Appendix A Weight Function

We follow the NeuS [61] paper to set the weight function $w(t)$ in Equation 1. First, a density function $\rho(t)$ is defined as:

$$\rho(t) = \max\left(\frac{-\frac{d\Phi_h}{dt}(s(\mathbf{p}(t)))}{\Phi_h(s(\mathbf{p}(t)))},\ 0\right) \tag{7}$$

where $\Phi_h(x) = (1 + e^{-hx})^{-1}$ is the Sigmoid function, and $w(t)$ can be constructed as:

$$w(t) = T(t)\rho(t), \text{ where } T(t) = \exp\left(-\int_0^t \rho(u)du\right). \tag{8}$$

Appendix B Additional Implementation Details

The Color Network (MLP) consists of 4 dense layers, each with 256 hidden dimensions, while the SDF Network has 8 dense layers of 256 hidden dimensions, the Motion Network has 5 dense layers of 64 hidden dimensions, and the Rigidity Network has 3 dense layers of 32 hidden dimensions.

Similar to NeRF++ [73], we use a separate NeRF network to render far-away points outside of a unit-sphere region of interest. The Motion and Rigidity networks are not applied to those regions since we assume the background to be static. If we do not disable these networks on the background, we observe significant jittering on those regions.

We have also tried enabling and disabling viewing direction as an input to the Color Network. It turns out that disabling viewing direction can result in smoother geometry, as it prevents over-fitting to the viewing direction as discussed in NeRF++. However, we also observed that disabling the viewing direction led to incorrect geometries (like holes) in some cases.

Appendix C Potential Applications

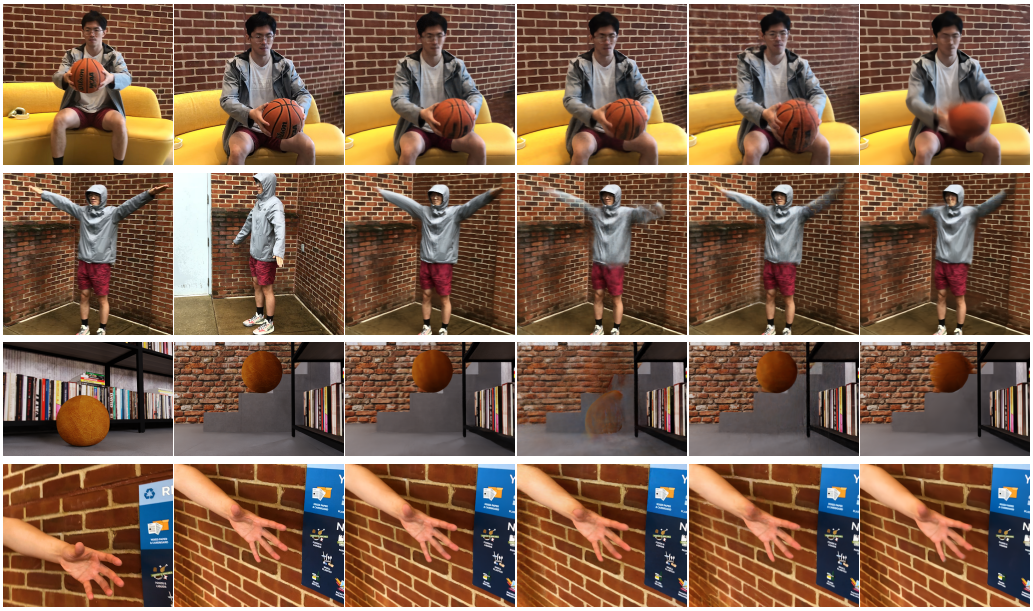
NeuPhysics enables high-resolution mesh reconstruction from a casual video captured by a smartphone. It would be easy for users to collect their own data to customize virtual body avatars for games and social applications. Accurate facial and body models can also be used in virtual try-on, or even customized suits, furniture, and sports equipment. Moreover, the physically-based, dynamic, photorealistic editing technique could be used in designing, content creation, and broader VR/AR applications.

Appendix D Other Supplementary Materials

More dynamic visualization results can be found in our supplementary video; we invite the readers to view these videos for visual comparison. A preliminary version of our code is included for reviewing only. We will also open-source a release version in the future for research reproduction with the paper publication.

Appendix E Video Reconstruction

For the video reconstruction task, we evaluate the methods on a dataset consisting of both real-world captured and synthetic videos. The synthetic data are animated by Blender, and the real-world videos are taken by a smartphone in both indoor and outdoor scenes. Figure 11 visualizes the datasets as well as the reconstruction videos by different methods. (a, b) are two frames from the input videos where both the scene and the camera positions have changed. (c) is by our method, while (d) NRNeRF [57] (e) D-NeRF [47], and (f) NeuS [61] are SOTA methods for comparison. Note that (d) and (e) are designed for dynamic videos as input, but not designed for geometry reconstruction. (f) learns SDF field during the video reconstruction as ours does, however, it is originally designed for *static* scenes so inconsistency among views would likely undermine its performance.



(a) Input 1 (b) Input 2 (c) Ours (d) NRNeRF [57] (e) D-NeRF [47] (f) NeuS [61]

Figure 11: **Qualitative evaluation of video reconstruction**. We evaluate four methods on both real and synthetic videos. Our method outperforms other SOTA methods with higher fidelity and less blur.