# NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction – Supplemental Document –


Yiming Wang*
University of Pennsylvania, Peking University
wym12416@pku.edu.cn

Qin Han*
Peking University
hanqin@pku.edu.cn

Marc Habermann
Max Planck Institute for Informatics
mhaberma@mpi-inf.mpg.de

Kostas Daniilidis
University of Pennsylvania
kostas@cis.upenn.edu

Christian Theobalt
Max Planck Institute for Informatics
theobalt@mpi-inf.mpg.de

Lingjie Liu
University of Pennsylvania, Max Planck Institute for Informatics
lingjie.liu@seas.upenn.edu


## Supplementary Material

In the following, we provide more details about our method. First, we present the derivation of Eqs. 5 and 6 (Sec. A), and the proof of Theorem 1 (Sec. B). Second, we introduce the dataset (Sec. C) we used in the experiments and show more quantitative and qualitative results (Sec. D) to further demonstrate the performance of our model. Finally, we provide implementation details of our method (Sec. E).

## A. Derivation of Equation 5 and Equation 6

We calculate the second-order derivatives of the hash table parameters $\Omega$ as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Omega} &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} \frac{\partial \frac{\partial d}{\partial \mathbf{x}}}{\partial \Omega} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} \frac{\partial (\frac{\partial d}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \mathbf{x}})}{\partial \Omega} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} (\frac{\partial \mathbf{e}}{\partial \mathbf{x}} \frac{\partial \frac{\partial d}{\partial \mathbf{e}}}{\partial \Omega} + \frac{\partial d}{\partial \mathbf{e}} \frac{\partial \frac{\partial \mathbf{e}}{\partial \mathbf{x}}}{\partial \Omega}) \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} (\frac{\partial \mathbf{e}}{\partial \mathbf{x}} \frac{\partial \frac{\partial d}{\partial \mathbf{e}}}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \Omega} + \frac{\partial d}{\partial \mathbf{e}} \frac{\partial \frac{\partial \mathbf{e}}{\partial \mathbf{x}}}{\partial \Omega})\end{aligned} \tag{1}$$

∗ Equal contribution

and the SDF network parameters $\Theta$ as

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Theta} &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} \frac{\partial \frac{\partial d}{\partial \mathbf{x}}}{\partial \Theta} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} \frac{\partial (\frac{\partial d}{\partial \mathbf{e}} \frac{\partial \mathbf{e}}{\partial \mathbf{x}})}{\partial \Theta} \\ &= \frac{\partial \mathcal{L}}{\partial \mathbf{n}} (\frac{\partial \mathbf{e}}{\partial \mathbf{x}} \frac{\partial \frac{\partial d}{\partial \mathbf{e}}}{\partial \Theta} + \frac{\partial d}{\partial \mathbf{e}} \frac{\partial \frac{\partial \mathbf{e}}{\partial \mathbf{x}}}{\partial \Theta})\end{aligned} \tag{2}$$

## B. Derivation of Theorem 1

First we recall the Definition 1.

**Definition 1** *Given a ReLU based MLP $f$ with $L$ hidden layers taking $x \in \mathbb{R}^d$ as input, it computes the output $y = H_L g(H_{L-1} \dots g(H_1 x))$, where $H_l \in \mathbb{R}^{n_l} \times \mathbb{R}^{n_{l-1}}$, $l \in \{1, \dots, L\}$ is the layer index, and $g$ is the ReLU function. We define $P_l^j \in \mathbb{R}^{n_{l-1}} \times \mathbb{R}^1$ and $S_l^i \in \mathbb{R}^1 \times \mathbb{R}^{n_l}$ as*

$$\begin{aligned}P_l^j &= G_l H_{l-1} \dots G_2 H_1^{(\cdot,j)} \\ S_l^i &= H_L^{(i,\cdot)} G_L \dots H_{l+1} G_{l+1}\end{aligned} \tag{3}$$

*where $H_1^{(\cdot,j)}$ is the $j$th column of $H_1$, $H_L^{(i,\cdot)}$ is the $i$th row of $H_L$, and* $G_l = \begin{cases} 1, H_{l-1} \dots g(H_1 x) > 0 \\ 0, \text{ otherwise} \end{cases}$.

Now the second-order derivatives of a ReLU-based MLP with respect to its input and intermediate layers can be defined as follows.

**Theorem 1 (Second-order derivative of ReLU-based MLP)** *Given a ReLU based MLP $f$ with $L$ hidden layers with*

*the same definition in Definition 1. The second-order derivative of the MLP $f$ is:*

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l} = (P_l^j S_l^i)^T, \qquad \frac{\partial^2 y}{\partial \mathbf{x}^2} = 0 \tag{4}$$

*where $\frac{\partial y}{\partial x}_{(i,j)}$ is the matrix element (i,j) of $\frac{\partial y}{\partial x}$, and $S_l^i$ and $P_l^j$ are defined in Definition 1.*

**Proof** First, by applying the chain rule, we have

$$\begin{aligned}
\frac{\partial y}{\partial x} &= \frac{\partial H_L g(H_{L-1}P_{L-1})}{\partial x} \\
&= H_L \frac{\partial g(H_{L-1}P_{L-1})}{\partial x} \\
&= H_L \frac{\partial g}{\partial H_{L-1}P_{L-1}} H_{L-1} \frac{\partial P_{L-1}}{\partial x} \\
&= H_L G_L H_{L-1} \frac{\partial P_{L-1}}{\partial x} \\
&= H_L G_L H_{L-1} G_{L-1} H_{L-2} \frac{\partial P_{L-2}}{\partial x} \\
&= H_L G_L H_{L-1} \cdots G_2 H_1
\end{aligned} \tag{5}$$

And, the element $(i,j)$ of $\frac{\partial y}{\partial x} \in \mathbb{R}^{n_L} \times \mathbb{R}^{n_1}$ is

$$\frac{\partial y}{\partial x}_{(i,j)} = H_L^{(i,\cdot)} G_L \cdots G_2 H_1^{(\cdot,j)}. \tag{6}$$

We can then calculate

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l} = (\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l P_l^j})^\top \frac{\partial H_l P_l^j}{\partial H_l}. \tag{7}$$

On the one hand, since $P_l^j$ is independent of $H_l$, we have

$$\frac{\partial H_l P_l^j}{\partial H_l} = (P_l^j)^\top. \tag{8}$$

On the other hand, we have

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l P_l^j} = \frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_{L-1}P_{L-1}^j} \frac{\partial H_{L-1}P_{L-1}^j}{\partial H_{L-2}P_{L-2}^j} \cdots \frac{\partial H_{l+1}P_{l+1}^j}{\partial H_l P_l^j}. \tag{9}$$

we can further derive that

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_{L-1}P_{L-1}^j} = H_L^{(i,\cdot)} G_L, \tag{10}$$

and

$$\begin{aligned}
\frac{\partial H_l P_l^j}{\partial H_{l-1}P_{l-1}^j} &= \frac{\partial H_l G_l H_{l-1} P_{l-1}^j}{\partial H_{l-1}P_{l-1}^j} \\
&= H_l G_l + H_l \frac{\partial G_l}{\partial H_{l-1}P_{l-1}^j} \\
&= H_l G_l,
\end{aligned} \tag{11}$$

since $\frac{\partial G_l}{\partial H_{l-1}P_{l-1}^j} = 0$. Thus we have

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l P_l^j} = H_L^{(i,\cdot)} G_L H_{L-1} \cdots H_{l+1} G_{l+1} = S_l^i. \tag{12}$$

Consequently,

$$\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l} = (\frac{\partial \frac{\partial y}{\partial x}_{(i,j)}}{\partial H_l P_l^j})^\top \frac{\partial H_l P_l^j}{\partial H_l} = (S_l^i)^\top (P_l^j)^\top. \tag{13}$$

Second, we define $x_l \in \mathbb{R}^{n_l}$, $l \in \{1, \ldots, L-1\}$ as $x_l = g(H_{l-1}x_{l-1})$, specifically $x_1 = g(H_1 x)$. Then we have

$$y = H_L x_{L-1}. \tag{14}$$

Then we calculate

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = \frac{\partial \frac{\partial y}{\partial x_1}\frac{\partial x_1}{\partial x}}{\partial x} = \frac{\partial \frac{\partial y}{\partial x_1}}{\partial x}\frac{\partial x_1}{\partial x} + \frac{\partial \frac{\partial x_1}{\partial x}}{\partial x}\frac{\partial y}{\partial x_1}. \tag{15}$$

Note that $\frac{\partial \frac{\partial x_1}{\partial x}}{\partial x} = 0$. So we have

$$\begin{aligned}
\frac{\partial^2 y}{\partial \mathbf{x}^2} &= \frac{\partial \frac{\partial y}{\partial x_1}}{\partial x}\frac{\partial x_1}{\partial x} \\
&= \frac{\partial^2 y}{\partial x_1^2}(\frac{\partial x_1}{\partial x})^2 \\
&= \frac{\partial^2 y}{\partial x_{L-1}^2}(\frac{\partial x_{L-1}}{\partial x_{L-2}})^2 \cdots (\frac{\partial x_1}{\partial x})^2.
\end{aligned} \tag{16}$$

Since $y = H_L x_{L-1}$, we have $\frac{\partial^2 y}{\partial x_{L-1}^2} = 0$. Thus we get

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = 0 \tag{17}$$

## C. Dataset

**Dataset for Static Scene Reconstruction.** For static scene reconstruction, we use 15 scenes from the DTU dataset [5], same as those used in NeuS [13], with a wide variety of materials, appearance and geometry. These scenes involve challenging cases for reconstruction algorithms, such as non-Lambertian surfaces and fine structures. Each scene contains 49 or 64 images with an image resolution of $1600 \times 1200$. We split each scene into training and testing parts following NeuS [13]. Specifically, we set images indexed at 8, 13, 16, 21, 26, 31, 34 and 56 if available for testing and the other images for training. We train and test each scene with foreground masks provided by IDR [15].

**Dataset for Dynamic Synthetic Scene Reconstruction.** We use three synthetic scenes with various types of deformations and motions to evaluate our method. The Lego scene is shared by NeRF [7] in form of Blender [1] file. We transform the Lego object into different poses and positions

and render images at the resolution of $400 \times 400$ in Blender. The Lego scene contains 150 frames with 40 training camera views and 40 test camera views. The human scene is provided by RenderPeople[2]. We render images at the resolution of $512 \times 512$ following [10], and the whole sequence contains 100 frames with 48 camera views for training and 12 camera views for testing. The Lion scene is shared by Artemis [6], which has 177 frames with 30 camera views for training and 6 camera views for testing. **Dataset for Dynamic Real Scene Reconstruction.** We select three sequences from the Dynacap dataset [4], denoted as D1, D2 and D3, for real-scene reconstruction. These sequences are captured under a dense camera setup at a resolution of $1285 \times 940$. We crop the images with a 2D bounding box which is estimated from the foreground masks to obtain the target images at a resolution of $512 \times 512$. For each sequence, we choose 500 frames containing large movements for evaluation to show our advantages (D1: 17,760 to 18,260, D2: 6,095 to 6,595, D3: 3,450 to 3,950). The D1 sequence has 94 camera views, from which we pick 9 camera views (7, 17, 27, 37, 47, 57, 67, 77, 87) for testing and the rest of the views for training. The D2 sequence has 50 camera views, from which we pick 5 camera views (7, 17, 27, 37, 47) for testing and the rest of the views for training. The D3 sequence has 101 camera views, from which we pick 10 camera views (7, 17, 27, 37, 47, 57, 67, 77, 87, 97) for testing and the rest of the views for training. We train and test each scene with foreground masks provided by Dynacap [4]. We choose a sequence of a human walking around in the D1 sequence, which contains 200 frames, to conduct the ablation study.

# D. Additional Result

**Video Results.** We provide a supplementary video to better demonstrate the qualitative results of our method. We highly encourage the readers to check our video.

**Static Scene Reconstruction.** In Tab. 1, we provide the per-scene breakdown analysis of the quantitative comparisons on the DTU dataset presented in the main paper (Tab. 1). We also present additional qualitative comparisons on the DTU dataset in Fig. 1. Additionally, we provide a quantitative comparison with a concurrent work, Voxurf [14], in Tab. 2 under its experimental settings.

**Dynamic Scene Reconstruction.** In Tab. 3, we present the quantitative comparison between our method and Instant-NGP [8] for synthetic scenes.

**Effect of the accuracy of Global Transformation Prediction.** For our global Transformation Prediction compoment, a few minor inaccuracies will not have a significant impact on the final result, since our incremental training strategy can further fine-tune the model parameters to compensate for these errors. To illustrate this statement, we conduct an experiment on one sequence in the DynaCap. We use the $R$ and $T$ of the SMPL obtained by EasyMocap as ground truth, and then add different levels of noises to get different accuracy levels of $R$ and $T$. The results in Fig. 2 demonstrate that our model is rather robust for the predicted rotation $R$ and transition $T$. The reconstruction performance only drops when the predicted transition and rotation are very inaccurate.

# E. Implementation Details

## E.1. Baselines

**COLMAP [11, 12].** We directly refer to COLMAP's results on the DTU dataset reported in NeuS[13].

**NeuS [13].** The Chamfer Distance scores of NeuS shown in the paper are directly referred to the results reported in the original paper. The geometry reconstruction results are produced using the officially released pre-trained models with mask supervision. The PSNR scores and novel view synthesis results are obtained by training the officially released code on the DTU training dataset with mask supervision, and testing it on the DTU testing dataset.

**Instant-NGP [8].** We use the officially released code to train the model on the DTU dataset for 50k iterations. The training takes about 5 minutes. For dynamic scenes, we train the model separately for each frame from scratch, and limit the training time to 20 seconds which is consistent with our method.

**Instant-NSR [16].** The results on DTU dataset are provided by the authors.

**Voxurf [14].** The results on DTU dataset are refered to the original paper.

**D-NeRF [9].** We use the officially released code to train the model for 800k iterations. The training time of D-NeRF on real scenes is longer than on synthetic scenes, 50 and 20 hours, respectively. This is because the length of real sequences is longer than that of the synthetic sequences, which are around 500 frames and 150 frames, respectively. Moreover the number of camera views for real scenes is greater than for synthetic scenes. For long sequences with dense camera views, the model cannot upload all the images at once due to the GPU memory limitation, so extra time is needed to load the images during training.

**TiNeuVox [3].** We use the officially released code and train the model for 80k and 150k iterations on synthetic scenes and real scenes, respectively. Due to the same reason mentioned above, extra time is needed during the training on real scenes. The training time of TiNeuVox on real scenes is longer than on synthetic scenes, 3 and 1 hours, respectively.

https://github.com/Totoro97/NeuS
https://github.com/NVlabs/instant-ngp
https://github.com/albertpumarola/D-NeRF
https://github.com/hustvl/TiNeuVox

Table 1. Quantitative comparisons on the DTU dataset. We color code the **best** and **second best** results. Our method outperforms other baselines for geometry reconstruction regarding the Chamfer Distance (CD) and is on par with Instant-NGP of novel view synthesis in terms of PSNR.

| | COLMAP | NeuS | | Instant-NGP | | Instant-NSR | | Ours | |
|---|---|---|---|---|---|---|---|---|---|
| Runtime | 1h | 8 h | | 5 min | | 8.5 min | | 5 min | |
| ScanID | CD ↓ | PSNR↑ | CD ↓ | PSNR↑ | CD↓ | PSNR↑ | CD↓ | PSNR↑ | CD↓ |
| scan24 | 0.81 | 26.49 | 0.83 | 28.32 | 1.68 | 23.86 | 2.86 | 28.44 | 0.56 |
| scan37 | 2.05 | 26.17 | 0.98 | 27.19 | 1.93 | 24.97 | 2.81 | 27.14 | 0.76 |
| scan40 | 0.73 | 27.66 | 0.56 | 30.45 | 1.57 | 25.3 | 2.09 | 29.70 | 0.49 |
| scan55 | 1.22 | 27.78 | 0.37 | 29.81 | 1.16 | 25.43 | 0.81 | 29.67 | 0.37 |
| scan63 | 1.79 | 30.63 | 1.13 | 31.22 | 2.00 | 29.52 | 1.65 | 31.75 | 0.92 |
| scan65 | 1.58 | 27.42 | 0.59 | 27.78 | 1.56 | 26.17 | 1.39 | 27.83 | 0.71 |
| scan69 | 1.02 | 25.83 | 0.60 | 24.79 | 1.81 | 22.93 | 1.47 | 24.84 | 0.76 |
| scan83 | 3.05 | 30.00 | 1.45 | 31.23 | 2.33 | 26.72 | 1.67 | 31.24 | 1.22 |
| scan97 | 1.40 | 26.40 | 0.95 | 26.96 | 2.16 | 25.94 | 2.47 | 26.86 | 1.08 |
| scan105 | 2.05 | 29.63 | 0.78 | 30.62 | 1.88 | 27.71 | 1.12 | 30.57 | 0.63 |
| scan106 | 1.00 | 25.87 | 0.52 | 25.62 | 1.76 | 23.12 | 1.22 | 26.05 | 0.59 |
| scan110 | 1.32 | 28.82 | 1.43 | 28.6 | 2.32 | 25.44 | 2.30 | 28.93 | 0.89 |
| scan114 | 0.49 | 28.80 | 0.36 | 29.5 | 1.86 | 26.7 | 0.98 | 28.98 | 0.40 |
| scan118 | 0.78 | 27.36 | 0.45 | 27.91 | 1.80 | 25.13 | 1.41 | 27.82 | 0.48 |
| scan122 | 1.17 | 31.19 | 0.45 | 32.93 | 1.72 | 28.19 | 0.95 | 32.48 | 0.55 |
| mean | 1.36 | 28.00 | 0.77 | 28.86 | 1.84 | 25.81 | 1.68 | 28.82 | 0.70 |

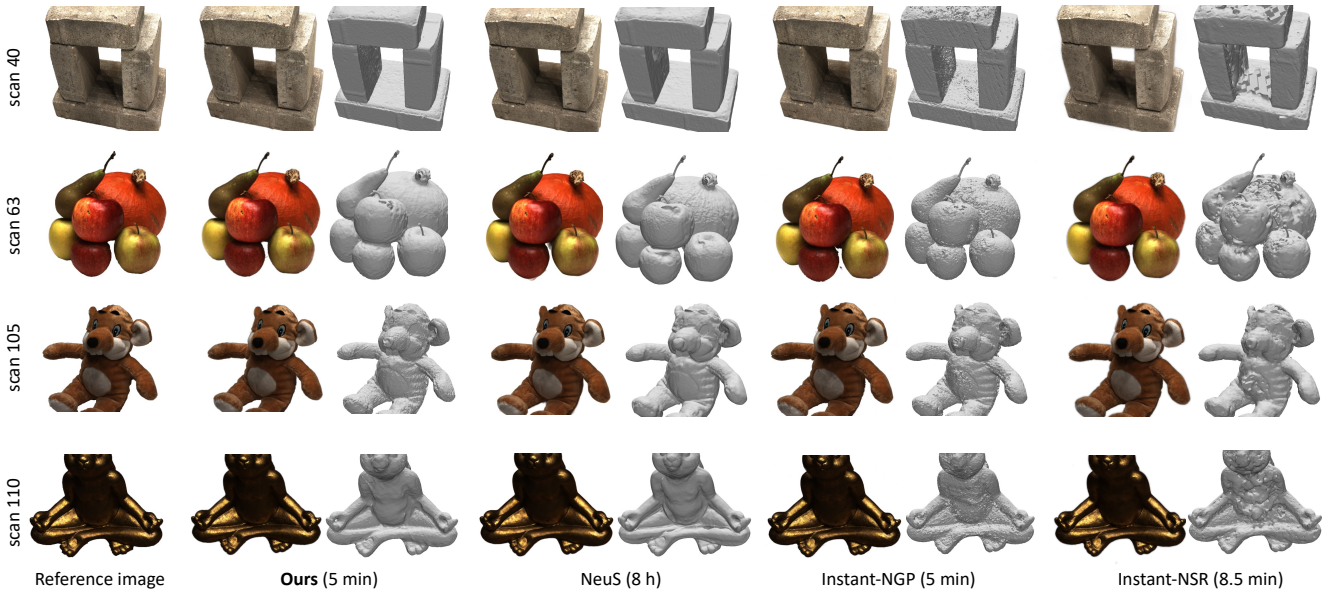


Figure 1. Qualitative comparisons on the DTU dataset for static scene geometry reconstruction and novel view synthesis. Our method demonstrates high-quality rendering quality, superior to NeuS and Instant-NSR, and comparable to Instant-NGP in terms of complex texture reconstruction. In addition, it outperforms all the baselines regarding 3D geometry reconstruction, with fine details without inducing noise.

Table 2. Quantitative comparison between our method and Voxurf [14], using the DTU dataset and the experimental settings employed by Voxurf.

| | Voxurf | | Ours | |
|---|---|---|---|---|
| Runtime | 16 min | | **5 min** | |
| ScanID | PSNR↑ | CD↓ | PSNR↑ | CD↓ |
| scan24 | 27.89 | 0.65 | **28.44** | **0.56** |
| scan37 | **26.90** | **0.74** | 26.53 | 0.76 |
| scan40 | 28.81 | **0.39** | **29.70** | 0.49 |
| scan55 | 31.02 | **0.35** | **31.47** | 0.37 |
| scan63 | **34.38** | 0.96 | 33.74 | **0.92** |
| scan65 | **31.48** | **0.64** | 30.99 | 0.71 |
| scan69 | **30.13** | 0.85 | 28.77 | **0.76** |
| scan83 | **37.43** | 1.58 | 36.78 | **1.22** |
| scan97 | **28.35** | **1.01** | 28.24 | 1.08 |
| scan105 | 32.94 | 0.68 | **33.30** | **0.63** |
| scan106 | **34.17** | 0.60 | 33.91 | **0.59** |
| scan110 | 32.70 | 1.11 | **34.50** | **0.89** |
| scan114 | 30.97 | **0.37** | **31.14** | 0.40 |
| scan118 | **37.24** | **0.45** | 37.17 | 0.48 |
| scan122 | **37.97** | **0.47** | 37.41 | 0.55 |
| mean | **32.16** | 0.72 | 32.14 | **0.70** |

| | Instant-NGP | | **Ours** | |
|---|---|---|---|---|
| Dataset | PSNR↑ | CD↓ | PSNR↑ | CD↓ |
| Lego | 29.05 | 37.19 | **29.5** | **17.1** |
| Lion | 33.09 | - | **33.60** | - |
| Human | **36.18** | 4.48 | 33.20 | **1.86** |
| Runtime | 1h | | 1h | |

Table 3. Quantitative comparison between our method and Instant-NGP [8] on synthetic scenes. The Chamfer Distance of Lion sequence is omitted since the ground truth geometry is not provided.

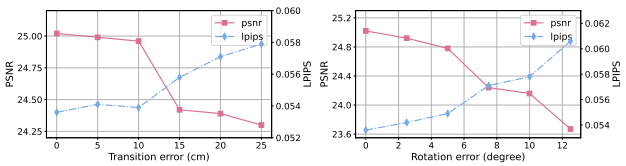


Figure 2. Influence of the predicted transformation accuracy to the reconstruction quality. We use the rotation $R$ and transition $T$ of SMPL as ground truth, and then add different levels of noise for different levels of accuracy.

## E.2. Network Architecture

As shown in Fig 3, the network architecture of *NeuS2* consists of the following components: (a) a multi-resolution hash grid with 14 levels of different resolutions ranging from 16 to 2048; (b) an SDF network modeled by a 1-layer MLP with 64 hidden units; (c) an RGB network modeled by a 2-layer MLP with 64 hidden units.

## E.3. Training Details

**Unbiased Volume Rendering.** To render an image, we apply the unbiased volume rendering of NeuS [13]. That is, we first transform the signed distance field into a volume density field $\phi_s(f(\mathbf{x}))$, where $\phi_s(x) = se^{-sx}/(1+e^{-sx})^2$ is the *logistic density distribution*, which is the derivative of the Sigmoid function $\Phi_s(x) = 1/(1+e^{-sx})$ and $s$ is a learnable parameter. Next, we construct an unbiased weight function in the volume rendering equation. Specifically, for each pixel of an image, we sample $n$ points $\{\mathbf{p}(\mathbf{t_i}) = \mathbf{o} + t_i\mathbf{v} | i = 0, 1, \ldots, n-1\}$ along its camera ray, where $\mathbf{o}$ is the center of the camera and $\mathbf{v}$ is the view direction. By accumulating the SDF-based densities and colors of the sample points, we can compute the color $\hat{C}$ of the ray with the same approximation scheme as used in NeRF [7] as

$$\hat{C}(\mathbf{o}, \mathbf{v}) = \sum_{i=0}^{n-1} T(t_i)\alpha(t_i)c(\mathbf{p}(t_i), \mathbf{v}) \tag{18}$$

where $T(t_i)$ is the discrete accumulated transmittance defined by $T(t_i) = \prod_{j=0}^{i-1}(1-\alpha(t_j))$, and $\alpha(t_i)$ is a discrete density value defined by

$$\alpha(t_i) = \max\left(\frac{\Phi_s(f(p(t_i))) - \Phi_s(f(p(t_{i+1})))}{\Phi_s(f(p(t_i)))}, 0\right). \tag{19}$$

As the rendering process is differentiable, we can learn the signed distance field $f$ and the radiance field $c$ from the multi-view images.

**Ray Marching Strategy.** We adopt a ray marching acceleration strategy used in Instant-NGP[8]. That is, we maintain an occupancy grid that roughly marks each voxel grid as empty or non-empty. The occupancy grid can effectively guide the marching process by preventing sampling in empty spaces and, thus, accelerate the volume rendering process. We periodically update the occupancy grid based on the SDF value predicted by our model. In detail, we first use the SDF value $d$ to calculate the density value $\phi_s(d)$ for each grid, where $\phi_s(x) = se^{-sx}/(1+e^{-sx})^2$ is the *logistic density distribution* and $s$ is a learnable parameter. We then use the density value to update the occupancy grid in the same way as Instant-NGP[8]. Additionally, to achieve faster convergence, we sample 90% rays on the foreground pixels and 10% rays on the background pixels.

**Hyperparameters.** For static scene reconstruction, we train our model for 15k iterations, which takes around 5

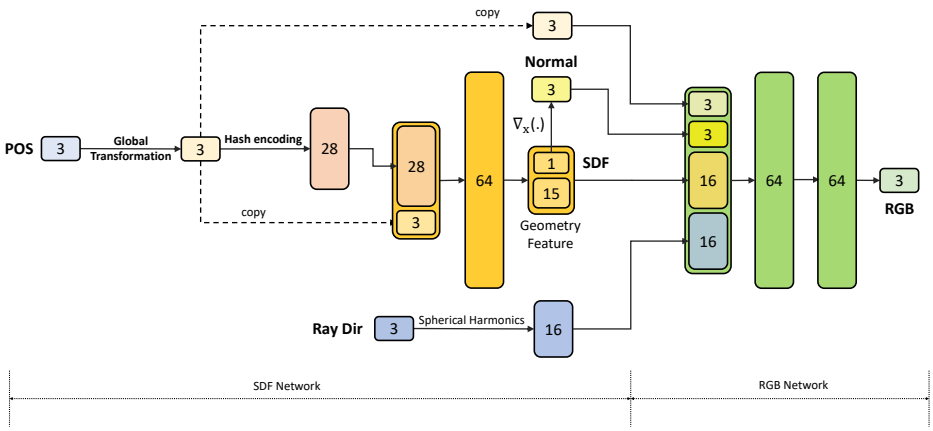


Figure 3. A visualization of the network architecture of *NeuS2*.

minutes. For dynamic scene reconstruction, we train the first frame from scratch for 2k iterations (4k iterations in particular for the Lego sequence due to its complex geometry), which takes about 40 seconds; then train each subsequent frame for 1.1k iterations, where we optimize the global transformation independently for the first 100 iterations and we fine-tune the network parameters and global deformation together for the remaining iterations.

## References


[1] Blender. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/. 2

[2] Renderpeople. https://renderpeople.com/3d-people/, 2018. 3

[3] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. *arxiv:2205.15285*, 2022. 3

[4] Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Real-time deep dynamic characters. *ACM Transactions on Graphics (TOG)*, 40(4):1–16, 2021. 3

[5] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014. 2

[6] Haimin Luo, Teng Xu, Yuheng Jiang, Chenglin Zhou, QIwei Qiu, Yingliang Zhang, Wei Yang, Lan Xu, and Jingyi Yu. Artemis: Articulated neural pets with appearance and motion synthesis. *arXiv preprint arXiv:2202.05628*, 2022. 3

[7] 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*, pages 405–421. Springer, 2020. 2, 5

[8] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*, 2022. 3, 5

[9] 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. 3

[10] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 3

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[12] 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. 3

[13] 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. *NeurIPS*, 2021. 2, 3, 5

[14] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction, 2022. 3, 5

[15] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502, 2020. 2

[16] Fuqiang Zhao, Yuheng Jiang, Kaixin Yao, Jiakai Zhang, Liao Wang, Haizhao Dai, Yuhui Zhong, Yingliang Zhang, Minye Wu, Lan Xu, et al. Human performance modeling and rendering via neural animated mesh. *arXiv preprint arXiv:2209.08468*, 2022. 3