arXiv:2305.06356v2 [cs.CV] 11 May 2023

HumanRF: High-Fidelity Neural Radiance Fields for Humans in Motion

MUSTAFA IŞIK, Synthesia, Germany
MARTIN RÜNZ, Synthesia, Germany
MARKOS GEORGOPOULOS, Synthesia, United Kingdom
TARAS KHAKHULIN, Synthesia, United Kingdom
JONATHAN STARCK, Synthesia, United Kingdom
LOURDES AGAPITO, University College London, United Kingdom
MATTHIAS NIESSNER, Technical University of Munich, Germany

Multi-view Capture

Novel-view Synthesis with HumanRF

Fig. 1. We introduce a novel multi-view dataset of humans in motion captured with a rig of 160 cameras, recording footage of 12MP each (left image). From an input multi-view recording of a specific person, our HumanRF method reconstructs a spatio-temporal radiance field which captures appearance and motion of the actor. From this representation, we can then synthesize highly-realistic images from unseen, novel view points (right images).

Representing human performance at high-fidelity is an essential building block in diverse applications, such as film production, computer games or videoconferencing. To close the gap to production-level quality, we introduce HumanRF[1], a 4D dynamic neural scene representation that captures full-body appearance in motion from multi-view video input, and enables playback from novel, unseen viewpoints. Our novel representation acts as a dynamic video encoding that captures fine details at high compression rates by factorizing space-time into a temporal matrix-vector decomposition. This allows us to obtain temporally coherent reconstructions of human actors for long sequences, while representing high-resolution details even in the context of challenging motion. While most research focuses on synthesizing at resolutions of 4MP or lower, we address the challenge of operating at 12MP. To this end, we introduce ActorsHQ, a novel multi-view dataset that provides 12MP footage from 160 cameras for 16 sequences with high-fidelity, per-frame mesh reconstructions[2]. We demonstrate challenges that emerge from using such high-resolution data and show that our newly introduced HumanRF effectively leverages this data, making a significant step towards production-level quality novel view synthesis.

CCS Concepts: • **Computing methodologies → Computer vision representations**.

Additional Key Words and Phrases: neural rendering, free-view video synthesis

ACM Reference Format:
Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. 2023. HumanRF: High-Fidelity Neural Radiance Fields for Humans in Motion. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 12 pages. https://doi.org/10.1145/3592415

1 INTRODUCTION

Photo-realistic image synthesis of virtual environments has been one of the core challenges in computer graphics research for decades. Traditionally, the underlying 3D assets have been created by artists

[1]Project website: synthesiaresearch.github.io/humanrf

Authors' addresses: Mustafa Işık, mustafa.isik@synthesia.io, Synthesia, Munich, Germany; Martin Rünz, martin@synthesia.io, Synthesia, Munich, Germany; Markos Georgopoulos, markos@synthesia.io, Synthesia, London, United Kingdom; Taras Khakhulin, taras.khakhulin@synthesia.io, Synthesia, London, United Kingdom; Jonathan Starck, jon@synthesia.io, Synthesia, London, United Kingdom; Lourdes Agapito, l.agapito@cs.ucl.ac.uk, University College London, London, United Kingdom; Matthias Nießner, niessner@tum.de, Technical University of Munich, Munich, Germany.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3592415.

[2]ActorsHQ dataset is publicly available under www.actors-hq.com including all raw RGB frames and per-frame reconstructed 3D meshes.

with heavy manual labor; however, recently, significant effort has been devoted to reconstructing the 3D representations from real-world observations. In particular, novel view synthesis of recorded humans has been the center of attention in numerous applications, ranging from movie and game production to immersive telepresence.

Yet, reconstructing photo-realistic digital humans from real-world captures involves significant technical challenges. The diverse granularity of fine-scale detail – e.g., on faces, hair, clothing – makes the reconstruction difficult to scale, while the margin for error is low due to the acute ability of the human visual system to perceive even the smallest inconsistencies in synthesized images. From a methodological standpoint, the main challenge lies in jointly reconstructing appearance and motion in realistic settings due to the large number of degrees of freedom that needs to be encoded. In particular, modeling fast and complex motions while obtaining photo-realistic results at a sufficient resolution remains an open problem in production.

In recent years, we have seen tremendous progress in addressing these challenges. More specifically, Mildenhall et al. [2020] reconstructs a 3D neural radiance field (NeRF) constrained by a multi-view volumetric rendering loss. The resulting 3D field is encoded in a multi-layer perceptron (MLP) which then enables novel-view synthesis. While NeRF originally focused on static scenes, recent works handle dynamic scenes implicitly via time conditioning [Li et al. 2022a] or explicitly via deformation fields [Park et al. 2021a,b]. These dynamic methods show impressive results; but, they still struggle to handle longer sequences with complex motion – especially for humans. In the mean time, obtaining high-quality output renderings requires high-resolution training data which is both difficult to capture and utilize in the subsequent radiance field reconstructions.

In this work, we propose to address these shortcomings of dynamic NeRF methods in the context of capturing moving humans. Therefore, we first introduce ActorsHQ, a new high-fidelity dataset of clothed humans in motion tailored for photo-realistic novel view synthesis. The dataset features multi-view recordings of 160 synchronized cameras that simultaneously capture individual video streams of 12MP each, as illustrated in Fig. 1. Leveraging our newly captured data, we propose a new scene representation that lifts Instant-NGP [Müller et al. 2022] hash encodings to the temporal domain by incorporating the time dimension in conjunction with a low-rank space-time tensor decomposition of the feature grid. We further split a sequence into segments, which allows representing very long sequences as only few of the segments need to reside in GPU memory during a training iteration – something existing methods struggle with due to using a single representation for an entire sequence. Finally, we demonstrate the effectiveness of our new representation on our newly introduced dataset where we significantly improve over existing state-of-the-art methods. Concretely, our contributions are as follows:

- We propose a new spatio-temporal decomposition that can efficiently reconstruct a dynamic radiance field representation from multi-view inputs, based on a low-rank decomposition.
- Additionally, we introduce an adaptive splitting scheme which divides a sequence into segments allowing us to capture arbitrarily long sequences.
- We further introduce ActorsHQ, a high-fidelity dataset, featuring footage of 8 actors from 160 cameras that record at a resolution of 12MP each.

2 RELATED WORK

HumanRF leverages hybrid implicit and volumetric representations to reconstruct free-viewpoint videos. In this section, we discuss related work on neural representations for static and dynamic scenes and for human performance capture.

2.1 3D Neural Representations

3D reconstruction is a long-standing problem that has been redefined with the advent of deep learning-based approaches. In particular, coordinate-based networks have become a popular choice for implicit 3D scene representations such as radiance [Mildenhall et al. 2020], signed distance [Park et al. 2019], or occupancy [Mescheder et al. 2019] fields. In the pioneering work of Mildenhall et al. [2020], an MLP is trained to encode a radiance field reconstructed from a set of input RGB images. Alternatively, some methods utilize explicit data structures, such as sparse grids [Fridovich-Keil et al. 2022], to achieve fast training and inference at the expense of a larger memory footprint. TensoRF [Chen et al. 2022b] addresses memory inefficiencies by using a low-rank tensor decomposition while Müller et al. [2022] propose using hash data structures accompanied with small MLPs. We elevate the ideas from TensoRF into spatio-temporal domain by representing the feature grids via 4D decomposition using four 3D hash grids and four 1D dense grids.

2.2 4D Dynamic Representations

The creation of free-viewpoint videos has been widely studied due to its numerous applications. The seminal work of Kanade et al. [1997] allows the reconstruction of shapes and textures using a multi-camera dome. Similarly, later efforts [Carranza et al. 2003; Starck and Hilton 2007] leveraged multiple cameras for free-viewpoint human rendering. More recently, the breakthrough work of Collet et al. [2015] proposes to track textured meshes in order to create streamable 3D videos. Broxton et al. [2020] presents a layered mesh representation to reconstruct and compress video from a multi-camera rig. More recently, deep learning-based approaches have been proposed for deformable 3D scenes. Neural Volumes [Lombardi et al. 2019] use an encoder-decoder architecture to optimize a 3D volume from 2D images. Similarly, instead of decoding a 3D volume, Lombardi et al. [2021] propose to decode a mixture of volumetric primitives that are attached to a guide mesh.

A plethora of efforts has been dedicated to extending the success of NeRF into the temporal domain using implicit representations. Li et al. [2022a] extends NeRF with time-conditioning and introduces a keyframe-based training strategy. Alternatively, Park et al. [2021a,b]; Pumarola et al. [2021] introduce a separate MLP to predict scene deformations for multi-view and monocular videos, respectively. In similar vain, Li et al. [2021] leverages 2D flow supervision to model a dynamic scene. Similarly to static scenes, the slow convergence of such methods has been addressed using explicit [Liu et al. 2022] and hybrid [Fang et al. 2022a; Guo et al. 2022a] representations. Orthogonal to these approaches, Wang et al. [2022b] fuses a set of

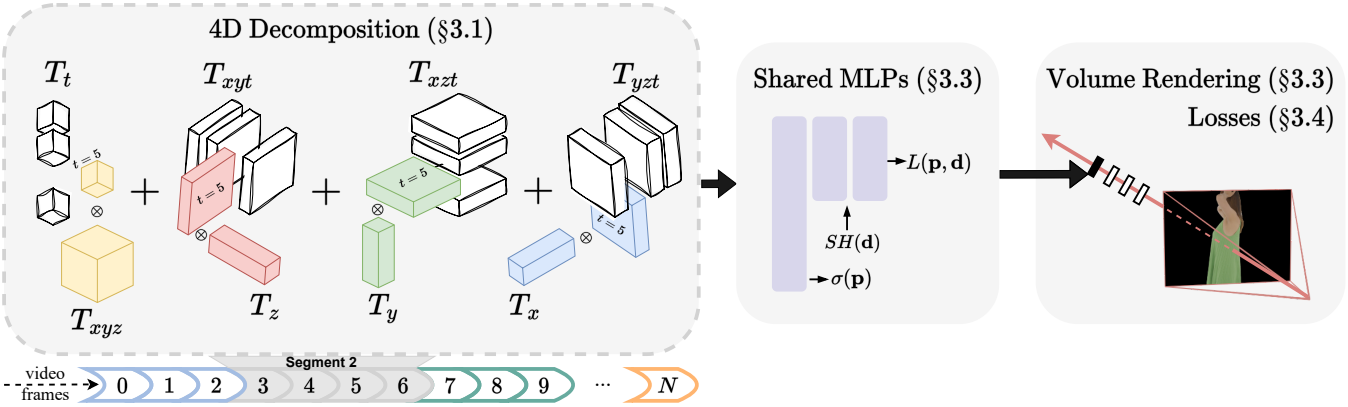

Fig. 2. **Overview of HumanRF**: Prior to training, our method starts by splitting the temporal domain into 4D segments with similar union occupancy in 3D (§3.2). Each segment is modeled by a 4D feature grid which is compactly represented by utilizing tensor decomposition and hash grids (§3.1). During training, we sample a batch of rays across different time frames and cameras. After each pixel color is predicted via volume rendering (§3.3), we enforce photometric constraints and regularize ray marching weights via foreground masks (§3.4).

static PlenOctrees [Yu et al. 2021] into a dynamic representation using DFT to achieve real-time inference.

Concurrently to our work, Song et al. [2022] decompose the 4D space into static, deforming and newly appeared regions, while Cao and Johnson [2023]; Fridovich-Keil et al. [2023]; Shao et al. [2022] also propose to represent 4D scenes using low-rank decompositions with 2D tensors. Unlike these methods, our method uses 3D and 1D tensors. In §5.3.1, we show our 3D-1D scheme is significantly better than its 2D-2D counterpart for rapid motions. Additionally, our method partitions a sequence into segments, which enables training at scale on modern GPUs without sacrificing quality.

2.3 Neural Human Performance Capture

Our goal is closely related to neural radiance field-based methods that specialize in rendering humans. This is often achieved by learning a canonical representation that is forward warped to a target frame [Chen et al. 2021; Wang et al. 2022a] or backward sampled from the observation space [Liu et al. 2021; Xu et al. 2022]. This deformation can be guided by a learned template model such as SMPL [Loper et al. 2015] or a sparse skeleton. For instance, utilizing sparse landmarks, Noguchi et al. [2021]; Su et al. [2021] reparametrize radiance fields relative to the pose of the skeleton. TAVA [Li et al. 2022b] optimizes a canonical shape and forward-skinning weights based on skeleton pose. Neural Body [Peng et al. 2021] employs a SMPL model to optimize a latent representation for each mesh vertex, while Liu et al. [2021] learns a backwards warping into the canonical pose. While these methods achieve impressive results, they also suffer from the innate limitations of template-based approaches; i.e., their approximate geometry (e.g., from SMPL) or ambiguous pose conditioning (e.g., skeletal joints) often poses challenges in novel view synthesis. This becomes particularly problematic for fine-scale deformations such as dynamic cloth animations or local detail in the face which cannot be represented by existing geometric template proxies. To address these challenges, recent efforts opt for template-free approaches. For instance, Zhang et al. [2022] train a time-conditioned network to predict hyper-spherical harmonics for free-viewpoint human rendering while Zhao et al. [2022] propose a static-to-dynamic approach where per-frame neural surface reconstruction is combined with a hybrid neural tracker to generate neural animated human meshes. In our work, we also propose a template-free approach since we are aiming for the highest visual quality.

3 METHOD

Given a set of input videos of a human actor in motion, captured in a multi-view camera setting, our goal is to enable temporally consistent, high-fidelity novel view synthesis. To that end, we learn a 4D scene representation using differentiable volumetric rendering [Lombardi et al. 2019; Mildenhall et al. 2020], supervised via multi-view 2D photometric and mask losses that minimize the discrepancy between the rendered images and the set of input RGB images and foreground masks. To enable efficient photo-realistic neural rendering of arbitrarily long multi-view data, we use sparse feature hash-grids in combination with shallow multilayer perceptrons (MLPs) [Müller et al. 2022; Sun et al. 2022; Wang et al. 2021].

The core idea of HumanRF – as illustrated in Fig. 2 – is to partition the time domain into optimally distributed temporal segments, and to represent each segment by a compact 4D feature grid (§3.1). For this purpose, we propose an extension to the TensoRF vector-matrix decomposition of Chen et al. [2022b] – designed for static 3D scenes – that can support time-varying 4D feature grids. Our adaptive temporal partitioning (§3.2) ensures that the total 3D space volume covered by each individual temporal segment is of similar size, which helps our method achieve superior representation power, regardless of the temporal context. Furthermore, we use shallow MLPs to transform features into density and view-dependent radiance to be used in the volumetric rendering framework (§3.3). Through sharing information across the temporal domain via both shared MLPs and 4D decomposition, our results are temporally consistent. We refer to the accompanying videos regarding temporal stability.

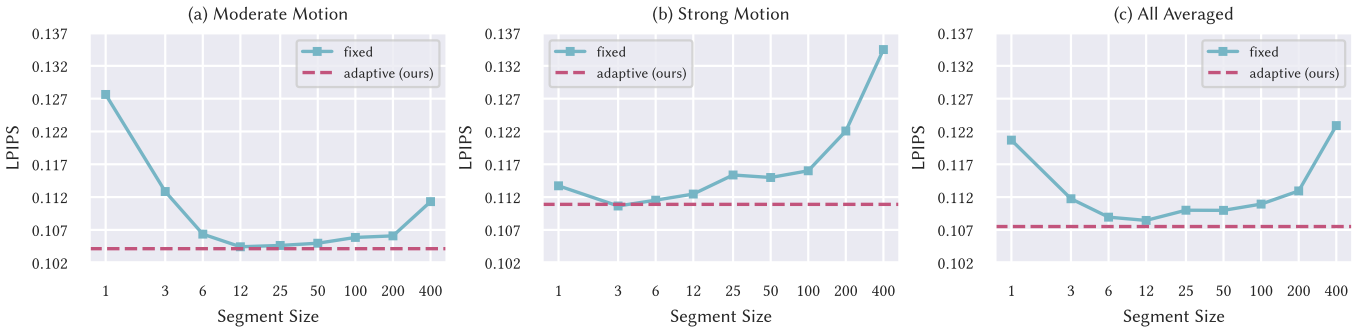

Fig. 3. **Fixed-segment vs. adaptive partitioning** Using a single 4D representation for an entire sequence (segment size 400) or using a 3D hash grid per frame (segment size 1), give poor results. We observe that finding the middle ground (segment sizes from 3 to 100) leads to better results (a, b, c). Sequences with moderate motions favor larger segment sizes whereas those with stronger motions favor smaller ones (b). Our Adaptive Temporal Partitioning scheme (§3.2) avoids the costly hyper-parameter search for the optimal, global segment size, and leads to results close to those of optimal segment sizes (a, b). On average, our adaptive method is better than using any fixed segment size (c). These experiments are performed on 400-frame sequences using shared MLPs. The total number of parameters is kept approximately the same while varying the segment size.

We supervise our differentiable rendering pipeline with 2D-only losses that measure the errors between the rendered and input RGB images and foreground masks (§3.4).

3.1 4D Feature Grid Decomposition

Our method models a dynamic 3D scene by combining optimally partitioned 4D segments. Each segment k has its own trainable 4D feature grid $T^{(k)}_{xyzt} : \mathbb{R}^4 \mapsto \mathbb{R}^m$ which encodes a set of N_k consecutive frames $\mathcal{T}^{(k)} \in \{t_s, t_{s+1}, t_{s+2}, ..., t_{s+N_k-1}\}$. Previous works [Chen et al. 2022b; Müller et al. 2022; Tang et al. 2022] have shown that dense 3D data exhibits redundancies and can be represented more compactly. We make the same argument for spatio-temporal data, and define our 4D feature grid as a decomposition of four 3D and four 1D feature grids (k is dropped for brevity below):

$$\begin{aligned} T_{xyzt}(\mathbf{p}_{xyzt}) = T_{xyz}(\mathbf{p}_{xyz}) \odot T_t(\mathbf{p}_t) \\ + T_{xyt}(\mathbf{p}_{xyt}) \odot T_z(\mathbf{p}_z) \\ + T_{xzt}(\mathbf{p}_{xzt}) \odot T_y(\mathbf{p}_y) \\ + T_{yzt}(\mathbf{p}_{yzt}) \odot T_x(\mathbf{p}_x) \end{aligned} , \tag{1}$$

where $\odot$ denotes the hadamard product, and $\mathbf{p}_{xyzt} \in \mathbb{R}^4$ is the queried point. We represent each 3D grid ($T_{xyz}, T_{xyt}, T_{xzt}, T_{yzt} : \mathbb{R}^3 \mapsto \mathbb{R}^m$) with a multi-resolution hash grid [Müller et al. 2022] – which has proven to be more efficient than using dense 3D grids – and each 1D grid ($T_t, T_z, T_y, T_x : \mathbb{R} \mapsto \mathbb{R}^m$) with dense array of vectors. In Fig. 9, we show that this compact representation lets our method surpass the quality of per-frame Instant-NGP while using only a fraction of the number of trainable parameters.

3.2 Adaptive Temporal Partitioning

Using a single 4D feature grid for an entire sequence becomes impractical for longer sequences. Figure 3 shows that representing a long sequence with a single 4D segment performs significantly worse than using multiple fixed-sized segments, especially when considering the total hash capacities are roughly the same. Therefore, partitioning the sequence plays a critical role in our representation. Figure 3 also highlights the impact of motion complexity on the optimal segment size (stronger deformations require shorter segments). To mitigate the prohibitive cost of hyper-parameter search for the optimal segment size, we present a greedy algorithm to adaptively select the sizes of segments prior to training. Unlike fixed-size temporal partitioning, our method does not require a unique segment size which would be less suited to very long sequences.

Occupancy Grids. To reason about temporal changes, we analyze per-frame occupancy grids $O^t : \mathcal{P} \subset \mathbb{R}^3 \mapsto \{0, 1\}$ that we compute by carving the free space [Kutulakos and Seitz 2000] using foreground masks. We define several terms that will be relevant hereafter. First, we define the occupancy grid of a set of frames $\mathcal{T}$ by the logical union of their occupancy grids:

$$O^{\mathcal{T}}(\mathbf{p}) = \bigvee_{t_j \in \mathcal{T}} O^{t_j}(\mathbf{p}) . \tag{2}$$

Second, for a set of frames $\mathcal{T}$, *total occupancy* is defined as the number of occupied voxels:

$$\delta(\mathcal{T}) = \sum_{\mathbf{p}_j \in \mathcal{P}} O^{\mathcal{T}}(\mathbf{p}_j) . \tag{3}$$

Finally, for a set of N consecutive frames $\mathcal{T} = \{t_0, t_1, t_2, ..., t_{N-1}\}$, we define the *expansion factor* as:

$$\phi(\mathcal{T}) = \frac{\delta(\mathcal{T})}{\delta(\{t_0\})} , \tag{4}$$

which practically indicates how much the union occupancy grid is enlarged from t_0 onwards, and positively correlates with the motion complexity.

Criteria for spawning new segments. Given a fixed budget of total number of trainable parameters, our objective is to keep the expansion factor (Equation (4)) similar for each segment. To this end, we iterate over each frame with a greedy heuristic and spawn a new segment when the expansion factor exceeds a certain threshold. This ensures that each segment represents a similar amount of volume in 3D space, which leads to a fair distribution of the total representation workload. This can also be regarded as maximizing

the efficiency of the 4D decomposition models by adjusting the temporal context of each segment such that the temporal sharing is encouraged for smaller movements and discouraged for larger ones. In Figure 8, we experiment with several threshold values for the expansion factor, and set it to 1.25 for all our experiments.

3.3 Shared MLPs and Volume Rendering

Similarly to previous work [Mildenhall et al. 2020], we describe the distribution of the radiance in a scene at time instance t using the volumetric rendering formulation with emission and absorption:

$$\hat{C}(\boldsymbol{r}, t) = \int_{\alpha_{min}}^{\alpha_{max}} T(\alpha)\sigma(\boldsymbol{r}(\alpha), t)L(\boldsymbol{r}(\alpha), \mathbf{d}, t)d\alpha \,, \tag{5}$$

where $T(\alpha)$ is the transmittance, $\boldsymbol{r}(\alpha)$ is the point on the ray $\boldsymbol{r}$ at distance α, $\sigma(\boldsymbol{r}(\alpha), t)$ denotes the volumetric density, and $L(\boldsymbol{r}(\alpha), \mathbf{d}, t)$ indicates the radiance emitted along the direction $\mathbf{d}$. We solve this integral numerically using quadrature [Max 1995]. Similarly to Müller et al. [2022], we use two shallow MLPs to model density and view-dependent radiance. First, we leverage a 3-layer network, $\text{MLP}_\sigma : \mathbb{R}^{32} \mapsto \mathbb{R}^{16}$, to generate density $\sigma(\mathbf{p}, t) \in \mathbb{R}$ and geometry features $F(\mathbf{p}, t) \in \mathbb{R}^{15}$ for any point $\mathbf{p} \in \mathbb{R}^3$ in time t of segment k

$$\{\sigma(\mathbf{p}, t), F(\mathbf{p}, t)\} = \text{MLP}_\sigma(T^{(k)}_{xyzt}(\mathbf{p}, t)) \,. \tag{6}$$

Then, we employ a 4-layer network $\text{MLP}_L : \mathbb{R}^{31} \mapsto \mathbb{R}^3$ to produce view-dependent RGB radiance values:

$$L(\mathbf{p}, \mathbf{d}, t) \;= \text{MLP}_L(\text{SH}(\mathbf{d}), F(\mathbf{p}, t)) \,, \tag{7}$$

where $\text{SH}(\mathbf{d}) \in \mathbb{R}^{16}$ is the encoding of the viewing direction $\mathbf{d}$ formed by using the first 4 bands of the spherical harmonics. Although each spatio-temporal segment has its own trainable 4D feature grid, these two MLPs are shared by an entire sequence.

3.4 Losses

We utilize RGB images and masks to guide the training. First, we enforce the Huber loss [Collins 1976] between the ground truth color $C(\boldsymbol{r}, t)$ and the predicted pixel color $\hat{C}(\boldsymbol{r}, t)$ (Equation (5)):

$$\mathcal{L}_{pho} = \frac{1}{|\mathcal{R}|} \sum_{\boldsymbol{r}\in\mathcal{R}} \begin{cases} \frac{1}{2}l^2, & \text{if } l \leq \delta \\ \delta \cdot (l - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \,, \tag{8}$$

where $l = |C(\boldsymbol{r}, t) - \hat{C}(\boldsymbol{r}, t)|$. This loss is averaged over 3 color channels and we set $\delta = 0.01$ in all of our experiments.

In addition to background removal and occupancy grid computation, we use foreground masks to regularize volumetric occupancy similarly to Yariv et al. [2020]. More specifically, we use the binary cross entropy loss between the ground truth mask $M(\boldsymbol{r})$ and the accumulated volume rendering weight $\hat{M}(\boldsymbol{r})$:

$$\mathcal{L}_{bce} = \frac{1}{|\mathcal{R}|} \sum_{\boldsymbol{r}\in\mathcal{R}} \left[M(\boldsymbol{r}) \log(\hat{M}(\boldsymbol{r})) + (1 - M(\boldsymbol{r})) \log((\hat{M}(\boldsymbol{r}))\right] \,, \tag{9}$$

where $M(\boldsymbol{r}) = 1$ and $M(\boldsymbol{r}) = 0$ denote the pixels on the foreground and the background, respectively, and $\hat{M}(\boldsymbol{r})$ is defined as follows,

$$\hat{M}(\boldsymbol{r}) = \int_{\alpha_{min}}^{\alpha_{max}} T(\alpha)\sigma(\boldsymbol{r}(\alpha))d\alpha \,. \tag{10}$$

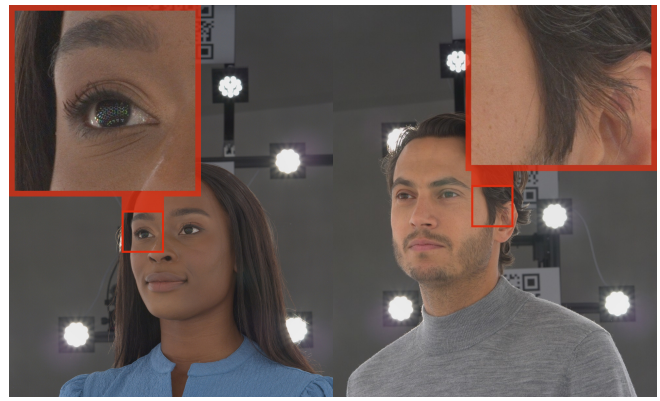

Fig. 4. **Dataset resolution** We show closeups for two actors in our ActorsHQ dataset. The cameras record at 12MP each, thus enabling the capture of eyelashes, wrinkles, and hair strands.

This loss helps to prune the empty space early in the training, which leads to significant speed up in the training iterations. Our final loss term is defined as

$$\mathcal{L} = \mathcal{L}_{pho} + \beta\mathcal{L}_{bce} \,, \tag{11}$$

where we set $\beta = 10^{-3}$ for all of our experiments. We refer the reader to the supplemental material for additional training details.

4 DATASET

Our dataset, ActorsHQ, consists of 39,765 frames of dynamic human motion captured using multi-view video. We used a proprietary multi-camera capture system combined with an LED array for global illumination. The camera system comprises 160 12MP Ximea cameras operating at 25fps. Close-up details that are captured at this resolution are highlighted in Fig. 4. The lighting system provides a programmable lighting array of 420 LEDs that are time-synchronized to the camera shutter. All cameras were set to a shutter speed of 650us to minimize motion blur for fast actions. We additionally reconstruct each frame independently using state-of-the-art multi-view stereo from RealityCapture [Epic Games 2022] with approximately 500k faces per frame (Fig. 5). The camera array was configured to cover a capture volume of 1.6m diameter and 2.2m height, enabling actors to perform a range of motions at the center.

The dataset comprises 4 female and 4 male actors. Each actor performed two 100 second motion sequences of choreographed actions wearing everyday clothing. The actors wore either short or long upper and lower body clothing to provide variation in cloth dynamics. In the first sequence, each actor followed the same set of 32 actions that activate key joint rotations for shoulders, arms, legs, and torso as well as combined joint activations. The actors were directed to return to a resting A-pose between sets of actions. In the second sequence, the actors performed 20 randomly selected everyday actions designed to produce more exaggerated body poses such as sports, dance, celebration, and gestures. The actors were directed to move continuously throughout the capture to provide more exaggerated body dynamics in motion. A comparison of the ActorsHQ with other standard benchmarks is tabulated in Table 1.

Fig. 5. **Actors and meshes.** Our ActorsHQ dataset contains 8 actors with casual clothing such as skirts or shorts. Each sequence is captured by 160 camera, each recording at 12MP. In addition to the recorded images, we also provide high-quality, per-frame mesh reconstructions with approximately 500k vertices.

Table 1. **Comparison of multi-view human video datasets.** Our new dataset features longer sequences with more cameras at a higher resolution.

Dataset	#ID	#Frames	Resolution	#Cameras
Human3.6M [Ionescu et al. 2013]	11	581k	1MP	4
MPI-INF-3DHP [Mehta et al. 2017]	8	>1.3M	4MP	14
ZJU-Mocap [Peng et al. 2021]	9	< 2,700	1MP	21
DynaCap [Habermann et al. 2021]	5	27k	1.2MP	50-101
THUman [Zheng et al. 2019]	500	500k	0.4MP	4
THUman4 [Zheng et al. 2022]	3	<15k	1.4MP	24
ActorsHQ (ours)	8	39,765	**12MP**	**160**

5 EVALUATION

To demonstrate the ability of HumanRF to represent long sequences and fine details at 12MP, we perform extensive quantitative and qualitative experiments with our ActorsHQ dataset. As HumanRF is a temporal method, we also highly recommend watching the supplementary videos.

We compare our method against six state-of-the-art baselines. There are three deformation-based approaches for general scenes: NDVG [Guo et al. 2022b], HyperNeRF [Park et al. 2021b], and TiNeuVox [Fang et al. 2022b] along with two human-specific methods: Neural Body [Peng et al. 2021] and TAVA [Li et al. 2022b]. As an additional baseline, we train Instant-NGP [Müller et al. 2022] independently on each frame. For all baselines, we use the official implementations that are publicly available and tune hyper-parameters to achieve best possible results. A visual comparison between the baselines and our method can be found on Fig. 6 and Fig. 11.

5.1 Evaluation Protocol

In all experiments, we use the same set of 124 training cameras, 10 validation cameras and 14 test cameras. For one frontal test camera we render a video that is used to compute the VMAF [Li et al. 2016] score, and we alternate through the remaining test cameras to compute PSNR, LPIPS [Zhang et al. 2018] and SSIM [Wang et al. 2004]. The numerical results are averaged over 8 actors for all the experiments. As some baselines fail to produce reasonable results at full resolution, we compare on 4× downscaled data (per axis), producing better relative performance compared to our results. To test the performance of HumanRF on high-resolution data, we perform additional experiments in full resolution (12MP) in §5.4. More details on the protocol can be found in the supplementary material.

5.2 Quality vs Number of Frames

In Table 2, we analyze each method for various sequence lengths using per-frame metrics PSNR, LPIPS, and SSIM, as well as the temporal metric VMAF, which measures perceptual quality of the generated videos and correlates well with temporal consistency. Due to our efficient 4D feature grid structure and its ability to scale to arbitrarily long sequences via temporal partitioning, HumanRF consistently outperforms the baselines. Existing methods that predict a deformation field struggle to represent long sequences with complex motion. This is mainly due to rapid topological changes and

Table 2. **Numerical evaluation on ActorsHQ**. We demonstrate results using the standard visual metrics and VMAF to measure perceptual video quality. HumanRF outperforms baselines on all sequences. Instant-NGP – trained per frame separately – demonstrates better LPIPS, but struggles in terms of temporal consistency and memory footprint (Fig. 9). The best and the second best results are highlighted.

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.095	0.100	0.097	0.100	0.102	0.107
	↑ PSNR	30.30	30.05	29.83	29.26	29.34	29.05
	↑ SSIM	0.918	0.918	0.921	0.920	0.919	0.913
	↑ VMAF	83.67	84.43	85.62	85.28	85.33	85.74
Instant-NGP	↓ LPIPS	0.095	0.092	0.094	0.093	0.093	0.093
	↑ PSNR	29.45	28.78	28.96	28.77	28.73	28.85
	↑ SSIM	0.881	0.898	0.902	0.904	0.904	0.905
	↑ VMAF	74.15	73.23	76.70	76.77	77.28	77.60
TiNeuVox	↓ LPIPS	0.312	0.305	0.327	0.346	0.348	0.371
	↑ PSNR	26.07	24.14	23.11	21.82	20.94	19.80
	↑ SSIM	0.792	0.800	0.794	0.792	0.786	0.772
	↑ VMAF	56.26	47.31	40.27	31.11	24.74	18.11
NDVG	↓ LPIPS	0.268	0.275	0.300	0.338	0.367	0.391
	↑ PSNR	26.60	23.65	22.16	19.75	17.93	16.17
	↑ SSIM	0.823	0.811	0.793	0.765	0.741	0.716
	↑ VMAF	63.26	50.53	38.13	21.88	12.76	6.183
HyperNeRF	↓ LPIPS	0.250	0.235	0.251	0.270	0.302	0.325
	↑ PSNR	25.70	25.23	24.72	23.82	22.58	21.77
	↑ SSIM	0.820	0.832	0.826	0.817	0.801	0.790
	↑ VMAF	73.08	73.05	67.17	57.51	44.84	37.01
Neural Body	↓ LPIPS	0.305	0.308	0.310	0.318	0.340	0.367
	↑ PSNR	27.03	25.16	26.92	24.66	24.35	25.58
	↑ SSIM	0.806	0.807	0.805	0.805	0.793	0.767
	↑ VMAF	46.72	45.27	41.83	38.71	32.13	26.95
TAVA	↓ LPIPS	0.270	0.277	0.295	0.344	0.388	0.429
	↑ PSNR	27.44	25.75	25.05	23.61	22.40	21.50
	↑ SSIM	0.820	0.821	0.816	0.792	0.765	0.740
	↑ VMAF	66.92	58.66	54.28	38.40	23.45	13.34

limited representation power for deformations, and the effect is typically reflected in both the per-frame and temporal metrics. Previous works [Li et al. 2022c; Shao et al. 2022] have also observed similar disadvantages on complex or fast-changing scenes for deformation-based approaches. Although human-specific baselines perform better than deformation-based ones on average, they still lack visual details, and tend to produce blurry results compared to HumanRF as illustrated in Fig. 6. On the other hand, per-frame Instant-NGP excels on per-frame metrics, but it lacks temporal stability, and uses 20× more trainable parameters compared to our method (Fig. 9).

5.3 Design Choices

In this section, we validate the effectiveness of our approach, and clarify how we select several hyper parameters. More specifically, we perform ablation studies regarding the 4D feature grid representation (§5.3.1). We discuss choosing the optimal grid resolution, feature dimensionality and hash size (§5.3.2), and argue how segment sizes and expansion factor thresholds are determined (§5.3.3).

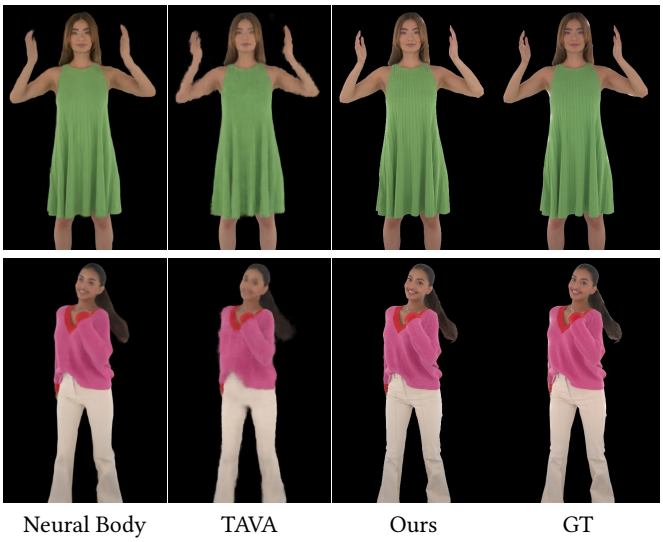

Fig. 6. **Comparison with human-specific methods.** Although Li et al. [2022b]; Peng et al. [2021] incorporate geometry and pose information, they fail to capture fine details, and produce blurrier results compared to HumanRF.

5.3.1 4D Feature Grid. To validate our choice of feature grid representation, we run a comparison against two ablations of our method where we only alter the 4D feature grid and use a single segment over 100-frame sequences. The first variant simply concatenates 3D spatial coordinates with time to form 4D input to a hash grid – we dub this as *tNGP*. The second variant also utilizes a 4D decomposition by using six multi-resolution 2D dense grids inspired by the concurrent work [Cao and Johnson 2023; Fridovich-Keil et al. 2023] – which we dub as *Hex4D*. Please refer to supplemental to see how Hex4D is formulated. Unlike Hex4D, our method uses four multi-resolution 3D hash grids and four 1D dense grids. Table 3 indicates that using a decomposition model (Hex4D and ours) for the feature grid is superior when the motion is moderate as the temporal context can be efficiently compressed into lower-rank tensors. Although Hex4D loses its advantage over tNGP for stronger motion, our method consistently outperforms both ablations in both cases.

5.3.2 Grid Resolution and Feature Dimensionality. In order to determine the optimal grid resolution and feature dimensionality of the 4D feature grid, we perform a parameter search in two dimensions: finest grid resolution (K_{max}) and per-level feature dimensionality (F). To facilitate the search, we perform the experiments on 4× downscaled data over 100-frame sequences, and we fix the coarsest resolution in our multi-resolution grids to $K_{\text{min}} = 32$ and number of resolution levels to $L = 16$. To narrow down the search even further, we fix the total number of trainable parameters per hash grid as $T \cdot L \cdot F = 2^{24}$, where T denotes per-level hash size. Finally, we restrict hash size to be $T \leq 2^{19}$, because further increase leads to performance penalty, which is also reported by Müller et al. [2022]. From the results presented in Fig. 7, we pick $K_{\text{max}} = 2048$ and $F = 2$. For experiments carried out in full resolution (see §5.4 and Fig. 10),

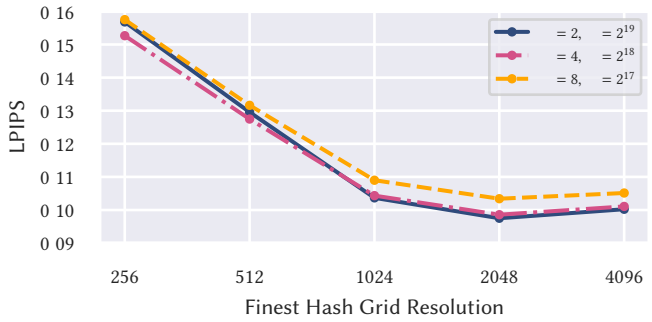

Fig. 7. **Optimal grid resolution and feature dimensionality.** To determine the ideal parameters for our feature grid, we search through several finest grid resolutions (K_{max}) and feature dimensionalities (F) while fixing the number of parameters. As a result, we use $K_{\text{max}} = 2048$ and $F = 2$.

Table 3. **Comparison between different feature grid representations.** Hex4D outperforms tNGP for scenes with moderate motion due to its ability to compress, however, its quality heavily degrades for rapid motion. On the other hand, our method benefits from its compact nature while not sacrificing the quality as much as Hex4D, and it consistently outperforms both ablations. We use default multi-resolution grid parameters for these ablations (see §5.3.2). The best and the second best results are highlighted.

Metric	Moderate Motion			Strong Motion		
	Hex4D	tNGP	Ours	Hex4D	tNGP	Ours
↓ LPIPS	0.105	0.129	0.090	0.184	0.126	0.110
↑ PSNR	29.89	29.27	30.79	26.04	28.10	28.87
↑ SSIM	0.915	0.906	0.931	0.851	0.902	0.906
↑ VMAF	77.50	79.18	81.15	75.22	87.76	89.00

we set $K_{\text{max}} = 8192$ and use $L = 24$ to maintain the model capacity so that the finer details can be reconstructed.

5.3.3 Model Size. Our method uses 4D spatio-temporal segments with various lengths to represent arbitrarily long sequences. Adaptive temporal partitioning (§3.2) tries to keep the number of trainable parameters per frame approximately the same in order not to lose its representation power. For this reason, the number of trainable parameters scales linearly with the sequence length. Nonetheless, our method remains more compact than most of the baselines. In Fig. 9, we demonstrate that HumanRF uses only 5.2% of the parameters compared to per-frame Instant-NGP while outperforming it.

Predefined segment sizes. During adaptive temporal partitioning, we choose segment sizes from a pool of predefined lengths where each one has a hash capacity proportional to its size. For our experiments, we specifically use the segment sizes 6, 12, 25, 50, 100 with per-level hash sizes $2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}$, respectively, using the *Tiny CUDA neural networks* framework [Müller 2021]. Moreover, we demonstrate the effect of motion complexity and expansion factor threshold on average segment size in Fig. 8.

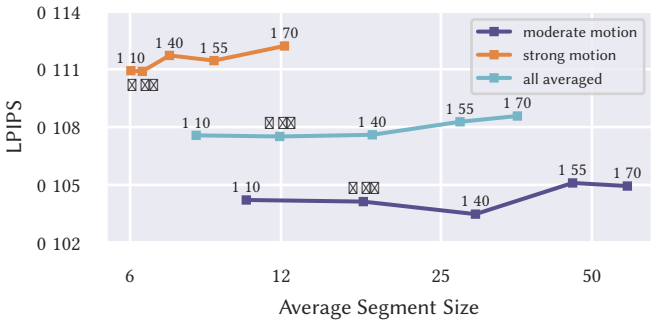

Fig. 8. **Impact of the expansion factor thresholds and motion complexity on average segment size and quality.** Larger threshold values (indicated by numbers) lead to larger segments on average. Unlike using fixed-size segments, we do not observe a striking difference in quality when the average segment size changes (see Fig. 3 for a comparison). Furthermore, rapid motions increase the frequency of spawning new segments, and hence lead to smaller segment sizes.

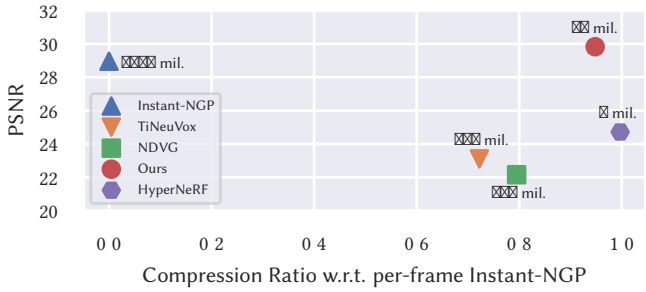

Fig. 9. **Compression ratio vs. PSNR on 100-frame sequence.** We demonstrate number of parameters for each baseline (bold numbers) and the compression ratio with respect to per-frame Instant-NGP. The resulted ratio for HumanRF is near the most compact one with the best PSNR quality. Here, we define compression ratio for a method M as $(1 - \frac{P_M}{P_{\text{I-NGP}}})$ where P_M denotes the number of trainable parameters.

5.4 Input Resolution

Previous publicly available datasets provide images at the resolution of 4MP or less, but our method is designed to capture details beyond this resolution. Fig. 10 illustrates the impact of using downscaled training data on the rendering quality at full resolution. We observe that HumanRF can recover finer details as the input resolution increases – see supplemental for numerical results. While it seems natural that scores improve with increasing resolution of training data, we observed that some baselines struggle to represent high-fidelity data and deteriorate instead.

5.5 Dynamic Furry Animal Dataset

Although HumanRF is tailored to ActorsHQ, it is a template-free method which is not necessarily restricted to humans. In fact, our method can be applied to any scene with a foreground object with masks. To demonstrate this ability, we run our method on Dynamic Furry Animal (DFA) [Luo et al. 2022] which is a multi-view dataset of furry animals in motion. In Table 4, we demonstrate that our

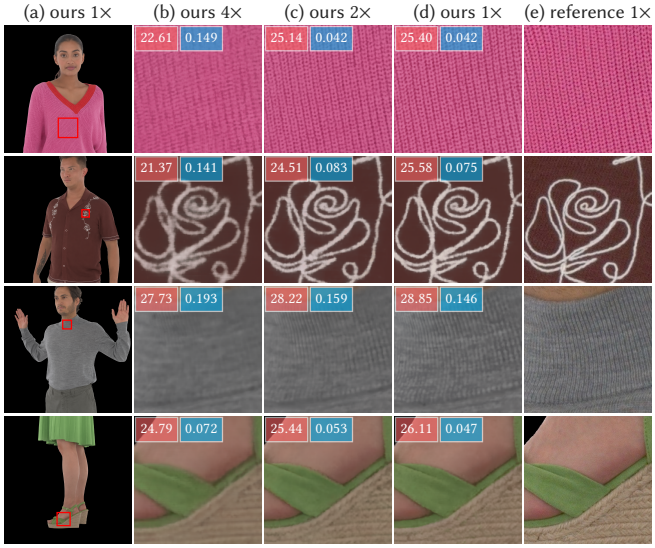

Fig. 10. **Impact of input resolution on the full-resolution results.** We illustrate the significance of high resolution data by training HumanRF on 4× downscaled (b), 2× downscaled (c) and full-resolution (a, c) input to generate full-resolution results. We observe striking differences in capturing finer details as the resolution increases, which is also reflected in highlighted ↑PSNR and ↓LPIPS values.

method can be applied to non-human scenes and it can still surpass the quality of the state of the art. For these experiments, we use default settings (§5.3.2) except by setting $L = 20$ to account for high-frequent fur details. Additional visual results can be found in the supplementary material.

5.6 Limitations and Future Work

Our method can produce high-fidelity radiance field reconstructions of humans in motion, achieving accurate results on novel view synthesis; however, important limitations remain. To achieve such high-quality results, HumanRF relies on our newly-introduced ActorsHQ dataset, and optimizes a separate radiance field for each sequence. It would be interesting to explore training a model on high-end recordings which could then be used as an avatar to target monocular-only test sequences. While our method reconstructs each frame of a motion sequence, we still do not have explicit control over articulation of the actor outside the training poses. One possible way to gain control could be to learn a deformation network for each segment, or to operate with a parametric model to control explicit parameters. At the same time, there is also significant room to speed up render times of our method. Here, a promising direction could be the conversion of our reconstructed radiance field into a hybrid, implicit-explicit representation such as in MobileNeRF [Chen et al. 2022a]. Finally, although our model is temporally-stable, the foreground masks are not necessarily consistent across different time frames because they are inferred from independent, per-frame mesh reconstructions, which leads to flickering effect on the silhouette edges. Here, our work would benefit from temporally-consistent background matting techniques, such as Lin et al. [2022].

Table 4. **Evaluation on the Dynamic Furry Animal (DFA) Dataset [Luo et al. 2022].** We show that HumanRF can reconstruct radiance fields for dynamic sequences of non-human subjects, such as animals in the DFA dataset. Our method achieves state-of-the-art results even though all the baselines except NeuralVolumes uses skeleton information provided in the DFA. For the starred (*) methods, we use the results from Table 1 in Luo et al. [2022] , and use the exact same evaluation configuration for our method. The best and the second best results are highlighted.

Method	Metric	Panda	Cat	Dog	Lion
Ours	↓ LPIPS	0.030	0.008	0.013	0.025
	↑ PSNR	36.00	38.43	37.79	35.40
	↑ SSIM	0.986	0.992	0.986	0.979
Artemis*	↓ LPIPS	0.031	0.012	0.022	0.035
	↑ PSNR	33.63	37.54	38.95	33.09
	↑ SSIM	0.985	0.989	0.989	0.966
Animatable NeRF*	↓ LPIPS	0.112	0.061	0.074	0.123
	↑ PSNR	26.51	31.37	31.19	27.87
	↑ SSIM	0.957	0.973	0.975	0.944
Neural Volumes*	↓ LPIPS	0.116	0.087	0.129	0.123
	↑ PSNR	30.11	28.14	26.80	29.59
	↑ SSIM	0.965	0.951	0.945	0.947
Neural Body*	↓ LPIPS	0.110	0.067	0.075	0.111
	↑ PSNR	30.38	30.77	32.27	30.11
	↑ SSIM	0.970	0.972	0.978	0.956

6 CONCLUSION

We have presented HumanRF, a novel method to reconstruct a spatio-temporal radiance field that captures human performance at high-fidelity. At the core of our method lies an intra-frame decomposition of a 4D representation based on a multi-resolution hash grid to capture details. To handle arbitrarily long sequences with a practical memory budget, we introduce an adaptive splitting technique to share as many features as possible between frames and produce a memory-efficient representation. To demonstrate the advantages of our method, we have introduced ActorsHQ, the first publicly available multi-view dataset captured with 160 cameras recording 12MP footage. Our results have shown high-quality free-viewpoint video, which we believe makes an important step towards production-level novel view synthesis. Finally, we hope that the release of ActorsHQ dataset and the source code for HumanRF will enable researchers to drive new advances in photo-realistic reconstruction of virtual humans.

ACKNOWLEDGMENTS

We thank Lee Perry-Smith and Henry Pearce at Infinite Realities Ltd for specialist volumetric scanning services to build ActorsHQ using the AEONX Motion Scanning System. Thanks also to Carolin Hecking-Veltman, Daniel Thul, and Tymoteusz Bleja for their generous efforts to help prepare the dataset, Haimin Luo for providing the details of the evaluation for DFA, and the anonymous Siggraph reviewers for their valuable suggestions.

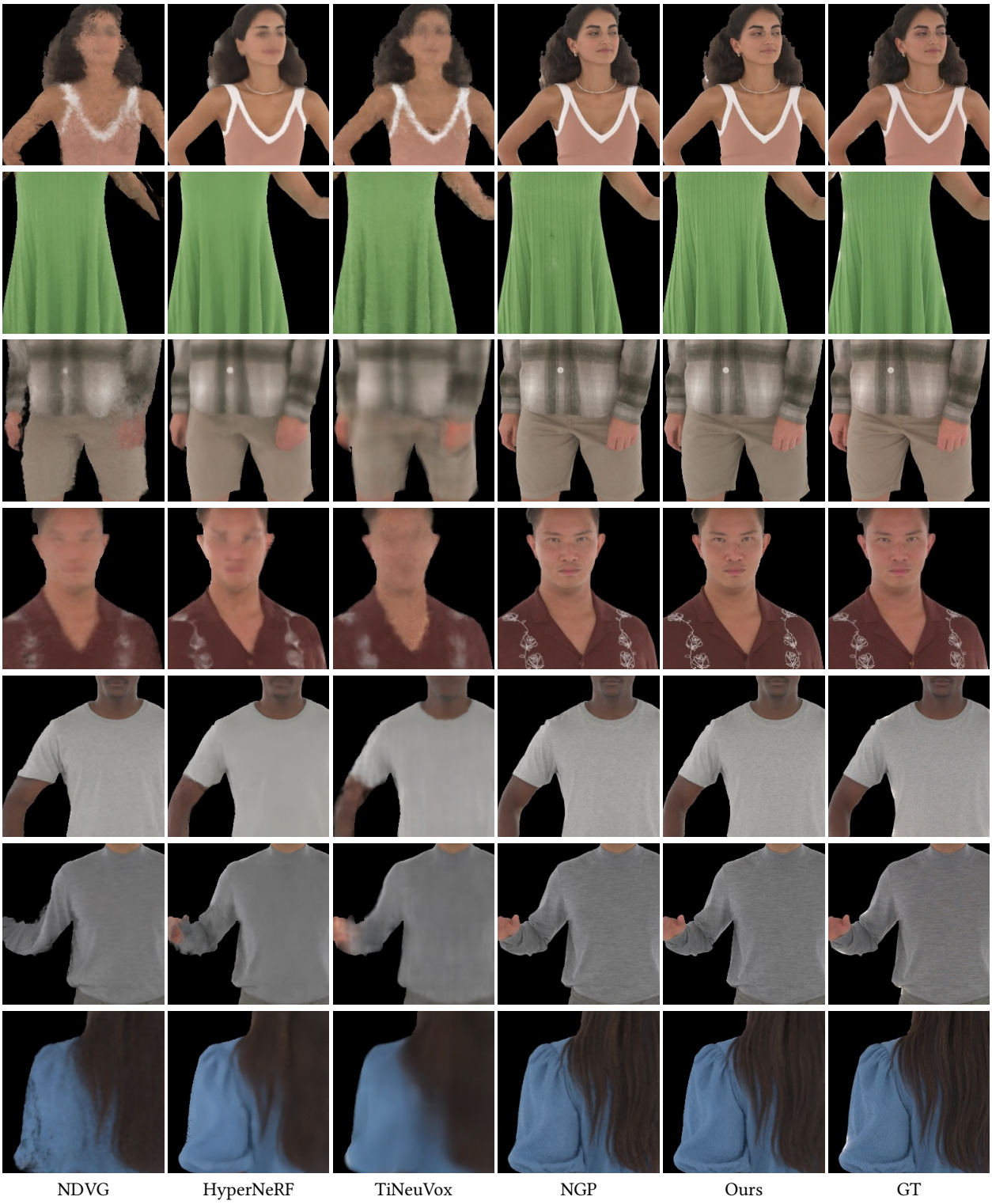

Fig. 11. **Qualitative comparison.** The synthesis quality of HumanRF is visually compared to the 4 baselines NDVG [Guo et al. 2022b], HyperNeRF [Park et al. 2021b], TiNeuVox [Fang et al. 2022b] and per-frame NGP [Müller et al. 2022] using a sequence of 100 frames. While deformation-based baselines tend to produce blurry results and can fail to capture rapid motions, NGP and ours are able to generate crisp images that are close to groundtruth.

REFERENCES

Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. 2020. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 86–1.

Ang Cao and Justin Johnson. 2023. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR* (2023).

Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. 2003. Free-viewpoint video of human actors. In *ACM Transactions on Graphics (TOG)*.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022b. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*.

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

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. 2022a. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277* (2022).

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam G. Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. In *ACM Transactions on Graphics (TOG)*, Vol. 34. 1 – 13.

John R Collins. 1976. Robust estimation of a location parameter in the presence of asymmetry. *The Annals of Statistics* (1976), 68–85.

Inc. Epic Games. 2022. *RealityCapture*. https://www.capturingreality.com Accessed: 2023-01-12.

Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. 2022a. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *arXiv preprint arXiv:2205.15285* (2022).

Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. 2022b. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. In *SIGGRAPH Asia 2022 Conference Papers*.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. 2023. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *CVPR*.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance Fields Without Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5501–5510.

Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. 2022a. Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis. In *Proceedings of the Asian Conference on Computer Vision*. 3757–3775.

Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. 2022b. Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

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

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2013. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence* 36, 7 (2013), 1325–1339.

Takeo Kanade, Peter Rander, and PJ Narayanan. 1997. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE multimedia* 4, 1 (1997), 34–47.

Kiriakos N Kutulakos and Steven M Seitz. 2000. A theory of shape by space carving. *International journal of computer vision* 38, 3 (2000), 199–218.

Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhöfer, Jürgen Gall, Angjoo Kanazawa, and Christoph Lassner. 2022b. Tava: Template-free animatable volumetric actors. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer, 419–436.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022a. Neural 3D Video Synthesis From Multi-View Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward a practical perceptual video quality metric. *The Netflix Tech Blog* 6, 2 (2016).

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2021. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6498–6508.

Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. 2022c. DynIBaR: Neural Dynamic Image-Based Rendering. *arXiv preprint arXiv:2211.11082* (2022).

Shanchuan Lin, Linjie Yang, Imran Saleemi, and Soumyadip Sengupta. 2022. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 238–247.

Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2022. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. *Advances in Neural Information Processing Systems*.

Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. 2021. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–16.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages.

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: a skinned multi-person linear model. *ACM Trans. Graph.* 34 (2015), 248:1–248:16.

Haimin Luo, Teng Xu, Yuheng Jiang, Chenglin Zhou, Qiwei Qiu, Yingliang Zhang, Wei Yang, Lan Xu, and Jingyi Yu. 2022. Artemis: Articulated Neural Pets with Appearance and Motion Synthesis. *ACM Trans. Graph.* 41, 4, Article 164 (jul 2022), 19 pages. https://doi.org/10.1145/3528223.3530086

Nelson Max. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 1, 2 (1995), 99–108.

Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. 2017. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*. IEEE, 506–516.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4460–4470.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Thomas Müller. 2021. *tiny-cuda-nn*. https://github.com/NVlabs/tiny-cuda-nn Accessed: 2022-10-21.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127

Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. 2021. Neural articulated radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5762–5772.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 165–174.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021a. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5865–5874.

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

Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. 2021. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9054–9063.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 2022. Tensor4D: Efficient Neural 4D Decomposition for High-fidelity Dynamic Reconstruction and Rendering. *arXiv preprint arXiv:2211.11610* (2022).

Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2022. NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. *arXiv preprint arXiv:2210.15947* (2022).

Jonathan Starck and Adrian Hilton. 2007. Surface capture for performance-based animation. *IEEE computer graphics and applications* 27, 3 (2007), 21–31.

Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. 2021. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. *Advances in Neural Information Processing Systems* 34 (2021), 12278–12291.

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *CVPR*.

Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. 2022. Compressible-composable NeRF via Rank-residual Decomposition. *arXiv preprint arXiv:2205.14870*

(2022).

Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. 2022b. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13524–13534.

Shaofei Wang, Katja Schwarz, Andreas Geiger, and Siyu Tang. 2022a. Arah: Animatable volume rendering of articulated human sdfs. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer, 1–19.

Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhofer. 2021. Learning Compositional Radiance Fields of Dynamic Human Heads. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5704–5713.

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

Tianhan Xu, Yasuhiro Fujita, and Eiichi Matsumoto. 2022. Surface-Aligned Neural Radiance Fields for Controllable 3D Human Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15883–15892.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. *Advances in Neural Information Processing Systems* 33 (2020).

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5752–5761.

Jiakai Zhang, Liao Wang, Xinhang Liu, Fuqiang Zhao, Minzhang Li, Haizhao Dai, Boyuan Zhang, Wei Yang, Lan Xu, and Jingyi Yu. 2022. NeuVV: Neural Volumetric Videos with Immersive Rendering and Editing. *arXiv preprint arXiv:2202.06088* (2022).

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Fuqiang Zhao, Yuheng Jiang, Kaixin Yao, Jiakai Zhang, Liao Wang, Haizhao Dai, Yuhui Zhong, Yingliang Zhang, Minye Wu, Lan Xu, and Jingyi Yu. 2022. Human Performance Modeling and Rendering via Neural Animated Mesh. In *ACM Transactions on Graphics (TOG)*, Vol. 41. 1 – 17.

Zerong Zheng, Han Huang, Tao Yu, Hongwen Zhang, Yandong Guo, and Yebin Liu. 2022. Structured local radiance fields for human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15893–15903.

Zerong Zheng, Tao Yu, Yixuan Wei, Qionghai Dai, and Yebin Liu. 2019. Deephuman: 3d human reconstruction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7739–7749.

HumanRF: Supplementary Material

MUSTAFA IŞIK, Synthesia, Germany
MARTIN RÜNZ, Synthesia, Germany
MARKOS GEORGOPOULOS, Synthesia, United Kingdom
TARAS KHAKHULIN, Synthesia, United Kingdom
JONATHAN STARCK, Synthesia, United Kingdom
LOURDES AGAPITO, University College London, United Kingdom
MATTHIAS NIESSNER, Technical University of Munich, Germany

arXiv:2305.06356v2 [cs.CV] 11 May 2023

ACM Reference Format:
Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. 2023. HumanRF: Supplementary Material. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 11 pages. https://doi.org/10.1145/3592415

1 IMPLEMENTATION DETAILS

Here, we discuss our implementation tricks that enable neural rendering of terabytes of multi-view data.

1.1 HumanRF

Our method is implemented in PyTorch [Paszke et al. 2019] and in CUDA for some of the parts that require performance. We use *Tiny CUDA neural networks* framework [Müller 2021] to create four 3D hash grid representations. To reduce the amount of intermediate memory usage during training and improve performance, we write a CUDA kernel that samples from four 1D dense grids and compose the results with the sampled features from the hash grid. Furthermore, we utilize some of the functionalities from *torch-ngp* [Tang 2022] and *NerfAcc* [Li et al. 2022].

In our high resolution video results on the supplemental video, we make use of per-camera embeddings [Martin-Brualla et al. 2021] which are concatenated to the input of the radiance MLP. This helps removing the brightness and lighting inconsistencies that arise for some cameras. In addition, we filter the light bloom effect based on the light source annotations shown in Fig. 1. That is, we do not sample rays from the annotated circular regions to prevent using pixels that have light diffused into it. We note that this modified version is not used in any of the comparisons made in the main paper or supplementary material for fairness. It is simply used for the stand-alone 12MP video results.

Authors' addresses: Mustafa Işık, mustafa.isik@synthesia.io, Synthesia, Munich, Germany; Martin Rünz, martin@synthesia.io, Synthesia, Munich, Germany; Markos Georgopoulos, markos@synthesia.io, Synthesia, London, United Kingdom; Taras Khakhulin, taras.khakhulin@synthesia.io, Synthesia, London, United Kingdom; Jonathan Starck, jon@synthesia.io, Synthesia, London, United Kingdom; Lourdes Agapito, l.agapito@cs.ucl.ac.uk, University College London, London, United Kingdom; Matthias Nießner, niessner@tum.de, Technical University of Munich, Munich, Germany.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3592415.

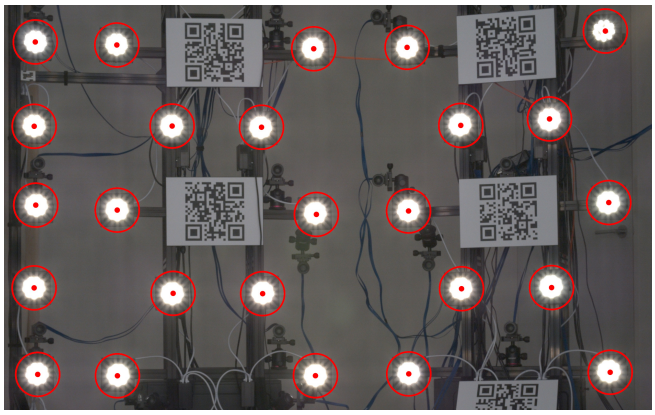

Fig. 1. **Light annotations.** Light bloom can have a significant impact on photometric consistency. Light annotations are used to avoid using these regions during training.

1.2 ActorsHQ Data Loader

Considering the terabytes of data we need to deal with, storing training data in memory, or preparing batches in a pre-computation stage and saving it to the file system is impractical in terms of memory or hard disk requirements. The idea behind our data loader is to bypass the reading and writing large chunks of data by sampling batch of rays on the fly from as many images as possible across different cameras and time frames. To do this, we define a pool of images, and randomly sample from this pool continuously in the main thread while another thread is working in the background to replace the images in the pool. By replacing and sampling concurrently, we use only a modest amount of GPU and CPU memory to accommodate the pool. Also, we implement custom CUDA kernels to make use of the occupancy grids (that are initialized from masks) to skip empty space during ray sampling. This speeds up the rendering significantly, and increases the effective capacity of the model because the empty space does not have to be modeled.

1.3 Training

We use ADAM optimizer [Kingma and Ba 2014] with the initial learning rate of 10^{-2}. We decay this learning rate to $5 \cdot 10^{-3}$ until the end of each training. We utilize FP16 operations for fast training and inference. For experiments with 4× downscaled input, we train for $N \times 1000$ iterations where N depicts the number of frames in the training sequence. On average, our implementation performs

8 to 12 training iterations per second on a single NVIDIA GeForce RTX 3090 with 24GB memory. This corresponds to roughly a day of compute for 1000-frame sequences. On the other hand, we train for $N \times 2500$ iterations for full-resolution trainings.

We define the training batch size in terms of maximum number of samples over all the training rays in a batch. We start by sampling 8192 rays per batch, and dynamically adjust number of rays such that the maximum number of samples is reached every iteration. This lets us achieve high GPU utilization during training. We set the maximum number of samples to 640K, 576K and 512K for 4× downscaled input, 2× downscaled input and full-resolution input, respectively.

As our method partitions a given sequence into segments, it is possible to scale the training to thousands of frames. This is because we sample rays across fixed number of time frames, which we set to 8 for all our experiments. Therefore, in the worst case scenario, only 8 different segments need to live in the GPU memory. On average, our segments have a size of 12 which would translate to around 64 million parameters (256MB) that need to reside in the GPU memory at a time instance on average.

2 EVALUATION

In this section, we clarify the details of camera and frame configurations used during training, validation and testing. In addition, we explain how the metrics are calculated to generate numerical results.

2.1 Evaluation Protocol

In the following, we describe the evaluation protocol used for our baseline comparison experiments. The dataset was split into 4 disjunct sets of cameras that are listed by their 1-based index:

- 124 training cameras: 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 18, 19, 21, 22, 23, 24, 26, 27, 28, 29, 30, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 46, 47, 48, 49, 50, 52, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63, 66, 67, 68, 69, 70, 72, 73, 75, 76, 77, 78, 79, 80, 81, 82, 83, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 99, 100, 101, 102, 103, 106, 107, 108, 109, 110, 111, 112, 113, 114, 116, 117, 119, 120, 121, 122, 123, 124, 125, 126, 128, 131, 132, 133, 134, 135, 136, 139, 140, 141, 142, 143, 144, 149, 150, 151, 152, 157, 158, 159, 160
- 10 validation cameras: 11, 20, 34, 45, 51, 74, 84, 91, 105, 118
- 13 per-frame test cameras: 1, 14, 25, 31, 44, 58, 64, 65, 71, 85, 98, 104, 115
- 1 VMAF test camera: 127

When computing per-frame scores, we alternate the test cameras in the following order: 1, 64, 98, 31, 14, 71, 115, 25, 85, 44, 65, 104, 58 and temporally subsample every fifth frame leading to frame-camera pairs such as $\{(1, 1), (6, 64), (11, 98), ...\}$. To reduce the computational burden to execute this comparison, we use one of the sequences per actor alternatingly, i.e. Actor1 Sequence1, Actor2 Sequence2, Actor3 Sequence1, Actor4 Sequence2, Actor5 Sequence1, Actor6 Sequence2, Actor7 Sequence1 and Actor8 Sequence2. Sequence1s contain moderate movements while Sequence2s contain stronger motion. PSNR scores are computed only on the foreground depicted by the ground truth masks while SSIM and LPIPS are computed by tightly cropping images to fit ground truth foreground masks. Finally, VMAF is computed on the video that is compiled by rendering every third frame from the hero camera (camera 127).

Table 1. **Frame resolution vs representation quality.** Training HumanRF at different input resolutions while rendering results at full resolution shows that additional details can be represented, and our method can make use of the extra information provided with the higher resolutions.

Resolution	PSNR ↑	LPIPS ↓	SSIM ↑	VMAF ↑
12MP	28.07	0.348	0.812	68.76
12MP/(2 × 2)	27.69	0.360	0.809	65.37
12MP/(4 × 4)	27.29	0.375	0.799	59.31

2.2 Numerical Results of Input Resolution Experiment

In Table 1, we provide additional results concerning the input resolution experiment we present in the main paper.

2.3 Additional Numerical Results on Baseline Comparison

We provide additional results over different motion complexities in Table 2 and Table 3, and per sequence results. Moreover, we provide a plot in Fig. 3 to better illustrate the effect of increasing the sequence length.

2.4 Visual Results on DFA Dataset

In Fig. 2, we present results of our method for four scenes we choose from DFA. We infer that HumanRF can produce high-fidelity results for non-human subjects as well.

3 HEX4D AND TNGP FORMULATIONS

Following the notations we have used to define Equation 1 in the main paper, we define Hex4D formulation as follows:

$$\begin{aligned} T_{xyzt}(\mathbf{p}_{xyzt}) = T_{xy}(\mathbf{p}_{xy}) \odot T_{zt}(\mathbf{p}_{zt}) \\ +T_{yz}(\mathbf{p}_{yz}) \odot T_{xt}(\mathbf{p}_{xt}) \ , \\ +T_{xz}(\mathbf{p}_{xz}) \odot T_{yt}(\mathbf{p}_{yt}) \end{aligned} \tag{1}$$

where we represent six 2D planes ($T_{xy}, T_{yz}, T_{xz}, T_{zt}, T_{xy}, T_{yt} : \mathbb{R}^2 \mapsto \mathbb{R}^m$) using multi-resolution dense grids. Notice that HexPlanes [Cao and Johnson 2023] uses concatenation operation instead of addition as opposed to our formulation. However, for our experiments, we did not observe an improvement when using concatenation over addition.

On the other hand, tNGP simply uses a 4D hash grid to represent $T_{xyzt} : \mathbb{R}^4 \mapsto \mathbb{R}^m$ without utilizing any kind of decomposition techniques.

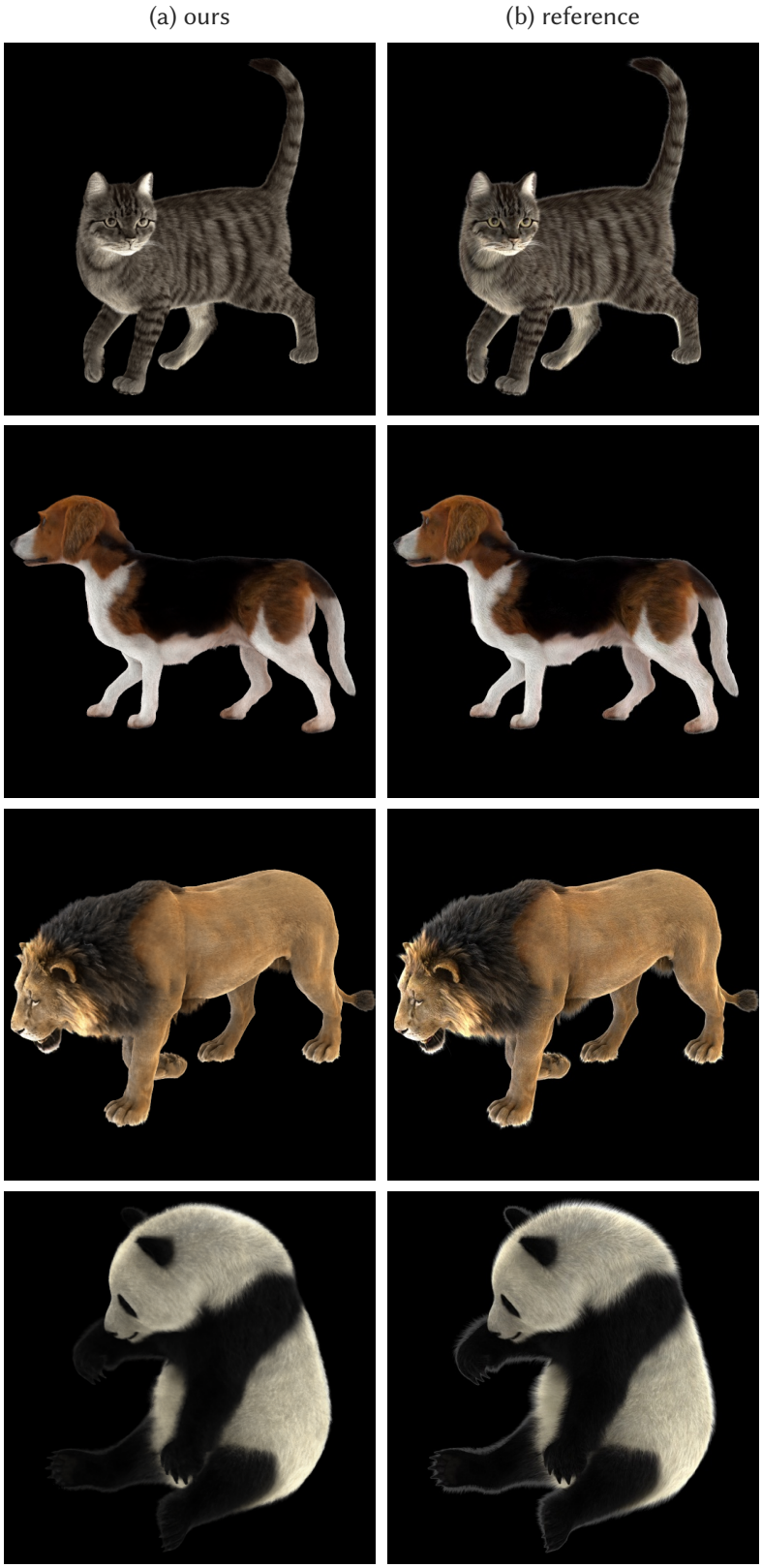

Fig. 2. **Visual results from DFA dataset.**

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.081	0.090	0.088	0.095	0.100	0.103
	↑ PSNR	31.58	31.12	30.72	30.10	30.15	29.93
	↑ SSIM	0.933	0.930	0.932	0.927	0.924	0.920
	↑ VMAF	79.13	79.49	81.66	81.93	82.05	83.15
Instant-NGP	↓ LPIPS	0.100	0.096	0.099	0.100	0.100	0.099
	↑ PSNR	29.88	29.72	29.67	29.65	29.60	29.76
	↑ SSIM	0.884	0.910	0.910	0.910	0.908	0.911
	↑ VMAF	68.91	65.83	70.84	71.45	72.11	73.00
TiNeuVox	↓ LPIPS	0.287	0.252	0.308	0.334	0.337	0.361
	↑ PSNR	27.60	26.61	25.01	23.86	23.03	21.56
	↑ SSIM	0.819	0.826	0.809	0.804	0.797	0.779
	↑ VMAF	59.43	59.58	47.69	38.91	32.23	22.75
NDVG	↓ LPIPS	0.195	0.212	0.247	0.312	0.337	0.376
	↑ PSNR	29.61	26.94	25.26	22.20	20.73	18.31
	↑ SSIM	0.875	0.860	0.834	0.794	0.778	0.739
	↑ VMAF	73.04	64.10	51.34	31.65	21.29	9.317
HyperNeRF	↓ LPIPS	0.228	0.222	0.236	0.256	0.294	0.321
	↑ PSNR	25.74	25.88	25.94	25.07	23.95	22.94
	↑ SSIM	0.841	0.842	0.835	0.823	0.805	0.794
	↑ VMAF	74.35	74.72	69.83	62.03	48.92	39.10
NeuralBody	↓ LPIPS	0.272	0.289	0.289	0.303	0.327	0.356
	↑ PSNR	27.28	25.89	27.87	25.38	25.16	26.66
	↑ SSIM	0.822	0.814	0.810	0.809	0.798	0.763
	↑ VMAF	42.95	42.22	41.02	37.28	32.35	29.29
TAVA	↓ LPIPS	0.218	0.236	0.260	0.313	0.362	0.411
	↑ PSNR	28.62	27.29	26.50	25.12	23.96	22.86
	↑ SSIM	0.848	0.841	0.830	0.806	0.782	0.757
	↑ VMAF	66.45	62.64	57.15	43.75	30.22	18.56

Table 2. moderate movements

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.108	0.111	0.106	0.105	0.104	0.112
	↑ PSNR	29.02	28.98	28.95	28.41	28.53	28.17
	↑ SSIM	0.903	0.906	0.910	0.912	0.913	0.906
	↑ VMAF	88.20	89.37	89.59	88.63	88.61	88.33
Instant-NGP	↓ LPIPS	0.090	0.087	0.088	0.086	0.086	0.087
	↑ PSNR	29.02	27.84	28.25	27.89	27.86	27.93
	↑ SSIM	0.877	0.886	0.894	0.899	0.900	0.899
	↑ VMAF	79.38	80.64	82.55	82.10	82.44	82.20
TiNeuVox	↓ LPIPS	0.337	0.357	0.346	0.359	0.359	0.381
	↑ PSNR	24.55	21.66	21.22	19.78	18.85	18.03
	↑ SSIM	0.764	0.774	0.779	0.780	0.775	0.765
	↑ VMAF	53.09	35.03	32.86	23.32	17.24	13.48
NDVG	↓ LPIPS	0.342	0.339	0.353	0.364	0.396	0.405
	↑ PSNR	23.60	20.36	19.06	17.30	15.13	14.02
	↑ SSIM	0.771	0.763	0.753	0.736	0.704	0.693
	↑ VMAF	53.48	36.96	24.92	12.11	4.236	3.048
HyperNeRF	↓ LPIPS	0.272	0.248	0.266	0.283	0.310	0.328
	↑ PSNR	25.65	24.58	23.51	22.58	21.22	20.60
	↑ SSIM	0.800	0.822	0.817	0.810	0.797	0.787
	↑ VMAF	71.81	71.38	64.51	52.98	40.77	34.92
NeuralBody	↓ LPIPS	0.339	0.326	0.332	0.333	0.353	0.377
	↑ PSNR	26.77	24.43	25.97	23.93	23.53	24.50
	↑ SSIM	0.791	0.800	0.800	0.801	0.789	0.770
	↑ VMAF	50.49	48.32	42.64	40.15	31.90	24.61
TAVA	↓ LPIPS	0.322	0.318	0.330	0.376	0.415	0.447
	↑ PSNR	26.26	24.21	23.60	22.11	20.84	20.13
	↑ SSIM	0.792	0.801	0.802	0.778	0.749	0.722
	↑ VMAF	67.40	54.67	51.41	33.06	16.68	8.129

Table 3. strong movements

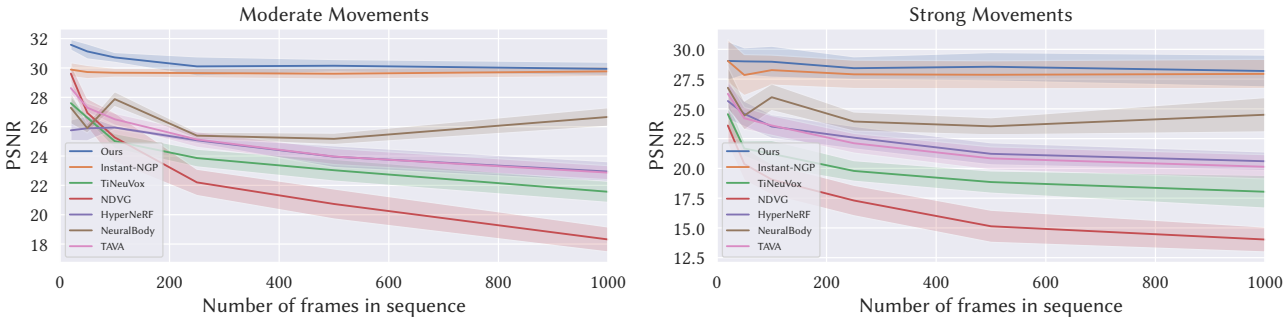

Fig. 3. **Influence of increasing the sequence length.** Thanks to using adaptively-placed 4D segments, our method consistently outperforms the deformation-based baselines as they struggle to capture complex motion over long sequences. Although NeuralBody does not lose its representation power for long sequences, its overall quality is inferior to HumanRF.

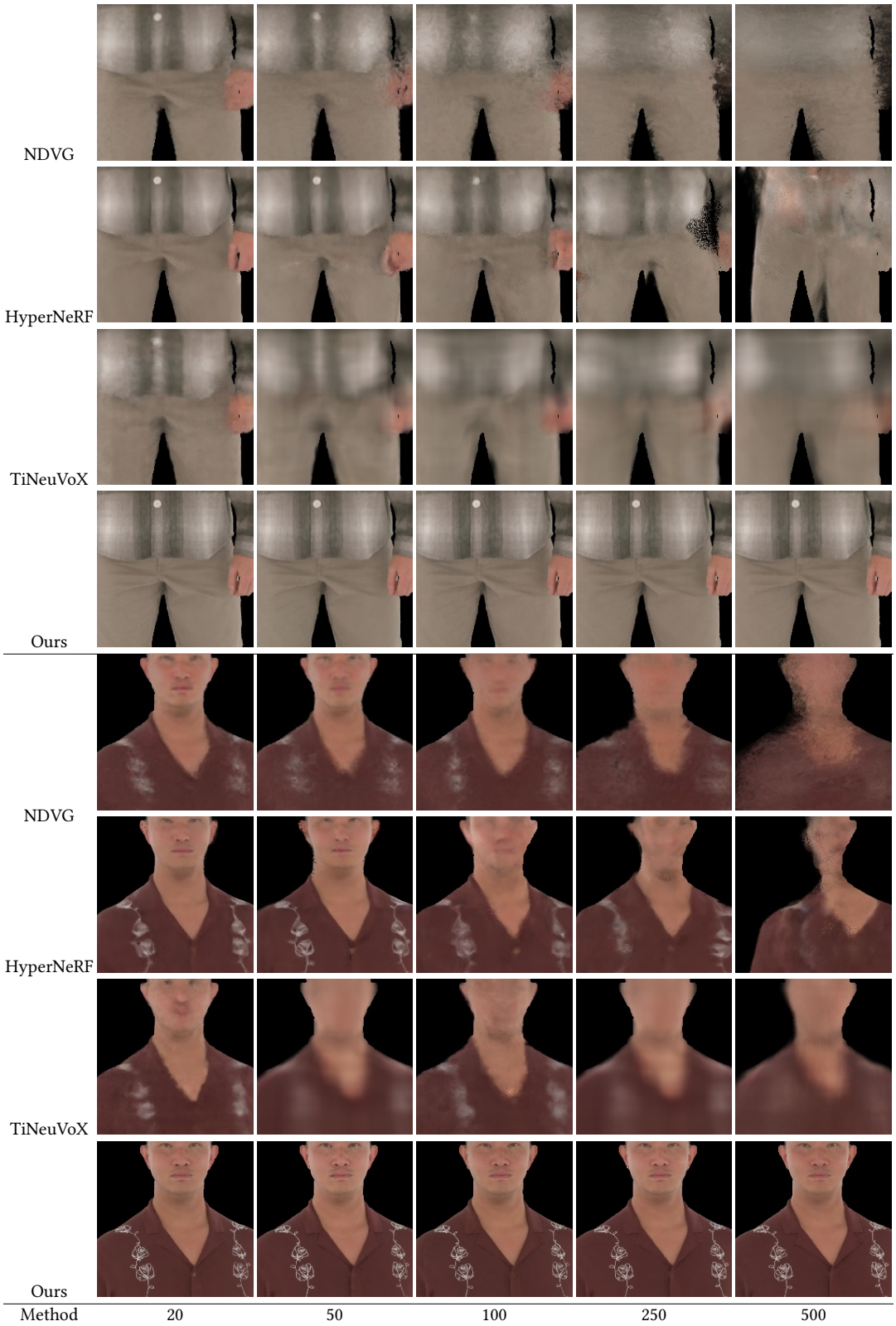

Fig. 4. **Impact of increasing sequence length**. For deformation-based baselines synthesis quality drops when rendering the same pose and frame while increasing the sequence length. Our results on the other hand have constant quality independent of the sequence length.

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.046	0.052	0.055	0.059	0.060	0.064
	↑ PSNR	31.51	31.45	30.53	30.42	30.23	30.31
	↑ SSIM	0.951	0.947	0.945	0.941	0.939	0.934
	↑ VMAF	83.10	84.43	86.90	85.27	85.47	87.41
Instant-NGP	↓ LPIPS	0.062	0.062	0.069	0.071	0.072	0.073
	↑ PSNR	30.60	29.87	29.71	29.65	29.56	29.64
	↑ SSIM	0.910	0.927	0.924	0.921	0.920	0.920
	↑ VMAF	73.27	69.36	73.84	73.95	75.41	76.38
TiNeuVox	↓ LPIPS	0.241	0.191	0.226	0.290	0.310	0.338
	↑ PSNR	27.78	26.55	25.36	23.61	22.36	20.50
	↑ SSIM	0.843	0.847	0.829	0.820	0.813	0.778
	↑ VMAF	65.61	64.23	59.43	43.08	31.88	21.92
NDVG	↓ LPIPS	0.121	0.168	0.213	0.276	0.313	0.361
	↑ PSNR	30.51	26.11	24.22	21.59	19.45	17.15
	↑ SSIM	0.904	0.867	0.841	0.803	0.786	0.736
	↑ VMAF	79.21	61.79	52.65	34.74	19.83	6.256
HyperNeRF	↓ LPIPS	0.186	0.178	0.206	0.222	0.269	0.301
	↑ PSNR	25.05	24.94	26.05	25.08	23.17	22.56
	↑ SSIM	0.868	0.864	0.854	0.844	0.820	0.806
	↑ VMAF	79.00	78.89	75.35	68.98	49.06	41.72
NeuralBody	↓ LPIPS	0.184	0.228	0.240	0.243	0.270	0.305
	↑ PSNR	28.87	26.23	27.63	25.79	25.38	26.74
	↑ SSIM	0.864	0.844	0.837	0.837	0.827	0.786
	↑ VMAF	48.10	46.36	41.70	46.55	43.24	35.05
TAVA	↓ LPIPS	0.149	0.178	0.210	0.259	0.308	0.373
	↑ PSNR	29.30	27.62	26.80	25.22	23.81	22.63
	↑ SSIM	0.879	0.868	0.852	0.827	0.803	0.770
	↑ VMAF	72.10	67.40	63.20	50.00	32.49	20.46

Table 4. Actor 1, Sequence 1

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.081	0.086	0.080	0.079	0.077	0.079
	↑ PSNR	30.67	30.21	30.54	29.63	29.81	29.54
	↑ SSIM	0.946	0.943	0.948	0.946	0.947	0.946
	↑ VMAF	85.23	84.66	87.98	85.73	85.58	87.73
Instant-NGP	↓ LPIPS	0.084	0.083	0.076	0.081	0.080	0.081
	↑ PSNR	29.60	26.99	27.99	27.75	27.76	27.77
	↑ SSIM	0.914	0.897	0.914	0.912	0.915	0.914
	↑ VMAF	76.35	81.46	81.37	81.31	81.65	81.57
TiNeuVox	↓ LPIPS	0.306	0.337	0.277	0.321	0.314	0.337
	↑ PSNR	25.09	21.33	22.48	21.05	20.05	19.57
	↑ SSIM	0.834	0.821	0.832	0.826	0.814	0.817
	↑ VMAF	46.73	21.32	33.18	20.47	12.83	11.13
NDVG	↓ LPIPS	0.299	0.286	0.300	0.333	0.363	0.372
	↑ PSNR	24.66	21.80	20.58	18.82	16.50	15.41
	↑ SSIM	0.841	0.827	0.823	0.791	0.762	0.753
	↑ VMAF	43.17	34.33	23.83	7.743	0.482	0.106
HyperNeRF	↓ LPIPS	0.223	0.194	0.223	0.244	0.273	0.291
	↑ PSNR	25.79	25.55	24.28	23.62	22.50	21.74
	↑ SSIM	0.849	0.876	0.865	0.851	0.842	0.833
	↑ VMAF	54.65	68.94	56.97	49.56	40.25	31.78
NeuralBody	↓ LPIPS	0.260	0.257	0.255	0.268	0.283	0.295
	↑ PSNR	27.55	25.48	26.89	24.94	24.58	25.76
	↑ SSIM	0.862	0.858	0.859	0.853	0.846	0.838
	↑ VMAF	46.52	46.26	39.96	43.91	37.46	31.37
TAVA	↓ LPIPS	0.278	0.272	0.283	0.331	0.362	0.391
	↑ PSNR	26.79	25.03	24.40	23.34	22.37	21.72
	↑ SSIM	0.860	0.857	0.855	0.830	0.814	0.794
	↑ VMAF	61.51	53.90	48.98	35.23	23.89	14.16

Table 5. Actor 2, Sequence 2

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.120	0.138	0.135	0.151	0.155	0.160
	↑ PSNR	31.02	30.26	30.25	28.98	29.50	29.19
	↑ SSIM	0.893	0.888	0.896	0.888	0.885	0.881
	↑ VMAF	72.05	71.11	75.40	78.37	76.96	78.22
Instant-NGP	↓ LPIPS	0.119	0.125	0.126	0.128	0.129	0.128
	↑ PSNR	29.44	29.22	29.28	29.23	29.10	29.32
	↑ SSIM	0.858	0.881	0.883	0.881	0.880	0.883
	↑ VMAF	62.37	60.88	67.69	68.24	68.54	70.39
TiNeuVox	↓ LPIPS	0.352	0.298	0.406	0.430	0.436	0.452
	↑ PSNR	27.51	26.62	24.13	22.98	22.30	21.28
	↑ SSIM	0.782	0.791	0.760	0.752	0.751	0.747
	↑ VMAF	49.37	51.83	29.76	24.86	19.28	12.17
NDVG	↓ LPIPS	0.240	0.281	0.354	0.435	0.453	0.481
	↑ PSNR	28.76	25.83	23.13	21.17	20.05	17.83
	↑ SSIM	0.841	0.812	0.763	0.731	0.724	0.692
	↑ VMAF	61.99	50.27	28.79	17.01	7.948	2.447
HyperNeRF	↓ LPIPS	0.233	0.250	0.275	0.322	0.374	0.388
	↑ PSNR	25.75	26.53	25.96	24.85	23.29	23.04
	↑ SSIM	0.827	0.818	0.800	0.777	0.758	0.761
	↑ VMAF	71.59	69.29	58.71	49.67	33.25	33.88
NeuralBody	↓ LPIPS	0.288	0.333	0.354	0.368	0.396	0.429
	↑ PSNR	27.51	25.88	27.18	25.30	24.81	25.68
	↑ SSIM	0.804	0.777	0.739	0.762	0.745	0.668
	↑ VMAF	42.89	42.13	34.00	33.25	26.65	21.11
TAVA	↓ LPIPS	0.261	0.303	0.341	0.410	0.467	0.504
	↑ PSNR	28.47	26.93	25.83	24.28	23.13	22.21
	↑ SSIM	0.820	0.801	0.782	0.749	0.721	0.704
	↑ VMAF	60.16	55.27	46.70	29.98	15.05	6.436

Table 6. Actor 3, Sequence 1

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.110	0.118	0.114	0.115	0.114	0.138
	↑ PSNR	27.52	27.60	27.55	27.28	27.04	26.42
	↑ SSIM	0.851	0.855	0.860	0.866	0.870	0.842
	↑ VMAF	90.93	91.46	90.47	90.80	90.27	89.87
Instant-NGP	↓ LPIPS	0.094	0.087	0.083	0.082	0.081	0.084
	↑ PSNR	26.70	25.66	26.68	26.48	26.53	26.53
	↑ SSIM	0.787	0.807	0.839	0.855	0.862	0.856
	↑ VMAF	83.50	87.28	87.09	87.17	87.39	87.59
TiNeuVox	↓ LPIPS	0.458	0.407	0.404	0.386	0.388	0.432
	↑ PSNR	22.62	20.78	19.39	18.62	17.43	15.81
	↑ SSIM	0.610	0.698	0.707	0.722	0.722	0.692
	↑ VMAF	51.25	39.78	30.67	29.42	25.38	15.91
NDVG	↓ LPIPS	0.483	0.421	0.414	0.428	0.443	0.454
	↑ PSNR	20.95	18.16	16.99	15.27	13.03	12.54
	↑ SSIM	0.608	0.662	0.669	0.662	0.632	0.625
	↑ VMAF	56.20	34.25	24.36	16.40	10.35	8.543
HyperNeRF	↓ LPIPS	0.394	0.297	0.309	0.322	0.335	0.366
	↑ PSNR	23.50	23.44	22.13	21.16	20.17	19.53
	↑ SSIM	0.651	0.739	0.742	0.747	0.745	0.727
	↑ VMAF	78.24	71.99	66.48	55.13	42.85	40.66
NeuralBody	↓ LPIPS	0.454	0.377	0.372	0.367	0.392	0.423
	↑ PSNR	25.02	23.10	24.20	22.82	22.62	22.05
	↑ SSIM	0.639	0.722	0.729	0.740	0.723	0.703
	↑ VMAF	54.05	50.99	41.15	43.07	27.27	24.65
TAVA	↓ LPIPS	0.431	0.359	0.375	0.416	0.452	0.500
	↑ PSNR	24.97	23.22	22.36	21.01	19.84	18.87
	↑ SSIM	0.644	0.724	0.727	0.712	0.683	0.640
	↑ VMAF	72.20	56.97	48.24	29.85	12.28	3.305

Table 7. Actor 4, Sequence 2

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.067	0.073	0.073	0.075	0.082	0.083
	↑ PSNR	32.01	31.22	31.04	30.47	30.18	30.29
	↑ SSIM	0.950	0.946	0.946	0.944	0.939	0.937
	↑ VMAF	78.99	77.68	79.22	78.70	79.40	81.50
Instant-NGP	↓ LPIPS	0.098	0.090	0.095	0.091	0.092	0.090
	↑ PSNR	29.47	29.40	29.55	29.58	29.62	29.79
	↑ SSIM	0.903	0.924	0.919	0.923	0.921	0.923
	↑ VMAF	66.84	63.65	67.78	69.37	70.21	71.07
TiNeuVox	↓ LPIPS	0.197	0.272	0.284	0.293	0.312	0.296
	↑ PSNR	28.20	26.43	25.38	24.35	23.38	22.11
	↑ SSIM	0.852	0.837	0.828	0.826	0.816	0.791
	↑ VMAF	64.80	57.06	51.81	45.29	37.90	29.19
NDVG	↓ LPIPS	0.176	0.175	0.187	0.254	0.278	0.329
	↑ PSNR	29.80	27.89	27.41	23.50	22.03	19.14
	↑ SSIM	0.894	0.892	0.878	0.829	0.806	0.762
	↑ VMAF	72.32	69.70	64.46	39.65	31.08	15.88
HyperNeRF	↓ LPIPS	0.237	0.234	0.223	0.223	0.253	0.278
	↑ PSNR	25.29	25.14	25.74	25.40	24.87	23.97
	↑ SSIM	0.839	0.838	0.845	0.846	0.828	0.816
	↑ VMAF	69.36	70.11	69.02	66.22	57.84	47.33
NeuralBody	↓ LPIPS	0.289	0.283	0.270	0.291	0.316	0.336
	↑ PSNR	26.91	26.10	28.18	25.27	24.77	26.77
	↑ SSIM	0.826	0.823	0.836	0.820	0.811	0.802
	↑ VMAF	39.81	37.83	40.60	36.36	25.49	29.24
TAVA	↓ LPIPS	0.208	0.213	0.227	0.269	0.324	0.373
	↑ PSNR	28.13	27.03	26.83	25.76	24.33	23.15
	↑ SSIM	0.857	0.852	0.849	0.835	0.808	0.782
	↑ VMAF	64.42	63.31	59.15	49.24	36.51	24.45

Table 8. Actor 5, Sequence 1

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.133	0.128	0.123	0.118	0.117	0.121
	↑ PSNR	27.26	28.10	27.80	27.68	27.69	27.33
	↑ SSIM	0.899	0.908	0.913	0.916	0.915	0.913
	↑ VMAF	85.85	89.94	88.05	88.09	88.18	86.53
Instant-NGP	↓ LPIPS	0.079	0.090	0.099	0.093	0.094	0.096
	↑ PSNR	28.18	28.38	28.13	27.57	27.40	27.53
	↑ SSIM	0.907	0.914	0.906	0.908	0.905	0.904
	↑ VMAF	76.56	76.17	79.68	79.01	79.44	79.31
TiNeuVox	↓ LPIPS	0.258	0.296	0.311	0.351	0.356	0.367
	↑ PSNR	24.68	22.13	21.62	19.57	18.86	18.44
	↑ SSIM	0.834	0.799	0.802	0.803	0.799	0.791
	↑ VMAF	55.69	44.06	39.54	23.80	14.92	15.44
NDVG	↓ LPIPS	0.244	0.307	0.344	0.343	0.413	0.415
	↑ PSNR	23.79	20.35	19.17	17.70	14.90	13.71
	↑ SSIM	0.839	0.795	0.778	0.764	0.707	0.697
	↑ VMAF	57.39	38.60	26.81	12.92	2.790	3.058
HyperNeRF	↓ LPIPS	0.197	0.243	0.262	0.276	0.321	0.327
	↑ PSNR	25.45	23.61	23.01	22.32	20.52	20.54
	↑ SSIM	0.866	0.843	0.839	0.835	0.811	0.807
	↑ VMAF	73.33	73.32	68.97	55.20	39.83	36.24
NeuralBody	↓ LPIPS	0.278	0.330	0.342	0.333	0.366	0.403
	↑ PSNR	25.18	23.35	25.81	23.58	23.20	25.21
	↑ SSIM	0.847	0.807	0.814	0.820	0.801	0.770
	↑ VMAF	51.62	46.30	46.14	38.76	32.29	19.65
TAVA	↓ LPIPS	0.258	0.330	0.325	0.374	0.436	0.460
	↑ PSNR	25.27	23.35	23.35	21.98	20.19	19.86
	↑ SSIM	0.849	0.807	0.824	0.797	0.751	0.731
	↑ VMAF	65.48	46.30	56.80	36.29	12.33	6.512

Table 9. Actor 6, Sequence 2

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.093	0.096	0.089	0.096	0.102	0.103
	↑ PSNR	31.76	31.56	31.04	30.55	30.69	29.95
	↑ SSIM	0.940	0.938	0.943	0.937	0.935	0.929
	↑ VMAF	82.38	84.74	85.12	85.40	86.37	85.45
Instant-NGP	↓ LPIPS	0.122	0.108	0.107	0.108	0.109	0.107
	↑ PSNR	30.03	30.38	30.13	30.14	30.13	30.31
	↑ SSIM	0.867	0.909	0.913	0.915	0.914	0.916
	↑ VMAF	73.15	69.42	74.06	74.24	74.28	74.17
TiNeuVox	↓ LPIPS	0.358	0.246	0.314	0.321	0.289	0.356
	↑ PSNR	26.89	26.84	25.16	24.48	24.07	22.34
	↑ SSIM	0.800	0.830	0.819	0.818	0.808	0.799
	↑ VMAF	57.94	65.20	49.75	42.40	39.87	27.73
NDVG	↓ LPIPS	0.243	0.222	0.232	0.282	0.304	0.334
	↑ PSNR	29.37	27.94	26.29	22.52	21.38	19.11
	↑ SSIM	0.861	0.868	0.854	0.811	0.797	0.766
	↑ VMAF	78.65	74.62	59.49	35.19	26.32	12.68
HyperNeRF	↓ LPIPS	0.255	0.228	0.240	0.259	0.280	0.319
	↑ PSNR	26.88	26.92	26.00	24.94	24.47	22.17
	↑ SSIM	0.828	0.851	0.841	0.827	0.812	0.792
	↑ VMAF	77.46	80.60	76.25	63.25	55.51	33.48
NeuralBody	↓ LPIPS	0.328	0.312	0.294	0.309	0.325	0.355
	↑ PSNR	25.84	25.37	28.50	25.17	25.69	27.44
	↑ SSIM	0.793	0.811	0.829	0.818	0.808	0.796
	↑ VMAF	41.02	42.56	47.78	32.95	34.04	31.75
TAVA	↓ LPIPS	0.253	0.252	0.263	0.314	0.349	0.392
	↑ PSNR	28.60	27.57	26.55	25.22	24.57	23.46
	↑ SSIM	0.835	0.843	0.837	0.814	0.796	0.773
	↑ VMAF	69.11	64.59	59.55	45.76	36.82	22.89

Table 10. Actor 7, Sequence 1

Method	Metric	20	50	100	250	500	1000
Ours	↓ LPIPS	0.107	0.112	0.108	0.109	0.109	0.108
	↑ PSNR	30.62	30.02	29.94	29.04	29.59	29.40
	↑ SSIM	0.917	0.917	0.920	0.920	0.921	0.921
	↑ VMAF	90.80	91.42	91.87	89.89	90.39	89.18
Instant-NGP	↓ LPIPS	0.102	0.088	0.093	0.089	0.088	0.087
	↑ PSNR	31.60	30.31	30.20	29.79	29.74	29.90
	↑ SSIM	0.900	0.925	0.917	0.920	0.920	0.922
	↑ VMAF	81.11	77.64	82.05	80.89	81.28	80.35
TiNeuVox	↓ LPIPS	0.326	0.389	0.392	0.379	0.379	0.387
	↑ PSNR	25.80	22.43	21.39	19.88	19.06	18.32
	↑ SSIM	0.780	0.778	0.773	0.770	0.768	0.761
	↑ VMAF	58.70	34.98	28.05	19.57	15.84	11.42
NDVG	↓ LPIPS	0.342	0.342	0.354	0.354	0.367	0.380
	↑ PSNR	24.98	21.14	19.48	17.40	16.10	14.42
	↑ SSIM	0.797	0.766	0.740	0.725	0.716	0.698
	↑ VMAF	57.17	40.64	24.69	11.41	3.313	0.487
HyperNeRF	↓ LPIPS	0.275	0.259	0.271	0.288	0.312	0.328
	↑ PSNR	27.88	25.73	24.63	23.21	21.67	20.57
	↑ SSIM	0.833	0.831	0.821	0.807	0.790	0.782
	↑ VMAF	81.03	71.27	65.61	52.05	40.14	30.99
NeuralBody	↓ LPIPS	0.363	0.342	0.358	0.364	0.372	0.388
	↑ PSNR	29.31	25.81	26.99	24.38	23.71	24.99
	↑ SSIM	0.817	0.813	0.801	0.792	0.787	0.771
	↑ VMAF	49.76	49.72	43.29	34.87	30.58	22.76
TAVA	↓ LPIPS	0.321	0.311	0.338	0.383	0.409	0.438
	↑ PSNR	27.99	25.23	24.30	22.10	20.95	20.07
	↑ SSIM	0.815	0.816	0.803	0.771	0.747	0.722
	↑ VMAF	70.40	61.50	51.62	30.88	18.23	8.532

Table 11. Actor 8, Sequence 2

REFERENCES

Ang Cao and Justin Johnson. 2023. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR* (2023).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. 2022. NerfAcc: A General NeRF Accleration Toolbox. *arXiv preprint arXiv:2210.04847* (2022).

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*.

Thomas Müller. 2021. *tiny-cuda-nn*. https://github.com/NVlabs/tiny-cuda-nn Accessed: 2022-10-21.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

Jiaxiang Tang. 2022. Torch-ngp: a PyTorch implementation of instant-ngp. https://github.com/ashawkey/torch-ngp.