Table 5. This is given *without* applying frustum culling since that is an insignificant portion of the total runtime and is scene and region dependent.

Scene	*Iterations* (×1000)	*Training* (hours)	ω_{SEG}	ω_{EIK}	ω_{NBR}	ω_{DIV}	ω_{FLO}	λ_1	λ_2
Cactus	300	17.0	1.0	0.5	20000	200	10	700	75
RootTrans	450	26.4	1.0	0.5	20000	200	10	700	75
Lego	450	19.9	0.75	0.25	10000	100	0	-	-
RealCactus	450	22.1	1.5	0.75	$15000^\dagger$	$50^\dagger$	0	-	-
Humanoid	450	21.5	1.25	0.25	50000	200	10	700	75

Table 4: Hyperparameters used in acquiring results presented. "†" indicates a constant weight without the increasing schedule of Tretschk et al. [47].

Resolution	*March Time*	
	(seconds)	(hours)
64	14	0.00
128	69	0.02
256	536	0.15
512	4224	1.17
1024	34.99×10^6	18.32

Table 5: Time required to march geometry (without frustum culling).

E.2. LASR [53]

We run LASR [53] in the manner shown in their code[2]. We progressively increase the number of bones and faces in a coarse-to-fine manner following the configurations provided. This progression is shown in Table 6. For the *RootTrans* sequence, we use a slightly modified version of the code. This was to prevent a complete failure case where bone re-initialization without CNN re-initialization results in the mesh entering a local minimum that no longer reprojects on the image. This code modification was made with the assistance of the lead author of LASR [53].

Step	*Bones*	*Faces*	*Hypotheses*	*Epochs*
r1	21	1280	16	20
r2	26	1600	1	10
r3	31	1920	1	10
r4	31	2240	1	10
r5	36	2560	1	10
final	36	2880	1	10

Table 6: A subset of parameters used when running LASR [53].

E.3. Direct, Dense, Deformable [58]

We run the method of Yu et al. [58] in the manner shown in their code[3]. We empirically explored a set of values and found those of Table 7 to perform best when comparing results after rigid alignment with ICP [3] to the ground truth.

Parameter	*Value*
Photometric weight	1
ARAP weight	20

Table 7: A subset of parameters used when running the method of Yu et al. [58]. If not mentioned otherwise, we use the parameters as proposed in the original code.

E.4. Neural Dense NRSfM [43]

We run Neural NRSfM in the manner shown in their code[4]. In order to acquire the Multi-Frame Optical Flow (MFOF) W matrix used as input by this implementation, we use the Matlab [23] code of Ansari et al. [1]. Additionally, this implementation requires input in a specific format which is computed using proprietary code provided by the authors. The loss function weights used are given in Table 8.

[2]`https://github.com/google/lasr`
[3]`https://github.com/cvfish/PangaeaTracking`
[4]`http://vcai.mpi-inf.mpg.de/projects/Neural_NRSfM/`

Parameter	*Value*
β	1
γ	1×10^{-4}
η	1
λ	0

Table 8: A subset of parameters used when running the method of [43].

E.5. Human Geometry Proxy

To generate the proxy geometry for our human character sequences (*i.e.,* the *RootTrans* synthetic sequence and the *Humanoid* real-world sequence), we employ SMPLify-X [31] to obtain the root orientation and joint angles of SMPLX [31] human mesh model from the input image sequence. We then solve the 2D reprojection based optimization $\mathcal{L}_{2\text{D}}$ to obtain the 3D root translation of the human mesh:

$$\mathcal{L}_{2\text{D}} = \frac{1}{K} \sum_{k=1}^{K} \left\| \Pi(X_k) - p_k \right\|_2^2, \tag{34}$$

where $\Pi(\cdot)$ and K represents the perspective projection operator and the number of joints, respectively. X_k and p_k denotes the kth 3D joint keypoint obtained from [31], and pseudo GT 2D joint keypoints obtained from OpenPose [6]. To solve the optimization, we use Adam [16] optimizer with the camera intrinsics estimated by COLMAP [40], [41], and use a fixed height for the human model of 180 cm. Finally, we transform the vertices using the estimated camera extrinsics to place the model in world space.

F. Geometric Proxy Resolution Ablation

While our experiments have used a complete SMPLX model [31] for the *RootTrans* scene as described in Section E.5, we have identified that the geometric proxy could be reduced further. This is shown by the use of only a 12 vertex skeleton for the *Humanoid* scene. A proxy that summarizes the motion of the scene by coarsely tracking the extremities would be sufficient. We validate this possible approach in Figure 12 by reducing the SMPLX mesh to just 7 vertices (one on each extremity, two on the body, and one on the head). This 7 vertex proxy is sufficient to constrain Ub4D to produce a single canonical copy, rather than the multiple copies when no proxy is supplied (see Figure 5b in the main paper).

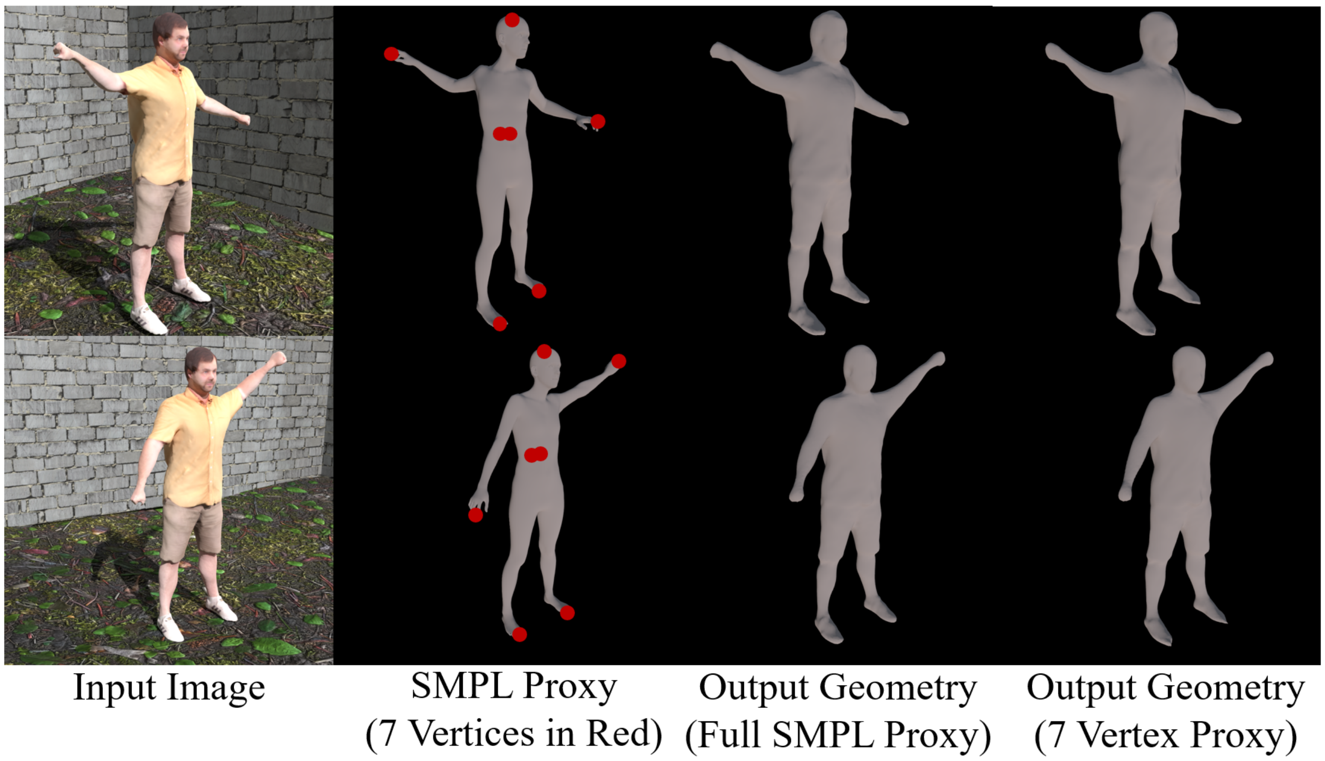


Figure 12: Comparison of the scene flow loss with full SMPL proxy ($10.4k$ vertices) and *just seven vertices* from it.

G. Per-Frame Latent Code Analysis and Novel Geometry Synthesis

In Ub4D, the entirety of the model's understanding of time is encoded into a per-frame latent code provided to the bending network. Initializing these latent codes with zeros gives our latent space a valuable property: a smooth, semantically meaningful latent representation. Demonstrating such a latent representation allows us to interpolate latent codes for certain applications, e.g. temporal super-resolution. It also opens the door for employing such deformation models using latent codes to analyze motion (e.g. periodicity detection, metrically comparing deformation states).

To validate the semantic meaning of our latent representation we perform PCA [32] on the 64 dimensional learned latent codes. The results are shown in Figure 13. Note that even though the latent space is never directly constrained in Ub4D, neighbouring frames (i.e. similar colors in Figure 13) tend to be nearby.

We wish to compare against the standard latent code initialization approach: random Gaussian initialization. However, the same concept of performing PCA [32] does not suffice. This is because PCA uses directions of maximum variance and randomly initialized latent codes could structure themselves "inside" of the variance. While a more complex dimensional reduction technique (e.g. t-SNE [48]) could yield results, a failure to visualize a meaningful structure would not definitively show that such a structure does not exist. Therefore, we use a reduced latent code dimension allowing visualization without dimensional projection.

Taking the *Cactus* scene, we train using 2D latent codes: once initializing with zeroes as proposed in Tretschk et al. [47] and once initializing with random Gaussian samples. We show the resulting learned latent codes in Figure 14. Note how spatially coherent the zero-initialized latent codes become during training, whereas the random Gaussian initialized latent codes do not have this property. Observing the particular structure of the zero-initialized case and slight clustering of similar frames in the random Gaussian initialization, one could imagine these latent codes as charged molecules, with similar states attracting and differing states repelling, resulting in a particular fold.

Some applications require semantically meaningful latent codes which we have demonstrated in the analysis above. This allows us to generate entirely new geometries by providing novel latent codes. Figure 15 shows samples of a novel latent path for the 2D latent codes from the left-hand side of Figure 14 (see webpage or video for a better visualization). A further investigation is required into the ability to generate new geometries from novel latent codes, particularly when using higher dimensional latent codes or exceeding the convex hull of the observations.

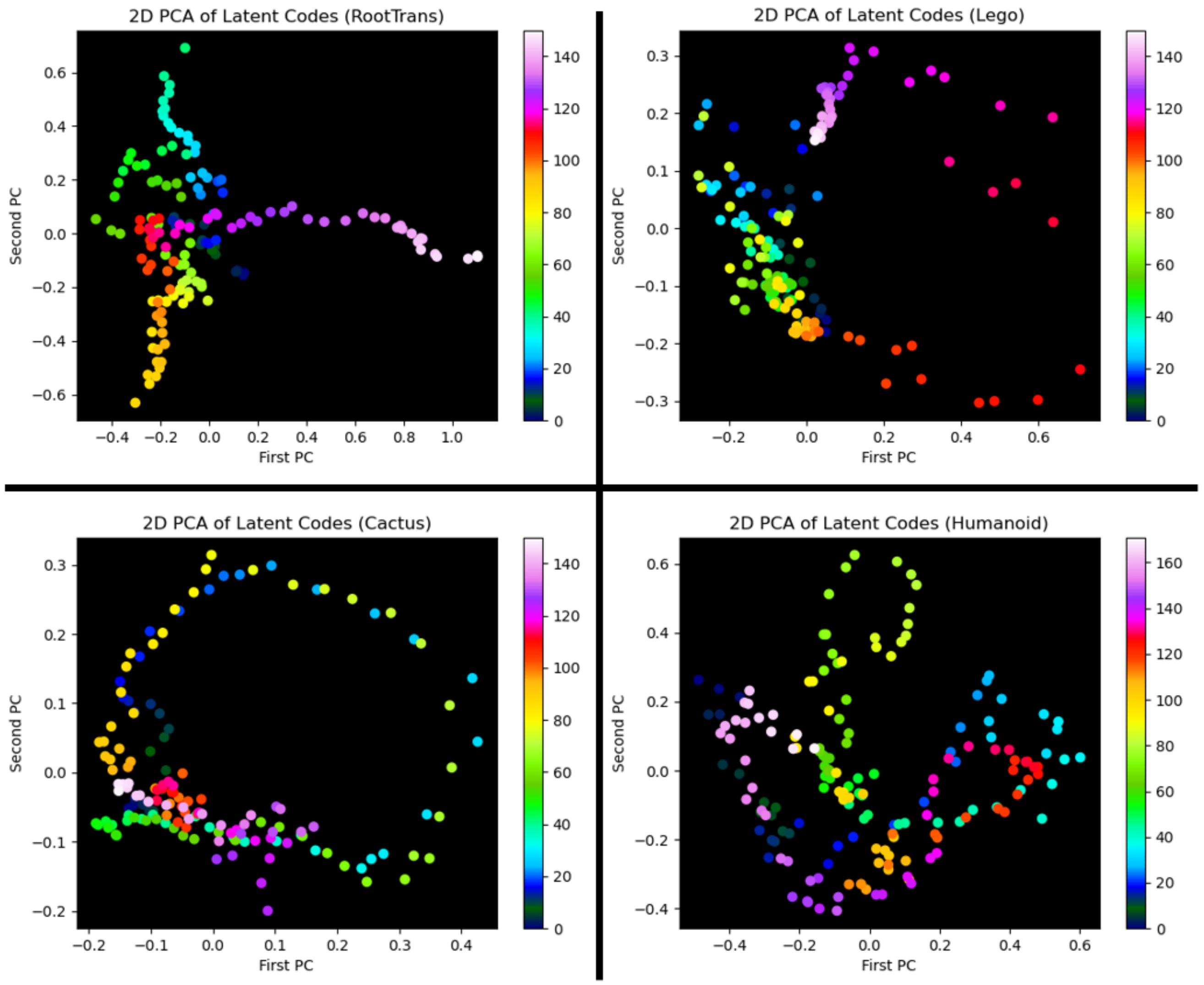


Figure 13: 2D PCA of learned 64D latent codes for the *RootTrans*, *Cactus*, *Lego*, and *Humanoid* scenes. The first two principal components explain 38%, 32%, 25%, and 24% of the variance, respectively. The colors correspond to frames. Note how similar colors are nearby.

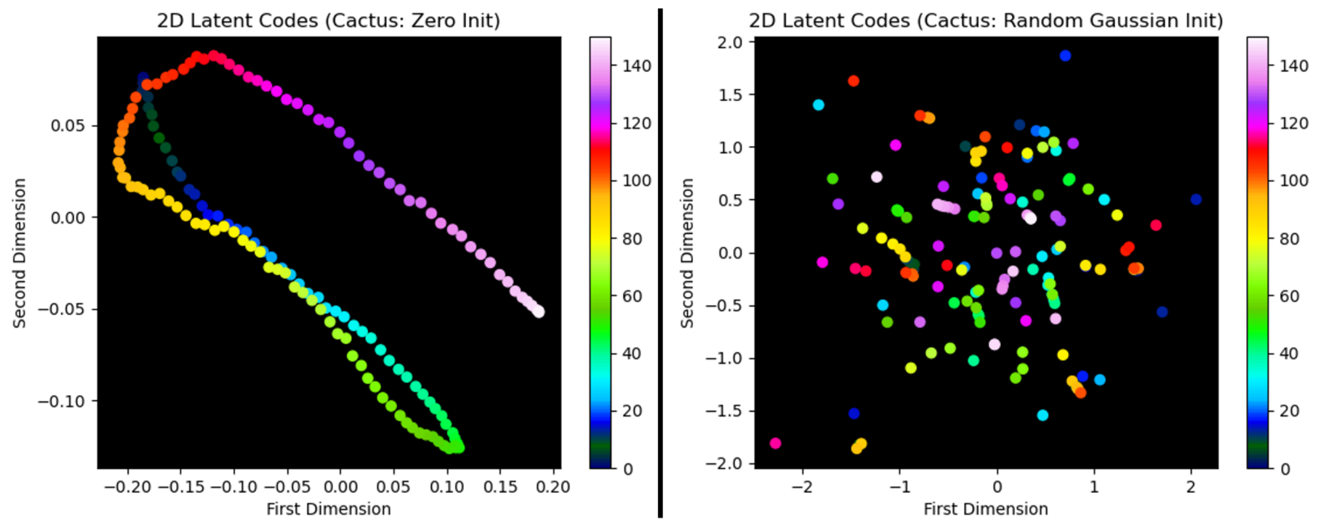


Figure 14: Learned 2D latent codes for the *Cactus* scene showing the latent code provided to the network without any dimensional projection. We initialize with zeroes on the left and random Gaussian samples on the right. The colors correspond to frames. Note how similar colors are nearby for zero-initialized latent codes, whereas the random Gaussian initialization does not give rise to such a property (although some local structuring is interesting).

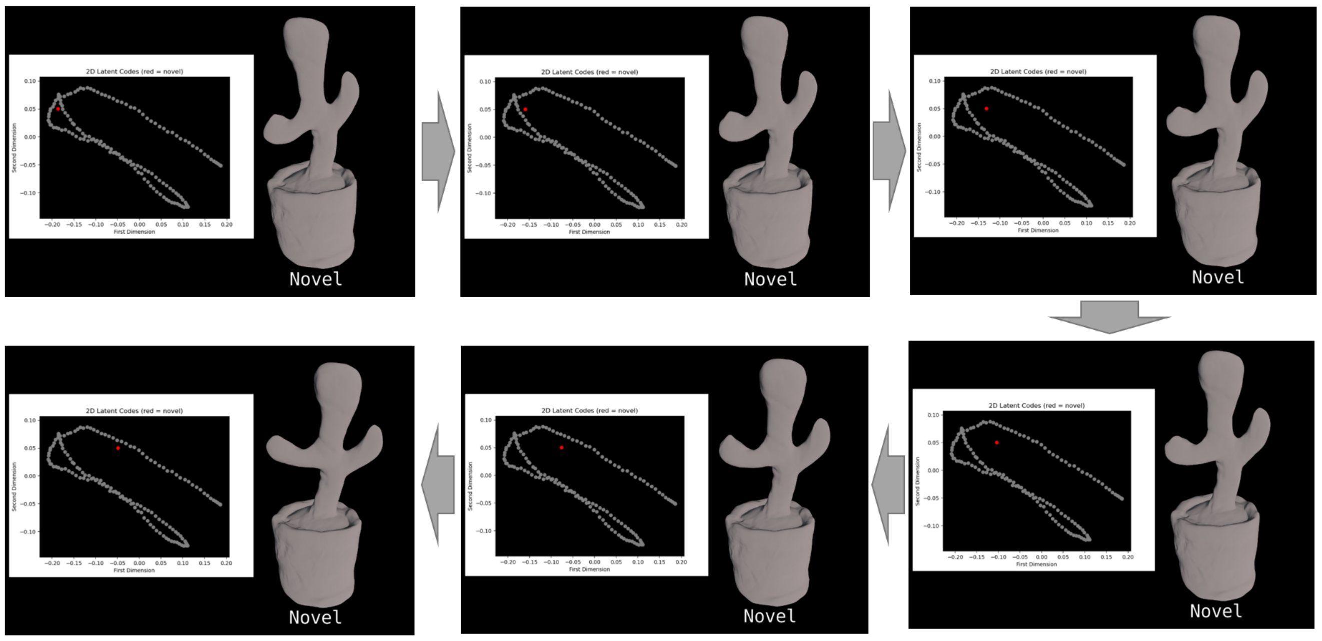


Figure 15: Synthesizing entirely new geometries with novel latent codes. Red dot in latent space plot shows the provided latent code while grey dots show the original sequence. Note the smoothness and plausibility of the deformation.

H. Additional Limitations

One limitation of Ub4D is that errors in the geometric proxy can increase the Chamfer distance of our reconstruction. Our scene flow loss is designed such that it does not require highly accurate correspondences to allow us to handle large deformations and prevent multiple canonical copies; however, we still inherit errors from the geometric proxy. Figure 16 illustrates this for a geometric proxy that is offset from the ground truth which results in the region with a high error on our reconstruction. Note that our method results in a decreased Chamfer distance for this frame compared to the geometric proxy (0.76 vs 0.92). Another limitation is that acquiring a geometric proxy may limit application if results without the scene flow loss are not satisfactory. Figure 17 shows one example of a common issue encountered without the scene flow loss. In this case, an additional appendage is used to satisfy the reconstruction losses while not grossly violating the other regularizers.

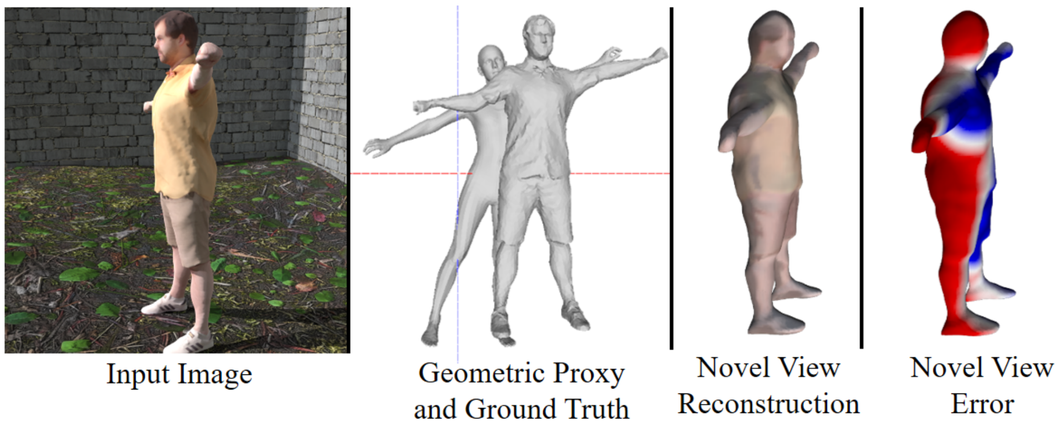


Figure 16: Impact of a significant geometric proxy error. Red regions have a higher Chamfer distance to the ground truth.

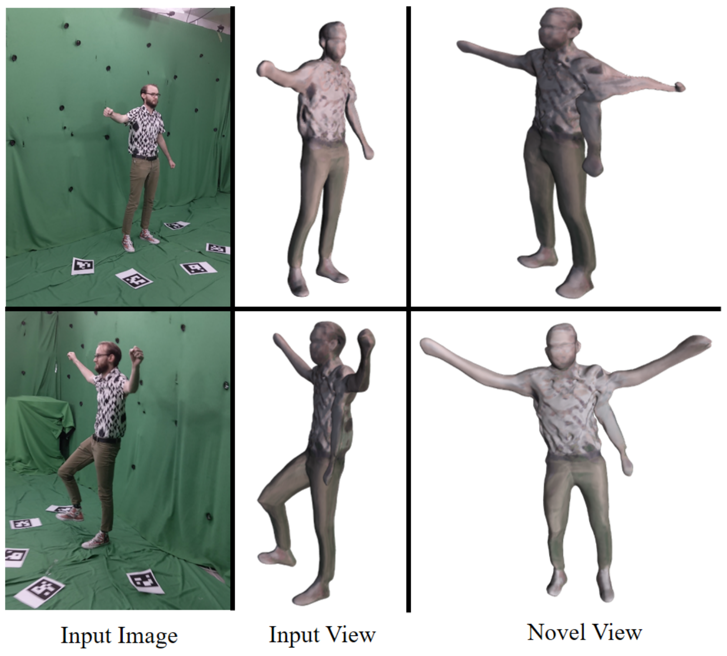


Figure 17: Example without the scene flow loss using an additional appendage to satisfy the reconstruction losses.