Unbiased 4D: Monocular 4D Reconstruction with a Neural Deformation Model – Supplemental Document –

Erik C.M. Johnson, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt
Max Planck Institute for Informatics, Saarbrücken, Germany
ecmjohnson@gmail.com, {mhaberma, sshimada, golyanik, theobalt}@mpi-inf.mpg.de

In this supplementary material, we present additional results in Section 1 and implementation details in Section 2. Section 3 provides a derivation of the discrete opacity equation for bent rays and Section 4 shows the unbiased nature of our rendering method. We provide a description of the metrics used in our quantitative analysis in Section 5. Additional details on the experiments performed are given in Section 6, including how we compute the geometric proxies for our human character scenes from the input images in Section 6.8. Section 7 investigates the necessary accuracy and resolution of the geometric proxies. Section 8 analyzes the learned latent codes and demonstrates novel geometry synthesis. Finally, we discuss some additional limitations in Section 9.

1. Additional Results

Figures 1 and 2 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 1; and row 4 in Figure 2). For further results, we also direct the reader to the video.

2. Implementation Details

We base our implementation on the codebase of Wang et al. [19], which is implemented in PyTorch [10]. Our method consists of 3 Multi-Layer Perceptron (MLP) networks: *Bending*, *SDF*, and *Rendering*. We provide a diagram showing the networks in Figure 3 and give additional details in Table 1. 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. [17] and initialize the latent codes with zeros. For each iteration during training, we select 512 pixels uniformly over the image

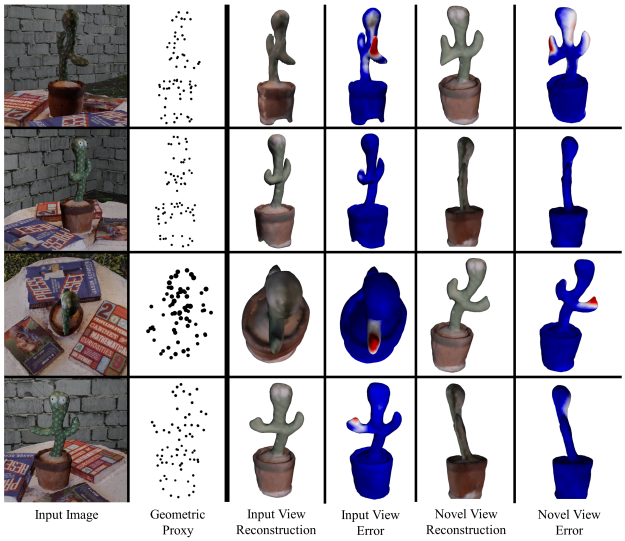

Figure 1. Additional results for our *Cactus* sequence. We include error coloring where blue is low error and red is high error, relative to the entire sequence.

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 Signed Distance Field (SDF) values.

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

Table 1. Network parameters used in our implementation.

3. 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

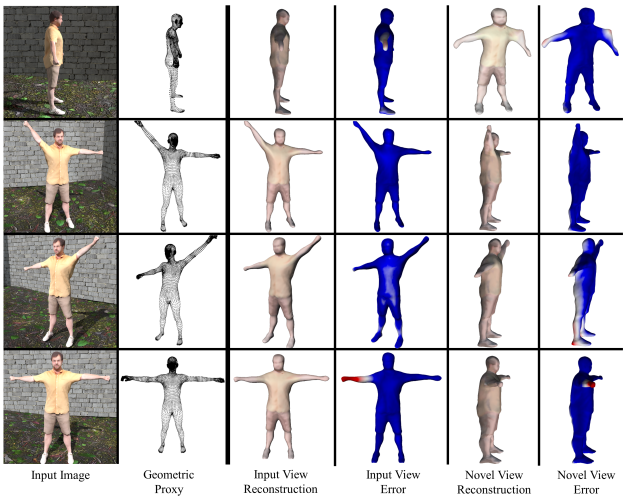

Figure 2. Additional results for our *RootTrans* sequence. We include error coloring where blue is low error and red is high error, relative to the entire sequence.

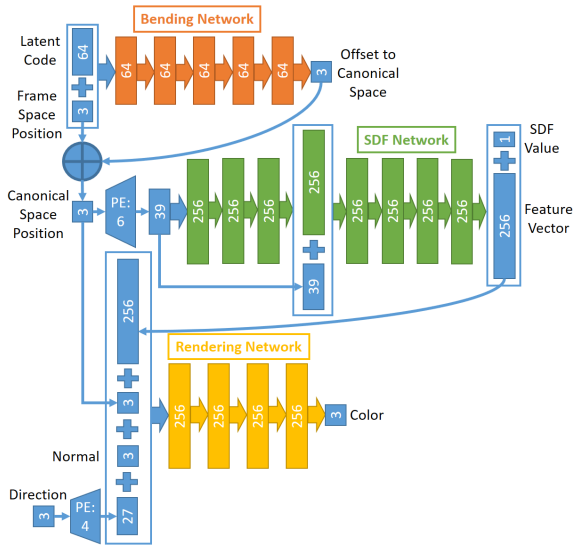

Figure 3. Network diagram of our method. PE denotes Positional Encoding [9] with the given number of additional frequencies. The addition node performs element-wise addition while the plus blocks represent vector concatenation.

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. [19], 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 $\mathbb{R}^3$:

$$\tilde{\boldsymbol{r}}_i(t) = \boldsymbol{r}(t) + \boldsymbol{b}_i\big(\boldsymbol{r}(t)\big), \quad \boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d} \tag{1}$$

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:

$$\frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{r}(t)\big] + \frac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{b}_i(\boldsymbol{r}(t))\big] \tag{2}$$

$$= \boldsymbol{d} + \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}\frac{\mathrm{d}}{\mathrm{d}t}\big[\boldsymbol{r}(t)\big] \tag{3}$$

$$= \boldsymbol{d} + \frac{\partial \boldsymbol{b}_i}{\partial \boldsymbol{r}(t)}\boldsymbol{d} \tag{4}$$

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:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \max\left\{\frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}, 0\right\} \tag{5}$$

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

$$\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big) = \phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\frac{\mathrm{d}}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t)\big) \tag{6}$$

$$= \phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\big] \tag{7}$$

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 Equation 7 into Equation 5, gives:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \tag{8}$$

$$\max\left\{-\frac{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\big]}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}, 0\right\} \tag{9}$$

There are two regions of interest identified in Appendix A of NeuS [19]: 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 4. Since we know that in this case:

$$\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} < 0 \tag{10}$$

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:

$$\rho_i(\tilde{\boldsymbol{r}}_i(t)) = \frac{-\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\Big[\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t}\Big]}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)} \tag{11}$$

$$= \frac{-\frac{\mathrm{d}\Phi_s}{\mathrm{d}t}\big(f(\tilde{\boldsymbol{r}}_i(t))\big)}{\Phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)} \tag{12}$$

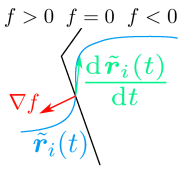

Figure 4. Graphical depiction of a bent ray (traveling left to right) entering an SDF surface. Note that the instantaneous viewing direction and gradient of the SDF must have a negative dot product for the bent ray to be entering the geometry.

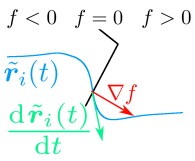

Figure 5. Graphical depiction of a bent ray (traveling left to right) exiting an SDF surface. Note that the instantaneous viewing direction and gradient of the SDF must have a positive dot product for the bent ray to be exiting the geometry.

The remaining derivation for the discrete opacity $\alpha_i^{(z)}$ follows exactly as in Appendix A of NeuS [19]:

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

Note that Equation 13 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 5. Given that:

$$\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \tfrac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} > 0 \tag{13}$$

and that both ϕ_s and Φ_s are non-negative, Equation 5 gives that $\rho_i(\tilde{\boldsymbol{r}}_i(t)) = 0$. Thus the discrete opacity $\alpha_i^{(z)}$ is:

$$\alpha_i^{(z)} = 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} \rho(t)\,\mathrm{d}t\right) \tag{14}$$

$$= 1 - \exp\left(-\int_{t^{(z)}}^{t^{(z+1)}} 0\,\mathrm{d}t\right) \tag{15}$$

$$= 0 \tag{16}$$

Since Equation 13 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:

$$\alpha_i^{(z)} = \tag{17}$$

$$\max\left\{\frac{\Phi_s\left(f\left(\tilde{\boldsymbol{r}}_i(t^{(z)})\right)\right) - \Phi_s\left(f\left(\tilde{\boldsymbol{r}}_i(t^{(z+1)})\right)\right)}{\Phi_s\left(f\left(\tilde{\boldsymbol{r}}_i(t^{(z)})\right)\right)}, 0\right\} \tag{18}$$

4. 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. $f\left(\tilde{\boldsymbol{r}}_i(t)\right) = 0$, given that s becomes sufficiently small. This demonstration follows a similar progression to that in the Appendix B of Wang et al. [19]. 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 4, 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 $\nabla f\left(\tilde{\boldsymbol{r}}_i(t)\right) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} < 0$. We can re-write the weight as:

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

Then we can establish that for:

$$F(t) = \underbrace{\left[-\nabla f(\tilde{\boldsymbol{r}}_i(t)) \cdot \frac{\mathrm{d}\tilde{\boldsymbol{r}}_i(t)}{\mathrm{d}t} \right]}_{G(t)} \phi_s\big(f(\tilde{\boldsymbol{r}}_i(t))\big)\,, \tag{19}$$

$\exists s > 0$ 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:

$$G(t^*)\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^*))\big) \overset{?}{>} G(t^\dagger)\,\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big) \tag{20}$$

$$\frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} \overset{?}{>} \frac{G(t^\dagger)}{G(t^*)} = \frac{1}{\epsilon}\,. \tag{21}$$

Taking the limit of the left-hand side of Equation (21) as s approaches 0 and using the definition of the logistic PDF ϕ_s:

$$\lim_{s\to 0} \frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} \tag{22}$$

$$= \lim_{s\to 0} \frac{\exp\left(\frac{f(\tilde{\boldsymbol{r}}_i(t^\dagger))}{s}\right)}{4\left(1+\exp\left(-\frac{f(\tilde{\boldsymbol{r}}_i(t^\dagger))}{s}\right)\right)^2} \tag{23}$$

$$= \infty\,. \tag{24}$$

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

$$\frac{\phi_s(0)}{\phi_s\big(f(\tilde{\boldsymbol{r}}_i(t^\dagger))\big)} > \frac{1}{\epsilon}\,, \tag{25}$$

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

5. Quantitative Metrics

This section presents the quantitative metrics used in our paper in detail. We use the Chamfer Distance (CD) as defined below:

$$\begin{aligned} \mathrm{CD}(E,G) &= \mathrm{E2G}(E,G) + \mathrm{G2E}(E,G) \\ &= \frac{1}{|E|}\sum_{\vec{x}\in E} \min_{\vec{y}\in G} ||\vec{x}-\vec{y}||_2^2 \\ &\quad + \frac{1}{|G|}\sum_{\vec{y}\in G} \min_{\vec{x}\in E} ||\vec{y}-\vec{x}||_2^2 \end{aligned} \tag{26}$$

where E is the estimated mesh and G is the ground truth mesh. In addition, we also report the unidirectional parts shown above individually due to the difference in methods to which we compare:

$$\mathrm{E2G}(E,G) = \frac{1}{|E|}\sum_{\vec{x}\in E} \min_{\vec{y}\in G} ||\vec{x}-\vec{y}||_2^2, \tag{27}$$

$$\mathrm{G2E}(E,G) = \frac{1}{|G|}\sum_{\vec{y}\in G} \min_{\vec{x}\in E} ||\vec{y}-\vec{x}||_2^2. \tag{28}$$

As mentioned in the paper, the CD metric is not always fair for each method which is why we also report the E2G and G2E values separately. The E2G metric is unfair for the novel view synthesis methods, that is NR-NeRF [17] and D-NeRF [13]. This is because we must select a region in which to employ marching cubes for generating geometry and this selection affects the E2G metric. The G2E metric is unfair for N-NRSfM [16] since this method only reconstructs the visible surface of the object.

6. Experimental Details

We present hyperparameters and additional details for the experiments with respect to our method and previous works. Sections 6.1, 6.4, 6.6, and 6.7 give the experimental details for Ub4D, LASR, DDD, and N-NRSfM, respectively. As a reminder, the datasets used are summarized in Table 2.

6.1. Ub4D

The hyperparameter settings for the experiments presented are contained in Table 3. Our loss weights are all relative to the colour weight:

$$\begin{aligned} L = L_{\text{COL}} &+ \omega_{\text{SEG}} L_{\text{SEG}} + \omega_{\text{EIK}} L_{\text{EIK}} + \omega_{\text{NBR}} L_{\text{NBR}} \\ &+ \omega_{\text{DIV}} L_{\text{DIV}} + \omega_{\text{FLO}} L_{\text{FLO}} \end{aligned} \tag{29}$$

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. [17].

We give the time to apply marching cubes over the scene, which is independent of scene, in Table 4. This is given *without* applying frustum culling since that is an insignificant portion of the total runtime and is scene and region dependent.

6.2. NR-NeRF [17]

We run NR-NeRF [17] in the manner shown in their code[1]. We use the default parameters provided by the authors and sweep the threshold to find the threshold giving minimal metric (5 for the *Cactus* scene and 10 for the *RootTrans* scene). We cannot fairly consider the estimate to ground truth metric since NR-NeRF produces surface crossings throughout the volume, thus we crop the total volume to a slightly (10%) expanded bounding box of the ground truth.

6.3. D-NeRF [13]

We run D-NeRF [13] in the manner shown in their code[2]. We use the default parameters provided by the authors and sweep the threshold to find the threshold giving minimal

[1] https://github.com/facebookresearch/nonrigid_nerf

[2] https://github.com/albertpumarola/D-NeRF

Name	*Creation*	*Frames*	*Resolution*	*Geometric Proxies?*	*GT?*
Cactus	Blender [5]	150	1024×1024	Yes (decimated GT)	Yes
RootTrans	Blender [5]	150	1024×1024	Yes (SMPL [11])	Yes
Lego	Blender [5]	150	800×800	No	No
Humanoid	Real World	171	960×1280	Yes (SMPL [11])	No
RealCactus	Real World	150	1080×1920	No	No

Table 2. 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.

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†	50†	0	-	-
Humanoid	450	21.5	1.25	0.25	50000	200	10	700	75

Table 3. Hyperparameters used in acquiring results presented. † indicates a constant weight without the increasing schedule of Tretschk et al. [17].

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 4. Time required to march geometry (without frustum culling).

metric (400 for the *Cactus* scene and 400 for the *RootTrans* scene). Note that D-NeRF has failed to produce reasonable geometry for the *RootTrans* scene, which we hypothesize is due to the large camera motion. We cannot fairly consider the estimate to ground truth metric for the same reason discussed for D-NeRF in Section 6.3. The same region cropping to a slightly (10%) expanded bounding box of the ground truth.

Note that it appears that D-NeRF [13] has failed for the *RootTrans* scene. We also try different thresholds (specifically, 40 which is the default in the D-NeRF code) and run the scene 6 times: 3 of these produce no geometry at all for all tried thresholds while 3 produce some geometry not accurately representing the *RootTrans* scene.

6.4. LASR [20]

We run LASR [20] in the manner shown in their code[3]. 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 5. 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 [20].

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 5. A subset of parameters used when running LASR [20].

6.5. ViSER [21]

We run ViSER [21] in the manner shown in their code[4]. For the initialization phase, we use the frames ranges shown

[3] https://github.com/google/lasr
[4] https://github.com/gengshan-y/viser-release

in Table 6. The selection of these frame ranges was done in accordance with the guidance that "the viewpoint coverage is large enough" [21].

Dataset	*Start frame*	*End frame*
Cactus	10	40
RootTrans	10	40

Table 6. Frame ranges used for initialization when running ViSER [21].

6.6. Direct, Dense, Deformable [22]

We run the method of Yu et al. [22] in the manner shown in their code[5]. 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. [22]. If not mentioned otherwise, we use the parameters as proposed in the original code.

6.7. Neural Non-Rigid Structure-from-Motion (NRSfM) [16]

We run Neural NRSfM in the manner shown in their code[6]. In order to acquire the Multi-Frame Optical Flow (MFOF) W matrix used as input by this implementation, we use the Matlab [7] 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.

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

Table 8. A subset of parameters used when running the method of [16].

[5] https://github.com/cvfish/PangaeaTracking

[6] http://vcai.mpi-inf.mpg.de/projects/Neural_NRSfM/

6.8. 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 [11] to obtain the root orientation and joint angles of SMPLX [11] human mesh model from the input image sequence. We then solve the 2D reprojection based optimization $\mathcal{L}_{\text{2D}}$ to obtain the 3D root translation of the human mesh:

$$\mathcal{L}_{\text{2D}} = \frac{1}{K} \sum_{k=1}^{K} \left\| \Pi(X_k) - p_k \right\|_2^2 , \tag{30}$$

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 [11], and pseudo GT 2D joint keypoints obtained from OpenPose [4]. To solve the optimization, we use Adam [6] optimizer with the camera intrinsics estimated by COLMAP [14], [15], 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.

7. Geometric Proxy Resolution Ablation

While our experiments have used a complete SMPLX model [11] for the *RootTrans* scene as described in Section 6.8, 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. This could allow the use of skeleton tracking systems (e.g. [8]) rather than methods with dense mesh output like SMPLify-X [11]. We validate this possible approach in Figure 6 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 5(b) in the main paper).

8. 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).

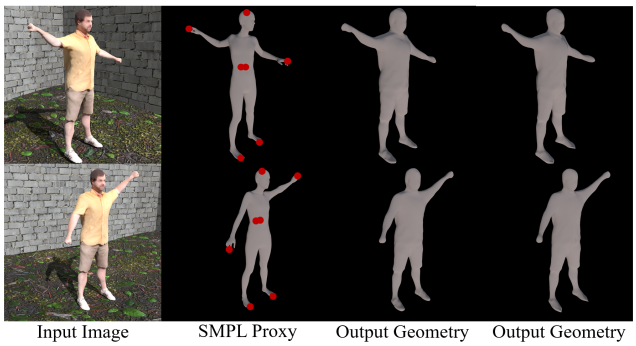

Figure 6. Comparison of the scene flow loss with full SMPL proxy ($10.4k$ vertices) and *just seven vertices* from the SMPL proxy.

To validate the semantic meaning of our latent representation we perform PCA [12] on the 64 dimensional learned latent codes. The results are shown in Figure 7. Note that even though the latent space is never directly constrained in Ub4D, neighbouring frames (i.e. similar colors in Figure 7) tend to be nearby.

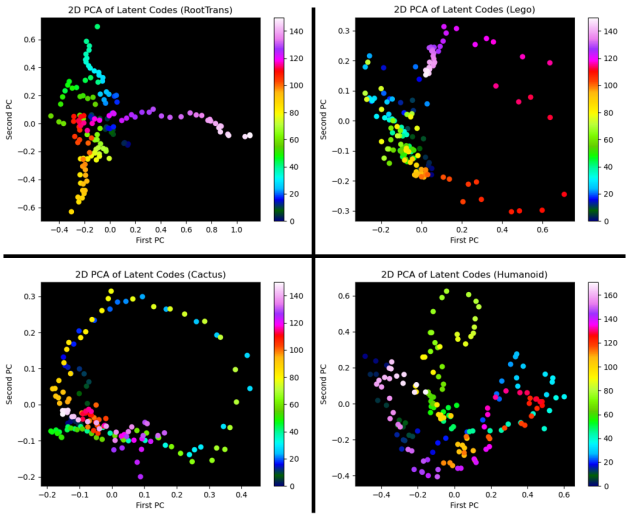

Figure 7. 2D PCA of learned 64D latent codes for the *RootTrans*, *Cactus*, *Lego*, and *Humanoid* scenes. The first two principal components explain 38%, 32%, 25%, and 24% of the variance, respectively. The colors correspond to frames. Note how similar colors are nearby.

We wish to compare against the standard latent code initialization approach: random Gaussian initialization. However, the same concept of performing PCA [12] 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 [18]) 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. [17] and once initializing with random Gaussian samples. We show the resulting learned latent codes in Figure 8. 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.

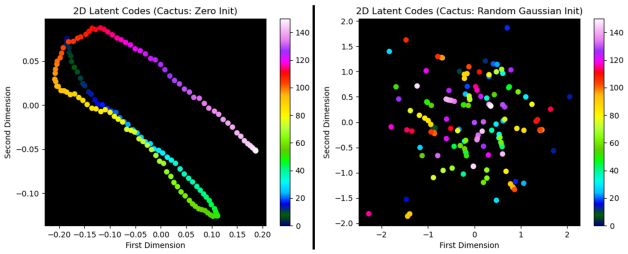

Figure 8. Learned 2D latent codes for the *Cactus* scene showing the latent code provided to the network without any dimensional projection. We initialize with zeroes on the left and random Gaussian samples on the right. The colors correspond to frames. Note how similar colors are nearby for zero-initialized latent codes, whereas the random Gaussian initialization does not give rise to such a property (although some local structuring is interesting).

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 9 shows samples of a novel latent path for the 2D latent codes from the left-hand side of Figure 8 (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.

9. 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.

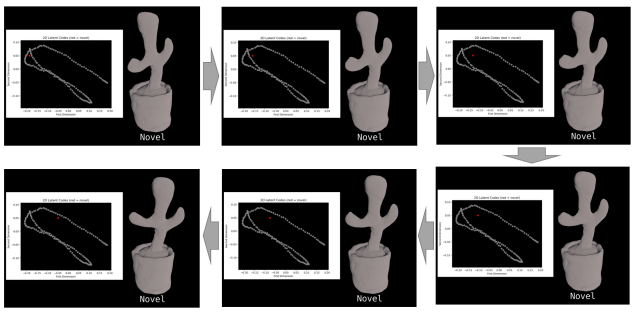

Figure 9. Synthesizing entirely new geometries with novel latent codes. Red dot in latent space plot shows the provided latent code while grey dots show the original sequence. Note the smoothness and plausibility of the deformation.

Figure 10 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 11 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 of our method.

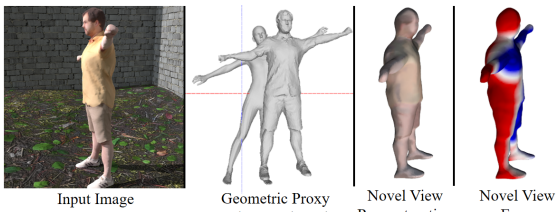

Figure 10. Impact of a significant geometric proxy error. Red regions have a higher Chamfer distance to the ground truth.

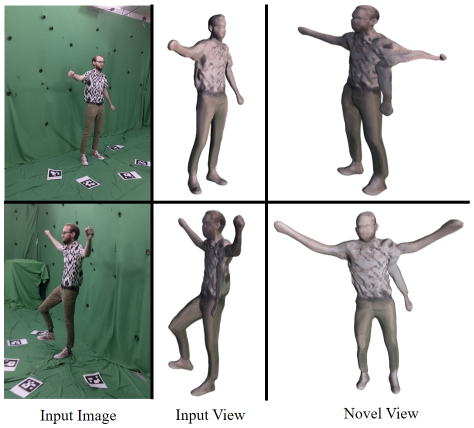

Figure 11. Example of our method without the scene flow loss using an additional appendage to satisfy the reconstruction losses.

References

[1] Mohammad D. Ansari, Vladislav Golyanik, and Didier Stricker. Scalable dense monocular surface reconstruction. In *International Conference on 3D Vision (3DV)*, 2017. 6

[2] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[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. 6

[4] 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. 6

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2018. 5

[6] 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. 6

[7] MATLAB. *R2020a*. The MathWorks Inc., Natick, Massachusetts, 2010. 6

[8] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. In *ACM Transactions on Graphics (TOG)*, 2017. 6

[9] 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

[10] 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. 1

[11] 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. 5, 6

[12] 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. 7

[13] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 4, 5

[14] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[15] 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. 6

[16] 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. 4, 6

[17] 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. 1, 4, 5, 7

[18] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. In *Journal of Machine Learning Research (JMLR)*, 2008. 7

[19] 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. 1, 2, 3

[20] 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. 5

[21] 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. 5, 6

[22] 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. 6