Artemis: Articulated Neural Pets with Appearance and Motion Synthesis

HAIMIN LUO, ShanghaiTech University, China
TENG XU, ShanghaiTech University, China
YUHENG JIANG, ShanghaiTech University, China
CHENGLIN ZHOU, ShanghaiTech University, China
QIWEI QIU, ShanghaiTech University and Deemos Technology Co., Ltd., China
YINGLIANG ZHANG, DGene Digital Technology Co., Ltd., China
WEI YANG, Huazhong University of Science and Technology, China
LAN XU, ShanghaiTech University, China
JINGYI YU*, ShanghaiTech University, China

Fig. 1. **Demonstration of our neural pet in a VR environment**. We present an approach, we call "Artemis", for generating photo-realistic and interactable virtual pets. Traditional approaches animate animals with rigged mesh models and skeleton-based skinning techniques, but they can not handle furs well. Instead, we represent animals as an animatable neural volume and render animal appearance and furs in real-time (see the right figure for rendering results). Moreover, we use the motion synthesis approach, i.e., the Local Motion Phase, to generate skeletal motion for the animal according to the user's control signals. We demonstrate this concept in the left figure, where the user gives a destination point, and the virtual lion follows the skeleton and motion and moves to the destination.

*corresponding author

Authors' addresses: Haimin Luo, ShanghaiTech University, China, luohm@shanghaitech.edu.cn; Teng Xu, ShanghaiTech University, China, xuteng@shanghaitech.edu.cn; Yuheng Jiang, ShanghaiTech University, China, jiangyh2@shanghaitech.edu.cn; Chenglin Zhou, ShanghaiTech University, China, zhouchl@shanghaitech.edu.cn; Qiwei Qiu, ShanghaiTech University and Deemos Technology Co., Ltd., China, qiuqw@shanghaitech.edu.cn; Yingliang Zhang, DGene Digital Technology Co., Ltd., China, yingliang.zhang@dgene.com; Wei Yang, Huazhong University of Science and Technology, China, weiyangcs@hust.edu.cn; Lan Xu, ShanghaiTech University, China, xulan1@shanghaitech.edu.cn; Jingyi Yu, ShanghaiTech University, China, yujingyi@shanghaitech.edu.cn.

We, humans, are entering into a virtual era and indeed want to bring animals to the virtual world as well for companion. Yet, computer-generated (CGI) furry animals are limited by tedious off-line rendering, let alone interactive motion control. In this paper, we present ARTEMIS, a novel neural modeling and rendering pipeline for generating ARTiculated neural pets with appEarance and Motion synthesIS. Our ARTEMIS enables interactive motion control, real-time animation, and photo-realistic rendering of furry animals. The core of our ARTEMIS is a neural-generated (NGI) animal engine, which adopts an efficient octree-based representation for animal animation and fur rendering. The animation then becomes equivalent to voxel-level deformation based on explicit skeletal warping. We further use a fast octree indexing and efficient volumetric rendering scheme to generate appearance

0730-0301/2022/7-ART164
https://doi.org/10.1145/3528223.3530086

and density features maps. Finally, we propose a novel shading network to generate high-fidelity details of appearance and opacity under novel poses from appearance and density feature maps. For the motion control module in ARTEMIS, we combine state-of-the-art animal motion capture approach with recent neural character control scheme. We introduce an effective optimization scheme to reconstruct the skeletal motion of real animals captured by a multi-view RGB and Vicon camera array. We feed all the captured motion into a neural character control scheme to generate abstract control signals with motion styles. We further integrate ARTEMIS into existing engines that support VR headsets, providing an unprecedented immersive experience where a user can intimately interact with a variety of virtual animals with vivid movements and photo-realistic appearance. Extensive experiments and showcases demonstrate the effectiveness of our ARTEMIS system in achieving highly realistic rendering of NGI animals in real-time, providing daily immersive and interactive experiences with digital animals unseen before. We make available our ARTEMIS model and dynamic furry animal dataset at https://haiminluo.github.io/publication/artemis/.

CCS Concepts: • **Computing methodologies** → **Image-based rendering**; **Motion capture**; **Volumetric models**.

Additional Key Words and Phrases: neural volumetric animal, novel view syntheis, neural rendering, neural representation, dynamic scene modeling, motion synthesis

ACM Reference Format:
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 (July 2022), 19 pages. https://doi.org/10.1145/3528223.3530086

1 INTRODUCTION

Our love for animals is a great demonstration of humanity. Digitization of fascinating animals such as Aslan the Lion in the Chronicles of Nardia to Richard Parker the Tiger in Life of Pie brings close encounters to viewers, unimaginable in real life. Despite considerable successes in feature films, accessing computer-generated imagery (CGI) digital animals and subsequently animating and rendering them at high realism have by far been a luxury of VFX studios where the process of creating them requires tremendous artists' efforts and high computational powers, including the use of render farms. In fact, even with ample resources, photo-realistic rendering of animated digital animals still remains off-line and hence is not yet ready for primetime on Metaverse, where a user should be able to guide the movement of the animal, interact with it, and make close observations, all at high photo-realism in real-time.

The challenges are multi-fold, but the conflict demand between real-time and photo-realistic rendering is at the core. First, animals are covered with furs traditionally modeled with hundreds of thousands of hair fiber/strands, using videos and photos as references. The traditional modeling process is tedious and requires exquisite artistic skills and excessive labor. The rendering process, on the other hand, is equally time-consuming: off-line ray tracing is typically adopted to produce photo-realism in translucency, light scattering, volumetric shadows, etc, and far from real-time even with the most advanced graphics hardware. In addition to rendering, animating the model at an interactive speed and with high realism remains challenging. For example, animations of animals in the latest remake of the Lion King were crafted by hand, based on the reference clips to match skeletal and muscle deformations. An automated tool is in urgent need for facilitating motion controls and even transfers (e.g., from dogs to wolves). Finally, for digital animals to thrive in the virtual world, they should respond to users' instructions, and therefore, both the rendering and animation components need tight integration with interaction.

In this paper, we address these critical challenges by presenting *ARTEMIS*, a novel neural modeling and rendering framework for generating *ARTiculated neural pets with appEarance and Motion synthesIS*. In stark contrast with existing off-line animation and rendering systems, ARTEMIS supports interactive motion control, realistic animation, and high-quality rendering of furry animals, all in real-time. Further, we extend ARTEMIS to OpenVR to support consumer-level VR headsets, providing a surreal experience for users to intimately interact with various virtual animals as if in the real world (see Fig. 1).

On appearance modeling and rendering, ARTEMIS presents a neural-generated imagery (NGI) production pipeline to replace the traditional realistic but time-consuming off-line CGI process to model animated animals. NGI is inspired by the Neural Radiance Field (NeRF) [Mildenhall et al. 2020], and various forms of its extensions for producing photo-realistic rendering of real-world objects [Peng et al. 2021a; Su et al. 2021; Tretschk et al. 2021; Zhang et al. 2021]. These approaches, however, cannot yet handle elastic motions while maintaining visual realism. In our case, we train on CGI animal assets (dense multi-view RGBA videos rendered using Blender) under a set of pre-defined motion sequences, where each animal model contains a skeleton with Linear Blend Skinning (LBS) weights. Using Spherical Harmonics factorization, we embed the animal's appearance in a high-dimensional latent space. Specifically, we follow the same strategy as PlenOctree [Yu et al. 2021] and build a canonical voxel-based octree for the whole sequence while maintaining the latent descriptors and the density field. Under this design, we can apply LBS skeletal warping to locate features under the canonical pose on live poses. To ensure image quality, we further add a convolutional decoder to enhance spatial texture details of furs. The complete process can be trained as a differentiable renderer where the result enables the real-time, photo-realistic, free-viewpoint rendering of dynamic animals.

Besides rendering, ARTEMIS provides realistic motion synthesis so that a user can control the animal in virtual space. We combine state-of-the-art animal motion capture techniques with the recent neural character control [Starke et al. 2020]. Since there still lacks a comprehensive 3D mocap database, we collect several new datasets using a hybrid RGB/Vicon mocap dome. Similar to human pose priors [Pavlakos et al. 2019], we pre-train an animal pose prior from the data via Variational AutoEncoder (VAE) and use it as a regularizer for motion estimation. The resulting motion sequences are used to drive neural character animations [Holden et al. 2017; Starke et al. 2019, 2020, 2021; Zhang et al. 2018]. In particular, we employ the local motion phase (LMP) technique [Starke et al. 2020] to guide the movement of the neural animal under common commands by real-world pet owners. Since both neural rendering and motion controls of ARTEMIS achieve real-time performance, we further integrate ARTEMIS into existing consumer-level VR headset platforms. Specifically, we fine-tune our neural training process for binocular rendering and develop a hybrid rendering

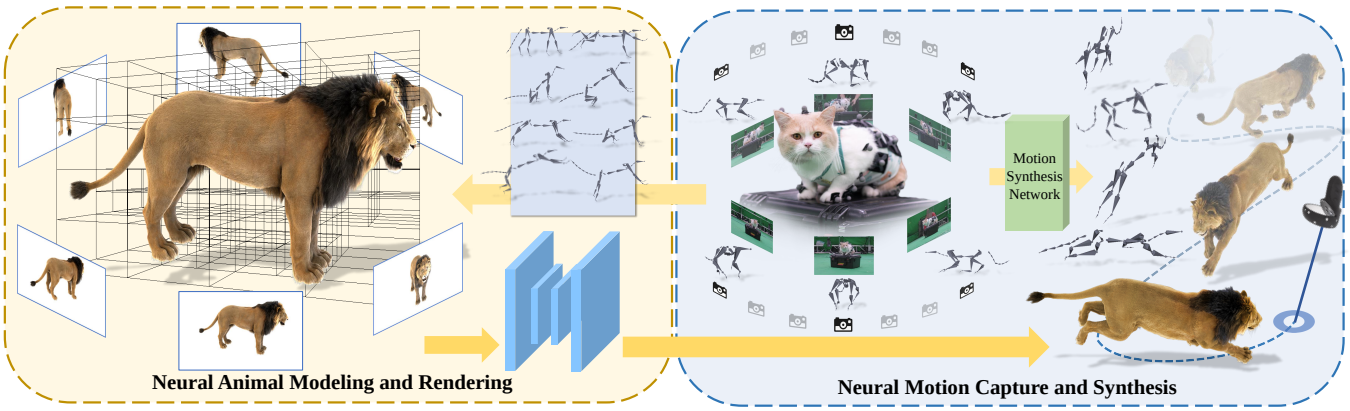

Fig. 2. **Overview of our ARTEMIS system for generating articulated neural pets with appearance and motion synthesis.** ARTEMIS consists of two core components. In the first module, given skeletal rig and skinning weights of CGI animal assets and corresponding multi-view rendered RGBA images in representative poses, we build a dynamic octree-based neural representation to enable explicit skeletal animation and real-time rendering for dynamic animals, which supports real-time interactive applications; In the second module, we build a hybrid animal motion capture system with multi-view RGB and VICON cameras to reconstruct realistic 3D skeletal poses, which supports to training a neural motion synthesis network to enable a user to interactively guide the movement of the neural animals. The ARTEMIS system is further integrated into existing consumer-level VR headset platforms for immersive VR experience of neural-generated animals.

scheme so that the foreground NGI results can be fused with the rasterization-produced background. Comprehensive experiments show that ARTEMIS produces highly realistic virtual animals with convincing motions and appearance. Coupled with a VR headset, it provides a surreal experience where users can intimately interact with a variety of digital animals in close encounters.

To summarize, our main contributions include:

- We propose a novel neural modeling and rendering system, ARTEMIS, that supports natural motion controls and user interactions with virtual animals on 2D screens or in VR settings. In particular, we collect a new motion capture dataset on animals of different scales and train a new skeleton detector.
- We present a differentiable neural representation tailored for modeling dynamic animals with furry appearances. Our new approach can effectively convert traditional CGI assets to NGI assets to achieve real-time and high-quality rendering.
- We provide controllable motion synthesis schemes for various NGI animals under the VR setting. A user can send commands to or intimately interact with the virtual animals as if they were real.

2 RELATED WORK

2.1 Fuzzy object modeling and rendering

Traditional methods focus on physical simulation-based and image-based modeling to model fuzzy objects like human hair and animal fur. Physical simulation-based modeling methods directly represent hair as existing physical models. Hair rendering can date back to texel object method [Kajiya and Kay 1989], which proposed anisotropic lighting model object to be rendered as furry surfaces. This model is still used in some modern CG rendering engines. Light scattering model [Marschner et al. 2003] of hair is then proposed, which modeling hair lighting with two specular layer effects, and is improved by considering energy-conserving [d'Eon et al. 2011]. A more complex microcosmic hair model is proposed, which adds a medulla kernel in hair and fur [Yan et al. 2015] and is accelerated by directly performing path tracing [Chiang et al. 2015]. Mass spring model [Selle et al. 2008] is proposed to simulate hair which represents straight hair better. Floating Tangents algorithm [Derouet-Jourdan et al. 2013] approximates hair curves with helices model. Grid hair model starts from NVIDIA HairWorks technology and is improved to mesh model which provides direct control of overall hair shape [Yuksel et al. 2009]. On the other hand, image-based modeling methods try to reconstruct hair models directly from multi-view or single view images. Multi-view images based hair reconstruction relies on direction field and hair tracking growth algorithm [Bao and Qi 2016; Luo et al. 2013; Paris et al. 2004, 2008; Wei et al. 2005]. Single view image-based hair reconstruction, which relies on direction field and needs hair guideline as input [Chai et al. 2015; Ding et al. 2016], the data-driven based method rely on hair model database to perform hair information matching [Chai et al. 2016; Hu et al. 2015].

Besides human hair, other methods focus on reconstructing furry objects. IBOH uses the multi-background matting method to construct opacity hull, which represents furry objects as opacity point clouds [Matusik et al. 2002]. Neural representation methods are also proved to perform well on furry objects like hair and fur. NOPC method implements a neural network algorithm to get similar opacity point clouds in neural network representation [Wang et al. 2020], which can efficiently generate high-quality alpha maps even with low-quality reconstruction from small data. The latter work ConvNeRF changes point clouds representation to opacity radiance field [Luo et al. 2021], which enables high quality, global-consistent, and free-viewpoint opacity rendering for fuzzy objects.

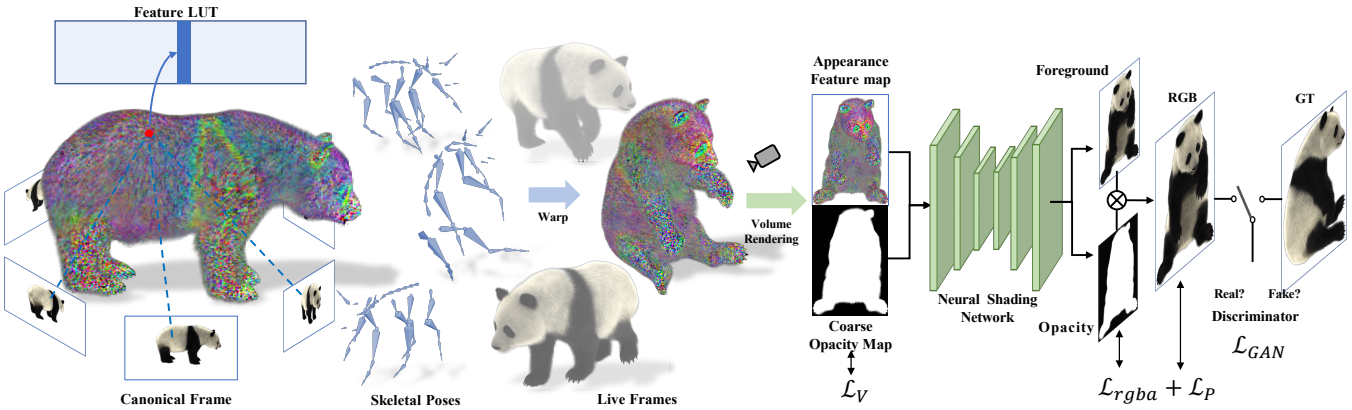

Fig. 3. **The algorithm pipeline of our Animatable Neural Animals.** Given multi-view RGBA images of traditional modeled animals rendered in canonical space, we first extract a sparse voxel grid and allocate a corresponding feature look-up table as a compact representation, together with an octree for quick feature indexing. Then we pose the character to training poses using the rig of the traditional animal asset and conduct efficient volumetric rendering to generate view-dependent appearance feature maps and coarse opacity maps. We next decode them into high-quality appearance and opacity images with the convolutional neural shading network. We further adopt an adversarial training scheme for high-frequency details synthesis.

Recall all furry objects modeling method, it is seen that traditional physical modeling and rendering method is able to generate high-quality method, but cost much times computation than simple mesh due to the complex ray reflection property of hair and fur, which is difficult to perform real-time rendering. Image-based methods directly reconstruct hair model from images, which is more efficient to generate static hair model but lose view consistency and cost much on dynamic scene effects. Recent neural representation methods perform well on fuzzy object modeling and rendering but still cannot generate real-time dynamic hair and fur effects.

2.2 Virtual animals

Animal parametric models have gradually been investigated in recent years, corresponding to general human parametric models. A series of works are proposed to capture and reconstruct animals like dogs [Biggs et al. 2020], horses [Zuffi et al. 2019], birds [Kanazawa et al. 2018] and other medium animals [Biggs et al. 2018; Cashman and Fitzgibbon 2012; Zuffi et al. 2017]. The seminal work [Cashman and Fitzgibbon 2012] of Cashman and Fitzgibbon learns a low-dimensional 3D morphable animal model by estimating the shape of dolphins from images. This approach was limited to specific classes(dolphins, pigeons) and suffered from an overly smooth shape representation. [Kanazawa et al. 2016] learns a model to deform the template model to match hand clicked correspondences. However, their model is also animal-specific. SMAL [Zuffi et al. 2017] extends SMPL [Loper et al. 2015] model to the animal domain; they create a realistic 3D model of animals and fit this model to 2D data, overcoming the lack of motion and shape capture for animal subjects with prior knowledge. [Kanazawa et al. 2018] combine the benefits of the classically used deformable mesh representations with a learning-based prediction mechanism. [Kearney et al. 2020] built the first open-source 3D motion capture dataset for dogs using RGBD cameras and Vicon optical motion capture system. Comparably, our method builds a hybrid capture system with marker-based capture for docile and small pets(e.g., dogs and cats) and only RGB footage for dangerous and large animals. Besides, we reconstruct the realistic 3D skeletal poses for each kind of animal above.

2.3 Neural Modeling and Rendering

Various explicit representations have been incorporated into the deep learning pipeline for geometry and appearance learning via differentiable renderings, such as pointclouds [Aliev et al. 2020; Anqi et al. 2021; Insafutdinov and Dosovitskiy 2018; Kolos et al. 2020; Lin et al. 2018; Roveri et al. 2018; Wu et al. 2020; Yifan et al. 2019], textured meshes [Chen et al. 2019; Habermann et al. 2021; Kato et al. 2018; Liu et al. 2020b, 2019; Shysheya et al. 2019] and volumes [Lombardi et al. 2019; Mescheder et al. 2019; Peng et al. 2020; Sitzmann et al. 2019]. However, they suffer from holes, topological changes, or cubic memory footprint. Recent implicit representations employ Multi-Layer Perceptrons (MLP) to learn a continuous implicit function that maps spacial locations to distance field [Chabra et al. 2020; Jiang et al. 2020; Park et al. 2019], occupancy field [He et al. 2021; Huang et al. 2020; Mescheder et al. 2019; Peng et al. 2020; Saito et al. 2019] or radiance field [Bi et al. 2020; Liu et al. 2020a; Martin-Brualla et al. 2021; Mildenhall et al. 2020]. They can naturally handle complicated scenes, but they usually suffer from high query time for rendering. For real-time rendering, explicit primitives [Lombardi et al. 2021] or spheres [Lassner and Zollhofer 2021] representations are employed, together with acceleration data structures [Yu et al. 2021]. However, they require good initialization or high memory for dynamic scenes. MonoPort [Li et al. 2020a,b] achieves impressive real-time reconstruction and rendering with an RGBD camera by combining PiFu [Saito et al. 2019] and octree for fast surface localization, but it does not support animation and suffers from insufficient rendering quality.

Recently NeRF [Mildenhall et al. 2020] has achieved impressive progress by learning radiance fields from 2D images. The following variants of NeRF aim to learn generalizable radiance fields [Chen

et al. 2021; Wang et al. 2021a], train with unposed cameras [Meng et al. 2021; Wang et al. 2021b; Yen-Chen et al. 2021], model high-frequency details [Luo et al. 2021; Sitzmann et al. 2020; Tancik et al. 2020] and opacity [Luo et al. 2021], handle relighting and shading [Bi et al. 2020; Boss et al. 2021; Kuang et al. [n.d.]] or accelerate for real-time applications [Garbin et al. 2021; Hedman et al. 2021; Neff et al. 2021; Reiser et al. 2021; Yu et al. 2021]. Most recent approaches extend NeRF to dynamic scenes by learning a deformation field [Li et al. 2021; Park et al. 2021a; Pumarola et al. 2021; Tretschk et al. 2021; Xian et al. 2021] or training a hypernet [Park et al. 2021b]. Such methods can only achieve playback or implicit interpolation. With estimated skeleton motion and parametric model, AnimatableNeRF [Peng et al. 2021a] achieves pose control by inversely learning blending weights fields; however, it struggles for large motion due to the limited capacity of MLPs. NeuralBody [Peng et al. 2021b] proposes structured latent codes for dynamic human performance rendering and is then extended to be more pose-generalizable by adopting pixel-aligned features [Kwon et al. 2021]. H-NeRF [Xu et al. 2021] constrains a radiance field by a structured implicit human body model to robustly fuse information from sparse views for better pose generalization. NeuralActor [Liu et al. 2021] further introduces dynamic texture to neural radiance fields for better pose-dependent details. These works, however, still suffer from high query time consumption without acceleration. Another line of research employs explicit volume [Lombardi et al. 2019], points [Wu et al. 2020], mesh [Habermann et al. 2021; Liu et al. 2020b, 2019; Shysheya et al. 2019], but suffers from low resolution representations. MVP [Lombardi et al. 2021] achieves real-time fine detail rendering but still cannot realize pose extrapolation.

3 ANIMATABLE NEURAL ANIMALS

Animating digital animals have long relied on experienced artists. To produce photo-realistic appearance and opacity of furry animals, it is commonly a group effort to model tens of thousands of hair fibers and simulate their movements with physical soundness. Finally, real-time rendering is difficult to achieve even on high-end render farms. For interactive applications such as virtual pets, it is essential to simultaneously maintain photo-realism and real-time performance on both motion and appearance.

Neural Opacity Radiance Fields. In contrast to physically modeling translucency of furs, we adopt the neural rendering approach and cast the problem as view synthesis on the radiance field. The seminal work of the Neural Radiance Field conceptually provides a natural solution to fur rendering: NeRF represents a scene in terms of the color and density on each point along a ray where the density naturally reflects the opacity of the point. However, the alpha matte produced from NeRF tends to be noisy and less continuous than natural fur or hair. The recent ConvNeRF [Luo et al. 2021] addresses this issue by processing the image in feature space than directly in RGB colors. They formulate a neural opacity field as:

$$I = \Phi(F_p), \;\; F_p = \sum_{i}^{|\mathcal{S}|} T_i \big(1 - \exp(-\sigma_i \delta_i)\big) f_i \tag{1}$$

where Φ is the opacity synthesis network, and F_p is the aggregated image features, $T_i = \exp(-\sum_{j=1}^{i} \sigma_j \delta_j)$. $\mathcal{S}$ is the set of sampled points, and σ_i, δ_i are density and distance between sampled points, respectively. The key here is the use of feature f_i to represent the opacity at each spatial point instead of directly using the density. The original ConvNeRF can only handle static objects. The brute-force approach is to conduct per-frame optimization for dynamic objects, which is clearly infeasible on neural animals that generally perform long motion sequences.

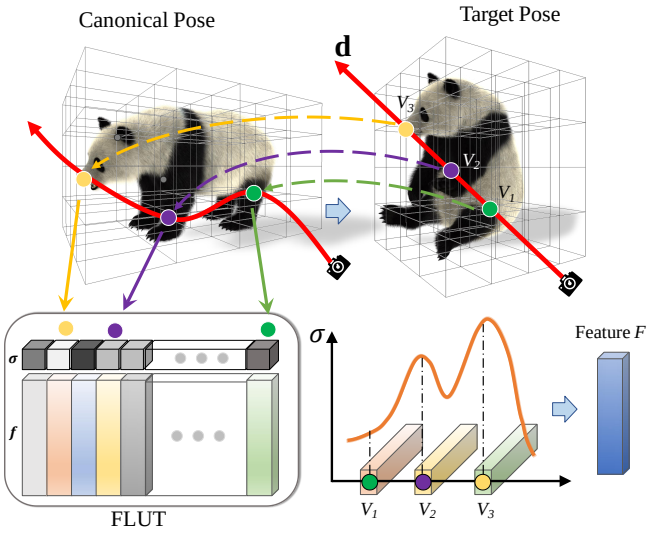

Fig. 4. **Dynamic Animal Rendering Details**. Given a target skeletal pose, we deform the canonical index octree to live frames, then perform ray marching for efficient volume rendering to integrate the features along the ray to generate feature maps for the further rendering process.

3.1 Animatable Neural Volumes

We first extend the neural opacity radiance field to dynamic animals. Notice that our goal is not only to accelerate training but, more importantly, to realize real-time rendering. Inspired by PlenOctree [Yu et al. 2021] for static scenes, we set out to store the opacity features in a volumetric octree structure for real-time rendering but under animations. Specifically, we devise a feature indexing scheme for quick feature fetching and rendering. We further introduce a skeleton-based volume deformation scheme that employs skinning weights derived from the original CGI model to bridge the canonical frame with live frames for animation. In addition, we devise a neural shading network to handle animated object modeling and adopt an efficient adversarial training scheme for model optimization.

Octree Feature Indexing. Unlike the original NeRF or PlenOctree, where the object's geometry is unknown, we have the CGI animal models as inputs. Therefore, we first convert a CGI animal character, such as a tiger or a lion, to the octree-based representation. Again, the original CGI model contains very detailed furs, and direct conversion onto discrete voxels can cause strong aliasing and significant errors in subsequent neural modeling. If we remove the furs and use only the bare model, the voxel representation will deviate significantly from the actual ones. Instead, we prefer "dilated" voxel representations that encapsulate the model. We apply a simple trick

to resolve this issue: we use the rendered alpha matte from a dense set of views as inputs and conduct conservative volume carving to construct the octree: we initialize a uniform volume and carve it using the masks dilated from the alpha mattes. Notice, though, that we also need the rendered multi-view alpha matte later for training the neural opacity field. The resulting octree contains a voxel array **P** occupied in 3D space. Using this volumetric representation, we aim to store a view-dependent feature f at each voxel. We, therefore, allocate a separate data array **F** called Features Look-up Table (FLUT) to store features and density values as in the original PlenOctree. Here FLUT is used to efficiently query features at an arbitrary 3D location to accelerate training and inference. For a given query point in space at the volume rendering process, we can index into FLUT in constant time and assign the corresponding feature and density to the point.

Inspired by PlenOctree [Yu et al. 2021] that factorizes view-dependent appearance with Spherical Harmonics (SH) basis, we model our opacity feature f also as a set of SH coefficients, i.e., $f = \{k_h^i\}_{h=1}^H$, where $k_h^i \in \mathbb{R}^C$ correspond to the coefficients for C components, and H is the number of SH functions. Given a query ray direction $d = (\theta, \phi)$, the view dependent feature $\mathbf{S} \in \mathbb{R}^C$ can be written as:

$$\mathbf{S}(f, d) = \sum_{h=1}^{H} k_h^i Y_h(\theta, \phi) \tag{2}$$

where $Y_h : \mathbb{S}^2 \rightarrow \mathbb{R}$ is the SH bases. In our implementation, we set the dimension of SH bases to 91.

Rigging and Deformation. To make the octree adapt to animated animals, we directly "rig" the octree $\mathcal{V}_c$ in terms of its voxels from the canonical pose to the target skeleton $\mathcal{S}$, where S is the skeleton parameter $\mathcal{S} = \{\mathbf{r}, R, T\}$, $\mathbf{r} \in \mathbb{R}^{J\times 3}$ is the joint rotation angles, $R \in \mathbb{R}^3$ is global rotation and $T \in \mathbb{R}^3$ is global translation.

To rig the octree under skeletal motion, the brute force way is to rig all voxels in terms of their distance to the skeletons. Following Linear Blending Skinning (LBS), we instead apply skinning weights to voxels using the skinned mesh provided by the CGI models. Specifically, given mesh vertices and corresponding skinning weights, we generate per-voxel skinning weight by blending the weights of m closest vertices $\{v_j\}_{j=1}^m$ instead of only one point [Huang et al. 2020] on the mesh as:

$$w(\mathbf{p}_i) = \sum_{j=1}^{m} \alpha_j w_j, \;\; \alpha_j = e^{\delta_j} / \Sigma_k^m e^{\delta_k} \tag{3}$$

where w_j is the skinning weight of v_j and $\delta_j = d_j - min_k^m \, d_k$ with d_j as the Euclidean distance between v_j and voxel position $\mathbf{p}_i$.

With the voxel set $\mathcal{V}_c$, their skinning weights $\mathcal{W}$ and a corresponding skeleton $\mathcal{S}_c$ ready, we then conduct deformation from the canonical pose $\mathcal{S}_c$ to the target pose $\mathcal{S}_t$ with the transformation matrices $M^c, M^t \in \mathbb{R}^{J\times 4\times 4}$ following LBS as:

$$\mathbf{p}_i^t = \sum_{j=1}^{J} w_j(\mathbf{v}_i) M_j^t (M_j^c)^{-1} \mathbf{p}_i^c \tag{4}$$

with canonical voxel position $\mathbf{p}_i^c$ and J joints.

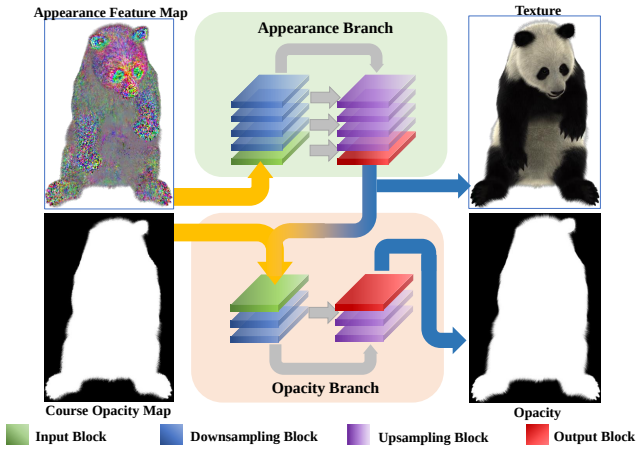

Fig. 5. **Neural Shading Network.** Our four blocks appearance branch translates the feature map to the foreground texture image, while the two blocks opacity branch takes the texture image to refine the coarse opacity map.

Dynamic Volume Rendering. Once we deform the octree to the target pose, we can render the neural animal via volume rendering and subsequently enable end-to-end training. We employ a differentiable volumetric integration scheme, as shown in Fig. 4. Given a ray $r_p^t = (\mathbf{o}_p^t, \mathbf{d}_p^t)$ with starting point $\mathbf{o}_p^t$ and ray direction $\mathbf{d}_p^t$ that corresponds to pixel p viewing the volume under pose $\mathcal{S}_t$, we can compute the view-dependent feature of p as:

$$F_p^t = \sum_{i}^{m} \alpha_i \mathbf{S}(f_i, \mathbf{d}_p^c) \tag{5}$$

$$\alpha_i = T_i\big(1 - \exp(-\sigma_i \delta_i)\big), \;\; T_i = \exp(-\sum_{j=1}^{i} \sigma_j \delta_j) \tag{6}$$

where δ_i is the distance of the ray traveling through a voxel $\mathbf{v}_i$, and f_i is the corresponding SHs coefficients fetched using the index stored in $\mathbf{v}_i$, m is the number of hit voxels. To further preserve consistency over view-dependent effects, we map the ray directions to the canonical space with a rotation matrix R_i^t by $\mathbf{d}_p^c = (R_i^t)^{-1}\mathbf{d}_p^t$.

Finally, we generate a view-dependent feature map $\mathcal{F}$ by applying Eqn.3.1 to each pixel and a coarse opacity map $\mathcal{A}$ by accumulating α_i along the rays. Since our neural animal model is dense and compact in space, uniform sampling used in NeuralVolumes [Lombardi et al. 2019] and NeRF [Mildenhall et al. 2020] can lead to rendering empty voxels. To tackle this problem, we employ a highly optimized data-parallel octree construction technique [Karras 2012] that only uses around 10ms to build an octree for millions of voxels. With this speed, we manage to rebuild the octree under pose variations and perform a ray-voxel intersection test for dynamic scenes. We further apply an early-stop strategy based on the accumulated alpha, i.e., we stop the integration once alpha is greater than $1 - \lambda_{th}$ (we use 0.01 in our implementation).

Neural Shading. We have by far rasterized the volume to a neural appearance feature map $\mathcal{F}$ and opacity $\mathcal{A}$ at the target pose

and viewpoint. The final step is to convert the rasterized volume to a color image with a corresponding opacity map resembling a classical shader. To preserve high frequency details of the fur, it is essential to consider spatial contents in the final rendered image. Notice, though, that neither NeRF nor PlenOctree considers spatial correlation as all pixels are rendered independently. We instead adopt an additional U-Net architecture Φ following ConvNeRF [Luo et al. 2021] to perform image rendering. Note that our ray-marching-based sampling strategy enables full-image rendering, in contrast to the patch-based strategy of ConvNeRF. Precisely, our neural shading network G consists of two encoder-decoder branches for RGB and alpha channels, respectively. The RGB branch converts $\mathcal{F}$ to texture images $\mathbf{I}_f$ with rich fur details. The alpha branch refines the coarse opacity map $\mathcal{A}$ and $\mathbf{I}_f$ to form a super-resolved opacity map $\mathbf{A}$. The process enforces multi-view consistency by explicitly utilizing the implicit geometry information encoded in $\mathcal{A}$.

Training. Recall that all features f are stored in FLUT. For acceleration, we randomly initialize FLUT and then jointly optimize the dense feature array $\mathbf{F}$ and the parameters of G from the multi-view animal motion videos. Our network aims to recover the appearances and opacity values of furry animals under free viewpoint. We therefore adopt the pixel-level L_1 loss as:

$$\mathcal{L}_{rgba} = \frac{1}{N_P}\sum_{i}^{N_P}(\|\hat{\mathcal{I}}_i - \mathcal{I}_i\|_1 + \|\hat{\alpha}_i - \alpha_i\|_1) \tag{7}$$

where $\mathcal{I}, \alpha$ are ground truth color and alpha rendered from ground truth fuzzy CGI animal models, and $\hat{\mathcal{I}}_i, \hat{\alpha}_i$ are synthesized color and alpha values from our network. In particular, $\hat{\mathcal{I}}_i$ is the blended result using the predicted alpha matte. N_P is the number of sampled pixels.

To recover fine texture and geometry details, we employ the *VGG19* perceptual loss [Johnson et al. 2016]:

$$\mathcal{L}_P = \frac{1}{N_I}\sum_{i}^{N_I}\sum_{l\in\{3,8\}}(\|\phi^l(\hat{\mathbf{I}}_i) - \phi^l(\mathbf{I}_i)\|_1 + \|\phi^l(\hat{\mathbf{A}}_i) - \phi^l(\mathbf{A}_i)\|_1) \tag{8}$$

where ϕ^l denotes the l^{th} layer feature map of *VGG19* backbone, and N_I is the number of sampled images.

To encourage cross view consistency as well as maintain temporal coherency, we impose geometry priors encoded in the ground truth alpha maps on the coarse alpha map $\mathcal{A}$:

$$\mathcal{L}_V = \frac{1}{N_I}\sum_{i}^{N_V}\|\mathcal{A}_i - \mathbf{A}_i\|_1 \tag{9}$$

Under motion, voxels may be warped to same locations that occlude previous frames. This leads to voxel collisions and feature overwrites of a grid and breaks the temporal consistency. To address this issue, we propose a voxel regularization term (VRT) that enforces features falling onto the same grid after warping should have identical values as:

$$\mathcal{L}_{vrt} = \lambda_{vrt}\frac{1}{N_v}\sum_{i}^{N_v}\|\mathbf{Q}(\mathbf{p}_i^t, \mathbf{T}^t) - \mathbf{f}_i^c\|_1 \tag{10}$$

$\mathbf{Q}$ is a query operator that outputs the corresponding feature of voxel at position $\mathbf{p}_i^t$ in octree $\mathbf{T}^t$. N_v is the voxel number.

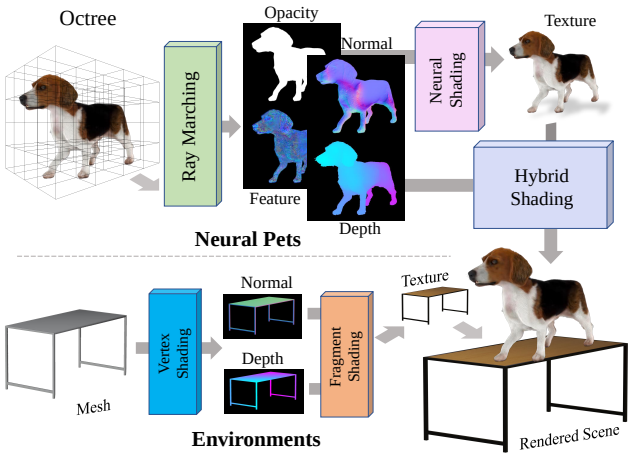

Fig. 6. **Neural Rendering Engine.** We develop a neural rendering engine based on our animatable neural volumetric representation. It can render our neural animals into standard frame buffers (e.g., texture, alpha, depth) used in the traditional 3D rendering pipeline.

To further improve the visual quality of high-frequency appearance of furs, we employ the PatchGAN [Isola et al. 2017] discriminator D to perform adversarial training with a GAN loss as:

$$\mathcal{L}_{GAN} = \sum_{i}^{N_I}(\|D(\hat{\mathbf{I}}_i)\|_2 + \|D(\mathbf{I}_i) - 1\|_2) \tag{11}$$

Hybrid Rendering. The processes of warping, octree construction, and volume rendering form a differentiable neural rendering pipeline and can be paralleled using C++ and CUDA C for efficient, end-to-end training. Once trained, the resulting octree structure supports real-time free-view rendering under animation. The solution can be further integrated with OpenGL and Unity engines, where a standard 3D rendering pipeline can conveniently render environments that host virtual animals. It is essential to address shading and occlusions in order to correctly fuse the environment with the neural rendering results to provide a realistic viewing experience. We, therefore, extend the volume rendering process described in Sec. 3.1 to render to different frame buffers. Firstly, the original outputs $\mathcal{F}$ and $\mathcal{A}$ correspond to neural textures for neural shading, and the final rendered images are stored in texture buffers. Neural rendered animals can also produce a depth buffer using a ray marching process: we cast a ray from each pixel from the image plane to our trained volumetric model, trace along the ray and locate the first voxel with a density larger than a threshold, and directly assign the distance as the depth. In contrast, we observe that directly applying such ray marching schemes on NeRF produces large errors near the boundary of the object, particularly problematic for furry objects. The rendered neural frame buffers can be combined with the ones produced by the standard rendering pipeline for resolving occlusions and alpha blending with the background. For example, the translucency of fur blends accurately with the floor or the background as the animal walks or jumps using our scheme.

Fig. 7. Setup of our animal motion capture system. The system consists of 12 Vicon cameras and 22 Z-CAM cinema cameras surrounding the animal.

4 NEURAL ANIMAL MOTION SYNTHESIS

For ARTEMIS to provide convincing motion controls, e.g., to guide an animal to walk or jump from one location to another, it is essential to synthesize realistic intermediate motions as the animal moves. For human motions, data driven approaches now serve as the gold standard. However, a sufficiently large animal mocap dataset is largely missing. We hence first set out to acquire a new animal mocap dataset along with a companion deep animal pose estimation that we intend to share with research community. Next, we tailor a neural character animator [Starke et al. 2020] for animals that maps the captured skeletal motions to abstract control commands and motion styles.

4.1 Animal Motion Capture

For data acquisition, we have constructed two types of animal motion capture systems, first composed of an array of RGB cameras for animals of large scales and the second combines RGB cameras and Vicon cameras for docile and small pets, as shown in Fig. 8.

MoCap Processes. We observe that quadrupeds, despite having similar skeletal structures across species, exhibit drastically different shapes and scales. It is simply impossible to capture a mocap dataset suitable for all types of quadrupeds. We hence set out to learn motion priors from docile and small pets and transfer the priors to larger animals such as tigers and wolves. For the latter, we further enhance the prediction accuracy using a multi-view RGB dome in partnership with the zoo and circus.

Fig. 7 shows our capture system for dogs and cats where we use 12 Vicon Vantage V16 cameras evenly distributed surrounding the animal subject, each capturing moving IR reflective markers at 120 Hz. We add additional 22 Z-CAM cinema cameras interleaved with Vicon and capturing at a 1920×1080 resolution at 30 fps. Cross-calibrations and synchronization are conducted in prior of actual motion capture. For small animals, we place the hybrid capture system closer to the ground so that they can clearly acquire limb movements while resolving occlusions between legs. We hire a professional trainer to guide these small animals to perform different movements including jumping and walking. Vicon produces highly accurate motion estimations that are used as priors for RGB camera based estimation.

For large animals, it is infeasible to place markers on large animals such as horses, tigers, elephants and etc. We partner with several zoos and circuses to deploy the RGB dome system where we use 22 $\tilde{6}0$ cameras and adjust their heights and viewing directions, depending on the size of the performance area. It is worth mentioning that several circuses further provided us surveillance footages of their animals where we manually select the usable ones and label their poses.

2D Key-points and Silhouettes Estimation. To automatically extract skeletons on multi-view RGB images for animal motion inference, we first estimate the 2D key-points and silhouettes in images [Biggs et al. 2020; Zuffi et al. 2019]. For key-points, we combine DeepLabCut [Mathis and Warren 2018] and the SMAL [Zuffi et al. 2017] model for processing imagery data towards quadrupeds, augmented with light manual annotations. Specifically, we annotate joints defined in the SMAL model on the first 10% frames and then allow DeepLabCut to track and estimate 2D poses for remaining frames. For silhouette extractions, we directly use the off-the-shelf DeepLab2 [Weber et al. 2021] with their pre-trained models.

Animal Pose Estimator. We adopt the parametric SMAL animal pose model. To briefly reiterate, SMAL is represented as a function $M(\beta, \theta, \gamma)$, where β is the shape parameter, θ the pose, and γ the translation. $\Pi(x, C_j)$ denote the projection of a 3D point x on the j-th camera whereas $\Pi(M, C_i)$ represents the projection of the SMAL model to camera C_i. Our goal is to recover SMAL parameters θ, ϕ, γ from the observed 2D joints and silhouette. Assume all RGB and Vicon cameras in our system are calibrated. We extend the key point reprojection error E_{kp}, silhouette error E_s as presented in [Zuffi et al. 2017] to multiview inputs. In our setup, we assume known animal species and therefore append shape constraints E_β from [Zuffi et al. 2017] to enforce the distribution of β to match the pre-trained species type. For our multi-view setup, we further add a 3D key-point constraint E_{3d} and a mocap constraint E_m as:

$$E_{3d}(\Theta; V) = \sum_k \left\| V_k - \mathbf{Tr}(v_k^i) \right\|_2 \tag{12}$$

where V_k denotes the kth SMAL joint in 3D space and v_k^i the observation of V_k in camera i. $\mathbf{Tr}(\cdot)$ corresponds to the triangulation operator. We hence have:

$$E_m(\Theta; M) = \sum_k \|M_k - \frac{1}{|\Omega|} \sum_{i \in \Omega_k} \mathcal{T}_k^i(\mathbf{Tr}(v_k^i)\|_2 \tag{13}$$

where M_k is position of the kth mocap marker. Ω_k corresponds to the set of joints used for recovering M_k. $\mathcal{T}_k^i$ is the transformation between the kth marker and vertex i. Notice not all vertices are usable for recovering M_k because the animal motion is non-rigid. We thus find Ω_k by identifying the vertices that have correspond to constant $\mathcal{T}_k^i$ across the motion sequence.

For human pose estimation, the SMPL-X human model [Pavlakos et al. 2019] introduces a prior using Variational AutoEncoder [Kingma and Welling 2014] to penalize on impossible poses. We adopt a similar idea: we per-train an animal motion prior using the same

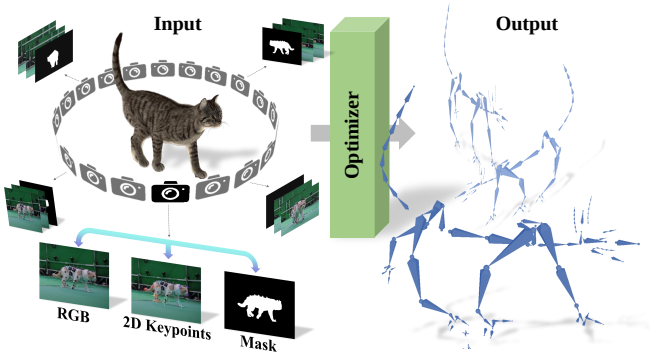

Fig. 8. We estimate the animal skeleton, SMAL parameters more specifically, from the multi-view video data and Vicon MoCap data. We adopt an optimization framework to regress SMAL parameter from multiview observations of 2D keypoints and silhouettes and Vicon Observations.

framework with all recovered skeletal poses of quadrupeds in our dataset and then use E_{prior} to penalize on the deviations from the prior. We thus can formulate the animal pose estimator in terms of an optimization problem as:

$$\beta, \theta, \gamma \leftarrow \arg\min(E_{kp} + E_s + E_{3d} + E_m + E_\beta + E_{prior}) \quad (14)$$

On RGB only captured images, we remove the marker loss term E_m. With recovered β, θ, γ for all frames, we can construct an animal motion sequence $\{\theta_j\}$ and $\{\gamma_j\}$.

4.2 Motion Synthesis

Our task is to generate virtual pets that exhibit realistic motions in response to a user's command. Physics-based controls are widely used in industry to generate vivid periodical motion patterns. However, those physical rules such as momentum conservation and reaction force can be confusing to users as they provide less direct guidance. Hence, we leverage the recent advances in data-driven methods and human movement animation, i.e., Local Motion Phases (LMP) [Starke et al. 2020], for our animal motion synthesis to learn movements from asynchronous behaviors of different body parts. Different from [Starke et al. 2019] that requires tedious manual annotation, LMP features can be automatically extracted and subsequently trained on unstructured motion data collected on animals.

Controlled Motion Synthesis with Local Motion Phases. To enable interactions between the user and the virtual animal, it is critical to synthesize animal motions according to user commands, e.g., for accomplishing specific tasks. We adopt the LMP framework that consists of a gating network and a motion prediction network, as shown in Fig. 9. The gating network takes the motion states (joint locations, rotations, and velocities) of the current frame and the past frames as inputs and computes expert activation, which is then dynamically blended with motion states and sent to the motion prediction network. Taking the expert weights and control signals from users, the motion prediction network would be able to calculate the motion state for future frames. The process can be expressed as:

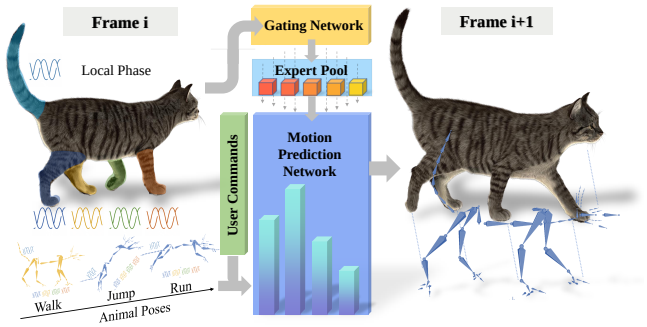

Fig. 9. **Neural Animal Controller Pipeline.** Leveraging ideas of Local Motion Phases, the neural animal controller is composed of a gating network and a motion prediction network. The gating network inputs local phases and calculates the expert blending coefficients. The coefficients are used to generate the motion prediction network. Then, the user-given control signals and motion information from previous frames are sent into the motion prediction network, which uses them to predict the motion information of the next frame.

$$\mathcal{M}_i = \Phi(\mathcal{M}_{i-1}, c) \quad (15)$$

where $\mathcal{M}_i$ is the motion state at frame i, c is the user control signal. In our implementation, we define the set of control signals as {'Idle', 'Move', 'Jump', 'Sit', 'Rest'}. Under these control signals, we extract the LMP feature accordingly, in terms of the contact state between the animal's feet or hips with the ground. The contact label is also extracted automatically by calculating the difference in positions and velocities between end-effectors and the environment collider. Furthermore, we process our motion data to calculate the action labels automatically.

For example, we compute Idle and Move states by calculating the root velocities and mapping them into values from 0 to 1. We detect Jump, Sit and Rest labels by jointly checking the position, velocity, and contact information and mapping them to values between 0 and 1. Specifically, Jump labels have no contact and y-axis velocity, whereas the Sit and Rest Labels have no velocity and all end-effectors contact with the environment but differ in the position of the head of the animal.

Motion transfer to articulated model. Notice that the generated motion state from LMP is bound to the skeleton captured in our Mocap systems. Yet the base models used for our neural pets are created by artists, and their skeletons do not match with the captured animals. We need to transfer the motions generated by LMP to animal models created by artists while keeping the model shape. Therefore, for each type of virtual animal, we manually apply offsets constraining the rotation and translation components in the rest pose. We then apply forward-kinematics and inverse-kinematics to calculate the target motion state and use transformation limits to fine-tune the impossible states. With the transformed motion data, we can transfer motion states from one to another and drive the neural animals to move freely.

Fig. 10. **Neural-generated animals in our ARTEMIS system.** We show our synthesized results of different neural volumetric animals in representative poses.

5 NEURAL ANIMALS IN IMMERSIVE ENVIRONMENTS

In previous sections, we have described how to train our animatable neural volumetric representation to synthesize animals under arbitrary poses and views, along with a hybrid rendering pipeline. Our neural motion controls and synthesis, enabled by our animal mocap data, provide convenient interfaces to guide the animal's movements. In this section, we assemble all these modules coherently under the VR setting as the ultimate ARTEMIS system, where a user can interact with the animal.

Interactions. Our motion synthesis module guides virtual animal movements by explicitly pointing to the target location and providing action types. We further explore high-level control patterns, emulating a commonly used set of commands by pet owners:

- Companion: As the user can move freely in virtual space, so will the virtual animal. It will follow the user.
- Go to: The user points to a 3D location in virtual space, and the animal reaches the target destination automatically. In a complicated environment where obstacles are presented, we use the A-Star algorithm to find the path for the animal to follow. The user can also control the speed of the movement, i.e., the animal can either walk to or run/jump to the destination.
- Go around a Circle: The user specifies a location, and the animal will reach the location and continue going around in circle. The user can even point themselves as the destination and ended being surrounded by single or multiple animals.
- Size and Speed Adjustment: Since our neural animal representation, as a continuous implicit function, can support infinite resolution, the user can adjust the animal's size by simply adjusting its octree accordingly. The user can also adjust the movement speed, either slowing it down or speeding it up. In particular, slow motion provides a fancy visual effect where the user can observe the animal at a close distance as if time freezes.
- Free Mode: when no command is given, the animal can take any reasonable movements, such as exploring the virtual world themselves. In our implementation, we randomly set destinations within a time interval where the animal reaches the destination one after another.

Implementations Details. The entire control flow for producing neural animals under the VR setting is as follows: We first obtain the pose of the VR headsets along with the controller status from OpenVR. Next, we then use LMP control signals and the current state to generate the estimated motion parameters. Based on the parameters, we rig the canonical model using LBS and then construct an octree for each frame. Finally, our neural rendering pipeline traces into the octree to generate the feature maps and subsequently conduct rendering where the background environments, obstacles in 3D environments, etc, are rendered using a standard graphics pipeline and fused with the neural rendering results.

Note that for VR applications, it is critical to generate two consistent views for presenting to the left and right eyes of a user, respectively, in the VR headsets. Inconsistent views between eyes may cause discomforts, such as nausea and dizziness. Thus when we deploy the trained neural animals to our system, we adopt the stereo loss proposed in LookinGood [Martin-Brualla et al. 2018], to

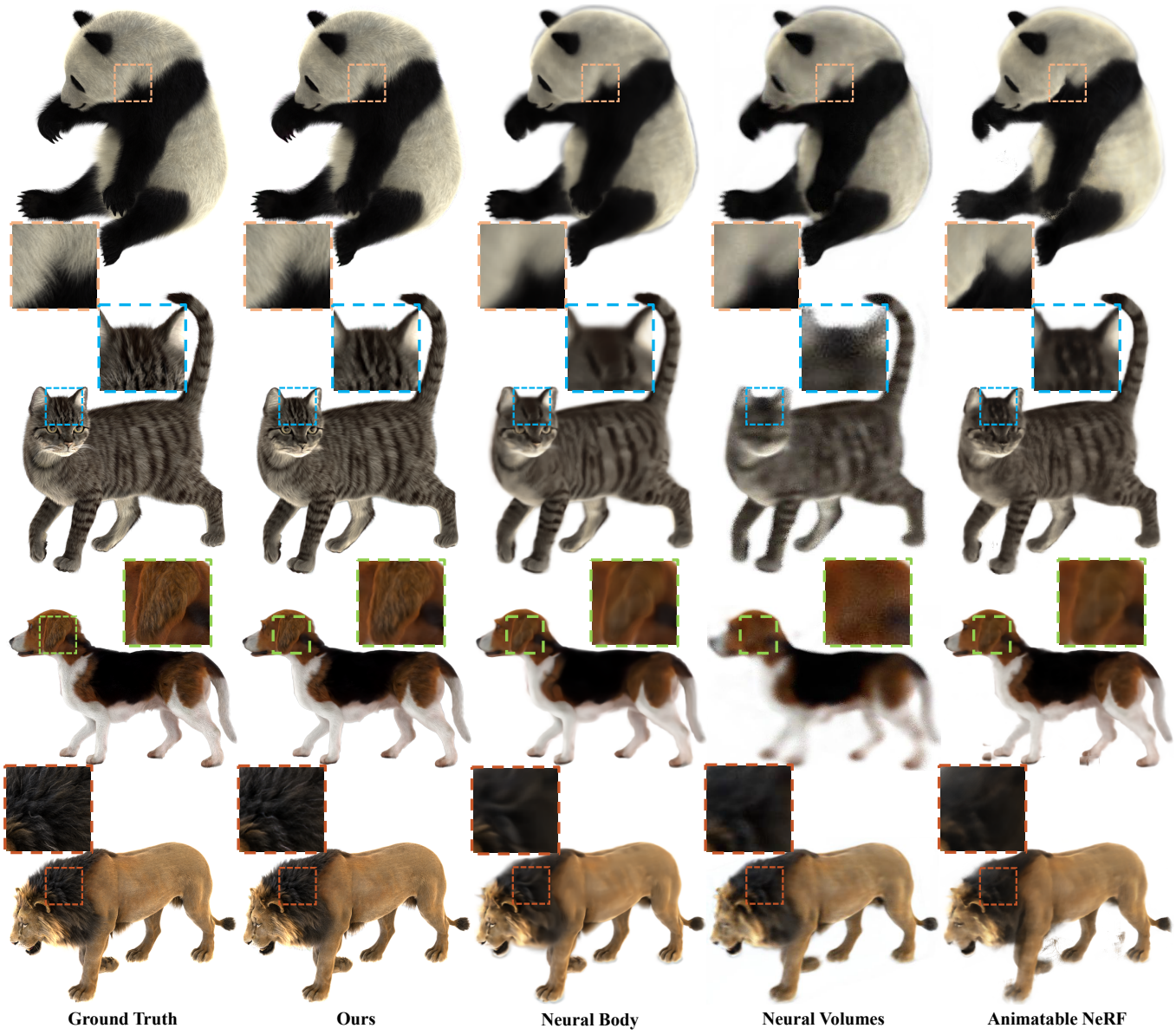

Fig. 11. **Qualitative comparison for dynamic appearance rendering** against NeuralVolumes, NeuralBody and AnimatableNeRF. Note that our approach achieves more photo-realistic rendering with finer details for appearance and fur.

finetune the trained model for better view consistency between left and right eyes in a self-supervised learning manner.

We follow the high cohesion and low coupling strategy to distribute time consumption as even as possible where the communication between core parts is lightweight via shared CUDA memory. The average computation time breakdown of individual parts are: 10ms for handle input (e.g., update the left and right camera poses, handle the controller action, controller environment), 29ms for motion synthesis (motion generation and motion transfer), 10ms for octree construction, 25ms for stereo neural rendering and environment synthesis, 10ms for output assembly and submit. By tailoring the assembly line of various components, we manage to produce the rendering at around 30 fps. The supplementary video shows live recordings of several examples.

6 RESULTS

In this section, we evaluate our ARTEMIS under various challenging scenarios. We first report our dynamic furry animal datasets, which are used for training our NGI animals, and our training and experimental details. Then, we provide the comparison of our neural rendering scheme with current state-of-the-art (SOTA) methods for both dynamic appearance rendering and static opacity generation,

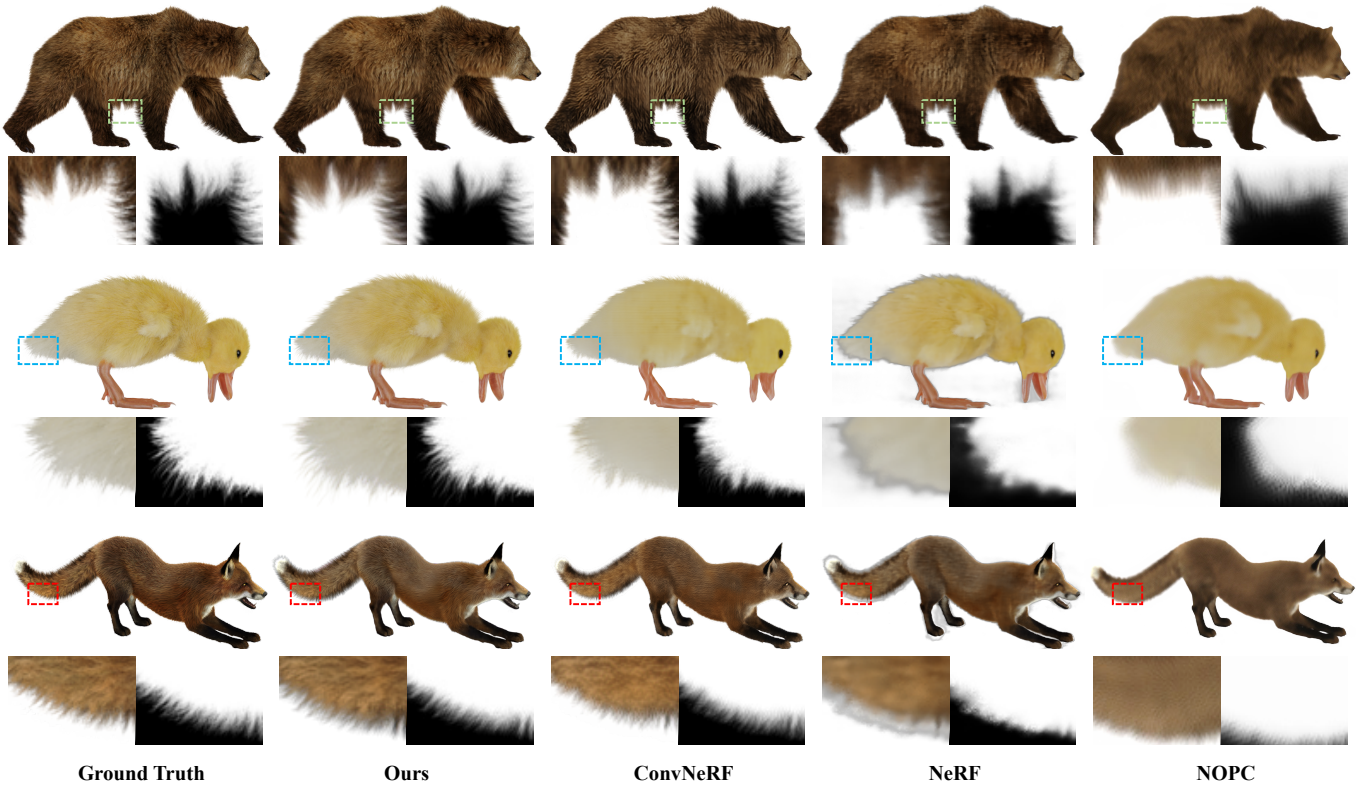

Fig. 12. **Qualitative comparison for free-view appearance and opacity rendering of static scene** with NOPC, NeRF and ConvNeRF. Our approach achieves modeling high-frequency geometry details and generates almost the same RGBA images with the ground truth.

which demonstrate that our method better preserves high-frequency details. We also conduct a detailed runtime performance analysis of our rendering approach and provide extra evaluation for our motion capture scheme. We further provide the VR applications of our ARTEMIS using consumer-level VR headsets where a user can intimately interact with various virtual animals. Finally, the limitation and discussions regarding our approach are provided in the last subsection.

Dynamic Furry Animal Dataset. To evaluate our ARTEMIS (neural pets) system, especially for our NGI animals, we seek help from traditional CGI animal animation and rendering pipelines to generate a comprehensive training dataset. Specifically, our Dynamic Furry Animal (DFA) dataset contains nine high-quality CGI animals with fiber/strands based furs and skeletal rigs modeled through tedious artists' efforts, including panda, lion, cat, etc. We utilize the commercial rendering engine (e.g., MAYA) to render all these CGI animal characters into high-quality multi-view 1080×1080 RGBA videos under various representative skeletal motions. Specifically, we adopted 36 camera views which are uniformly arranged around a circle towards the captured animal, and the number of representative poses ranges from 700 to 1000 for each animal. Note that on average it takes about 10 minutes to render a single frame of our high-quality DFA dataset using commercial render farms. For training an NGI animal from the corresponding CGI asset, we uniformly select 30 views to conduct per-animal training and leave six views as test data for evaluation. We train our NGI animals at 960×540 image resolution using 4 Nvidia TITAN RTX GPUs, and it takes about two days for the training process. Fig. 10 demonstrates the representative rendering results of our NGI animals under various novel skeletal poses, where the details of appearance, fur, and opacity are faithfully reconstructed with real-time performance. Besides, our high-quality DFA dataset will be made publicly available to the community to stimulate future research about realistic animal modeling.

6.1 Comparison to Dynamic Neural Rendering

Here we compare our neural rendering pipeline in ARTEMIS with recent SOTA methods for dynamic scene rendering. For thorough comparison, we compare against both the radiance field based methods, including NeuralBody [Peng et al. 2021b] and AnimatableNeRF [Peng et al. 2021a], as well as the volume-based approach NeuralVolumes [Lombardi et al. 2019]. Note that our NGI animal requires an additional pre-defined skeletal rig with skinning weights from a corresponding CGI digital asset. Thus, for fair comparisons, we train the baseline models using the same experiment settings as our approach and adopt the ground truth mesh vertices and skinning weights of our CGI animal models to train NeuralBody and AnimatableNeRF.

Table 1. Quantitative comparisons of synthesized appearance images on different dynamic animals. Compared with NeuralVolumes, NeuralBody, and AnimableNeRF, our method achieves the best performance in all metrics.

Method		Neural Body	Neural Volumes	Animatable NeRF	Ours
Panda	↑ PSNR	30.38	30.11	26.51	**33.63**
	↑ SSIM	0.970	0.965	0.957	**0.985**
	↓ LPIPS	0.110	0.116	0.112	**0.031**
Cat	↑ PSNR	30.77	28.14	31.37	**37.54**
	↑ SSIM	0.972	0.951	0.973	**0.989**
	↓ LPIPS	0.067	0.087	0.061	**0.012**
Dog	↑ PSNR	32.37	26.80	31.19	**38.95**
	↑ SSIM	0.978	0.945	0.975	**0.989**
	↓ LPIPS	0.075	0.129	0.074	**0.022**
Lion	↑ PSNR	30.11	29.59	27.87	**33.09**
	↑ SSIM	0.956	0.947	0.944	**0.966**
	↓ LPIPS	0.111	0.123	0.123	**0.035**

Fig. 11 provides the qualitative comparison results of our approach against the above baselines. Note that only fur with color much different from the background can be recognized as fur, like lion manes and cat forehead. Non-surprisingly, the three baseline methods suffer from artifacts of blurry fur and loss of appearance details. Specifically, NeuralVolumes suffers from severe noises and blurring effects due to its limited modeling ability, which is most obviously on the cat forehead. NeuralBody can only generate low-frequency fur details as blurry effects, while AnimatableNeRF behaves slightly better but still cannot faithfully recover the furry details. In stark contrast, our approach generates much more clear details favorably, especially for those furry regions which are common in animals. For quantitative evaluation, we adopt the peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) as metrics to evaluate our rendering accuracy. As shown in Tab. 1, our approach significantly outperforms the other baselines for all the above metrics, demonstrating the superiority of our approach in preserving rendering details.

6.2 Comparison to Static RGBA Rendering

Here we evaluate our approach for generating free-view opacity maps, which is important for fur modeling. To the best of our knowledge, there are no previous works that can generate opacity maps for dynamic scenes. Thus, we compare our approach against the recent SOTA methods on the opacity rendering task for static scenes. For thorough comparison, we select three baselines, including the explicit point-cloud based method called Neural Opacity Point Clouds (NOPC) [Wang et al. 2020], as well as the radiance field based methods NeRF [Mildenhall et al. 2020] and ConvNeRF [Luo et al. 2021].

Fig. 12 shows several RGB and alpha results of our method and other static methods. *NOPC* suffers from aliasing and ghosting on the boundary regions due to the interpolation of point-cloud based features. *NeRF* suffers from severe blur artifacts because of the limited representation ability of its MLP network, especially for high-frequency details, while *ConvNeRF* improves the alpha and RGB details but still causes grid-like artifacts due to patch-based training strategy on limited sparse training views. In contrast, our approach achieves the best performance where we manage to compensate for missing views at a specific frame using views from other frames.

Table 2. Quantitative comparisons of appearance and alpha on a single representative frame of different animals. Our approach achieves the best performance in almost all alpha-related metrics and comparable performance for RGB texture against ConvNeRF.

RGB		NOPC	NeRF	ConvNeRF	Ours
Bear	↑ PSNR	18.43	28.12	**32.34**	30.95
	↑ SSIM	0.886	0.954	0.953	**0.967**
	↓ LPIPS	0.140	0.113	0.063	**0.038**
Duck	↑ PSNR	25.45	30.35	34.31	**37.14**
	↑ SSIM	0.967	0.978	0.985	**0.986**
	↓ LPIPS	0.075	0.091	0.052	**0.026**
Fox	↑ PSNR	17.42	27.53	**33.42**	30.94
	↑ SSIM	0.914	0.966	0.973	**0.976**
	↓ LPIPS	0.106	0.099	0.047	**0.029**
Alpha		NOPC	NeRF	ConvNeRF	Ours
Bear	↑ PSNR	17.89	31.65	36.37	**40.13**
	↑ SSIM	0.918	0.986	0.992	**0.995**
	↓ SAD	144.2	199.2	11.80	**8.072**
Duck	↑ PSNR	19.77	30.09	33.02	**36.81**
	↑ SSIM	0.849	0.923	0.990	**0.994**
	↓ SAD	110.6	36.17	12.76	**8.558**
Fox	↑ PSNR	15.68	23.81	34.90	**36.43**
	↑ SSIM	0.903	0.968	0.993	**0.995**
	↓ SAD	192.4	52.32	11.30	**9.555**

Differently, our approach generates more realistic opacity rendering and even supports dynamic scenes in real-time. The corresponding quantitative result is provide in Tab. 2. For the evaluation of the alpha map, we further adopt Sum of Absolute Distance (SAD) as metrics besides PSNR and SSIM. Our approach outperforms all the baselines for alpha-related metrics and maintains comparable performance for RGB texture rendering against ConvNeRF. All these evaluations illustrate the effectiveness of our approach for high-quality fur and opacity detail rendering.

6.3 Ablation Study

We conduct extensive ablation studies on the "Bear" data. Let **w/o VRT** and **w/o GAN** denote our models trained without voxel regularization term (VRT) and GAN loss. Fig. 13 shows GAN loss helps to preserve more high-frequency details and synthesized sharper alpha map, and VRT leads to slightly smoothed texture while maintaining temporal consistency. We further train our model to directly render RGBA images through volumetric without neural shading

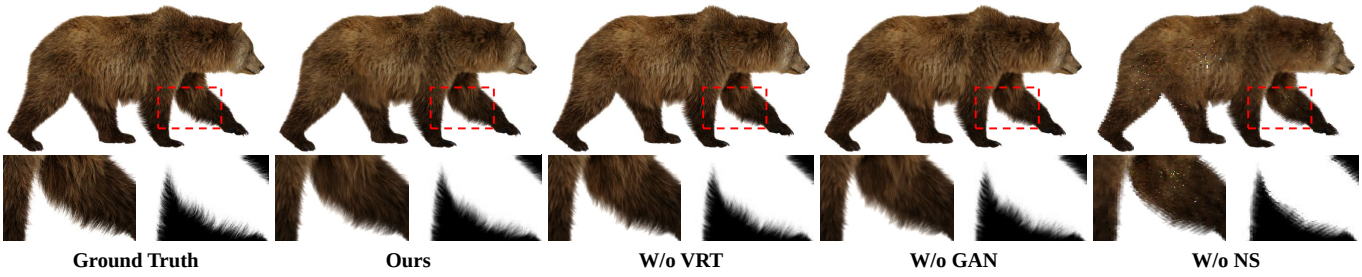

Fig. 13. **Visualization of the ablation study** on w/o the NS, GAN loss and VRT loss modules (corresponding to the Tab. 3).

Table 3. Quantitative evaluation of the ablation studies on w/o the NS, GAN loss and VRT loss modules. Our full model achieves the best performance.

Models	*RGB*			Alpha		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	SAD ↓
(a) w/o NS	29.47	0.939	0.152	29.27	0.981	24.68
(b) w/o GAN	34.29	0.964	0.051	35.84	0.991	12.47
(c) w/o VRT	33.81	0.961	**0.044**	35.74	**0.992**	12.23
(d) ours	**34.43**	**0.965**	0.045	**36.18**	**0.992**	**11.92**

network (NS) (**w/o NS**). It suffers from noises and artifacts around the legs and abdomen due to geometry misalignment and limited volume resolution. The qualitative result in Tab. 3 well supports the contribution of each module.

6.4 Runtime Performance Evaluation

We run our method on an Nvidia TITAN RTX GPU. Tab. 4 compares the running time of our method for one frame with others. For dynamic scenes, we compare our full runtime, i.e., warping to target pose and rendering, with other baselines. Our method achieves real-time novel pose synthesis and rendering. For static scene, we compare our free-view rendering runtime. Our method is much faster (in order of 10 times) than other methods, especially achieves three to four orders of magnitude rendering speed up to traditional CGI animal character, which further supports our novel ARTEMIS system.

Runtime Analysis. We further analyze the runtime of each part and the performance-speed trade-off on a single Nvidia GeForce RTX3090 GPU. As shown in Tab. 5, for a general case, where the resolution of our sparse volume is 512, animating (warp + build octree) costs around 10ms, volume rendering costs around 5ms, and the neural shading network costs around 13ms with half floating-point precision. It costs 27.43ms in total. To further accelerate, we design a light shading network by removing the opacity branch and adding an output channel to the appearance branch to predict the opacity map, denoted as **Light**. The rendering time reduces to 24.84ms with a slight performance downgrade. Finally, by combining the light network with 256 volume resolution, denoted as **256 + Light**, it takes 22.69ms with a slight performance drop. We adopt this scheme to provide a better experience in our ARTEMIS system.

Table 4. Runtime comparisons different Methods. Our method is significantly faster than existing methods and enables real-time applications.

Dynamic	CGI	Neural Body	Neural Volumes	Animatable NeRF	Ours
runtime (ms)	$\sim 5 \times 10^5$	2353	181.7	18142	**34.29**
fps	–	0.425	5.504	0.055	**29.16**
Static	CGI	NOPC	NeRF	ConvNeRF	Ours
runtime (ms)	$\sim 5 \times 10^5$	51.23	18329	2599	**20.38**
fps	–	19.52	0.055	0.385	**49.07**

Table 5. Runtime (in milliseconds) analysis of components of our model and performance-speed trade-off.

Model	Normal	Light	256 + Light
PSNR ↑	**33.01**	32.80	32.47
SSIM ↑	**0.991**	0.990	0.988
LPIPS ↓	**0.010**	0.011	0.012
warp ↓	1.950	1.975	**1.915**
build otree ↓	7.990	8.321	**6.285**
volume render ↓	4.521	4.520	**2.478**
neural render ↓	12.97	10.03	**10.01**
total ↓	27.43	24.84	**22.69**

6.5 Interactive NGI Animals in VR.

As shown in Fig. 14, we exhibit our rendering results in VR applications, including different level interactions and different perspectives. The first line, 'viewing' shows the vivid virtual panda we observed from the third and first view(VR headset), respectively. We can clearly see good fur effects even in VR headsets. The second line, 'Instructions' shows the low-level control animation results. We can explicitly drive our neural pets using control signals like 'Jump', 'Move' and 'Sit'. 'Commands' illustrates our high-level control pattern 'Go to'. The user points to a 3D location in virtual space, and the wolf reaches the target destination automatically. We can also call back the wolf by waving. Note that the speed of the movement is also controlled by the user. In 'Companion', our virtual pet will follow and accompany the user like a real pet. Finally, 'Exploring' shows our free mode, when no command is given, the animal can make any reasonable movements, exploring the virtual world themselves.

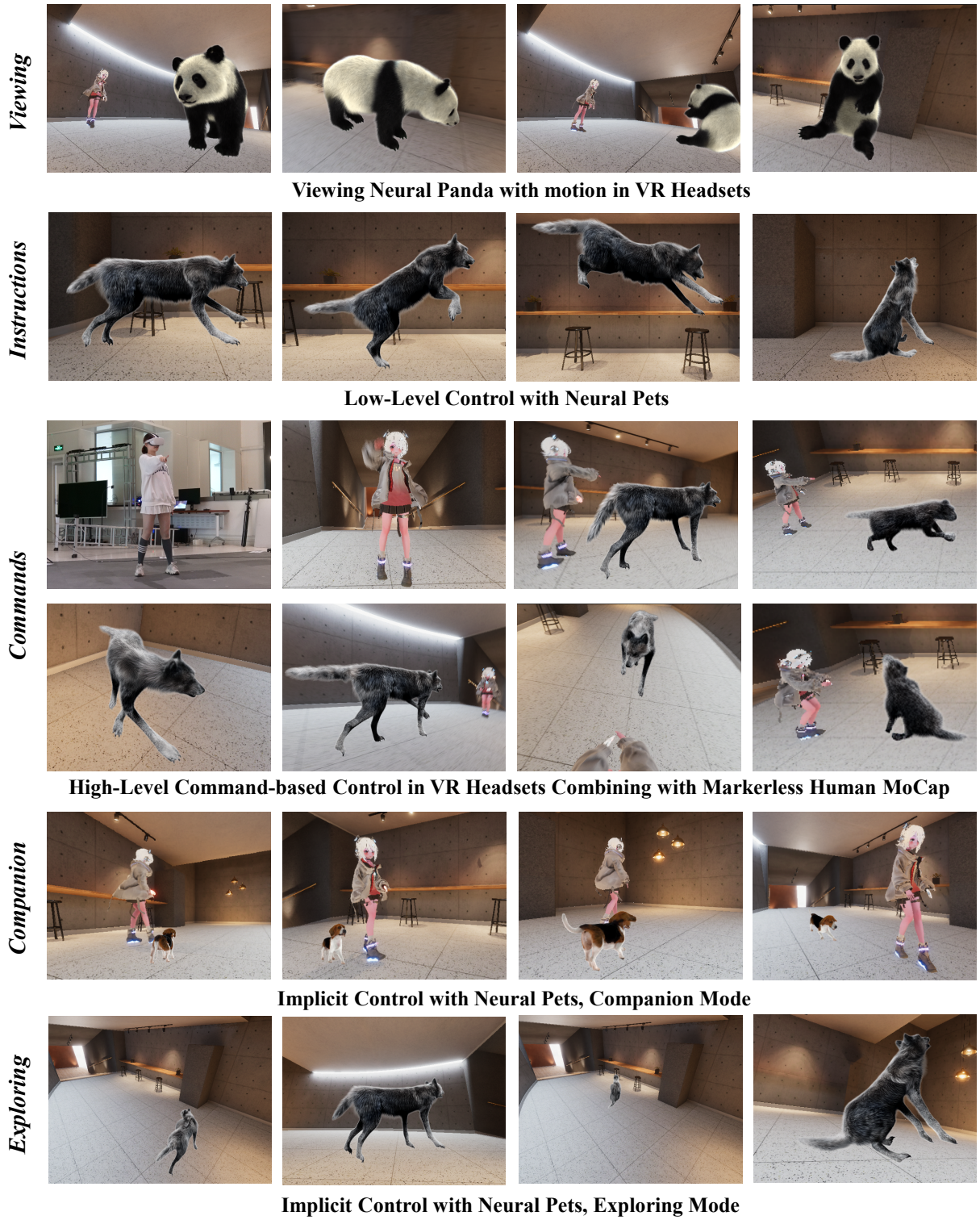

Fig. 14. **Example interactive modes of our ARTEMIS.** From top to bottom are different levels of interactions: viewing - we can observe the animal and see the fur clearly; instructions - drive the animals with explicit signals, such as 'jump' or 'sit'; commands - high-level control patterns, such as user points at a 3D location and the animal moves to there; companion - the animal follows the user; 'exploration' - animals show random movements.

Fig. 15. **A concept demo of animal and user interaction in a VR environment.** We develop a VR demo to exhibit how users can interact with a virtual pet in a VR environment. We obtain the coordinate of the user in VR according to the controller, the wolf pet then plays with the user, e.g., jumping around the user.

We show more interactive results in Fig. 15.

6.6 Limitations and Discussions

We have demonstrated the compelling capability of interactive motion control, real-time animation, and photo-realistic rendering of neural-generated (NGI) furry animals even with modern commercial-level VR headsets. However, as a first trial to bring photo-realistic NGI animals into our daily immersive applications with us humans, our ARTEMIS system still owns limitations as follows.

First, our NGI animal still heavily relies on a pre-defined skeletal rig and skinning weights of the corresponding CGI animal character. Despite the considerable improvement in terms of rendering time speed up from CGI to NGI animals, the problem of fully automatic neural animal generation from captured data without the artist's effort remains unsolved. In future work, we plan to combine the non-rigid tracking technique [Xu et al. 2020] to alleviate our reliance on skeletal parameters. Moreover, within the paradigm of neural rendering, our approach suffers from appearance artifacts for those unobserved animal body regions during training and under those challenging rendering views that are too different compared to the captured camera views. It is promising to combine the appearance prior of certain kinds of animals to alleviate such neural rendering shortcomings. Our current rendering results still suffer from flickering artifacts when the rendering view or the captured animal is moving, even using our voxel regularization. We suspect that it's partially caused by the voxel collision of our dynamic Octree reconstruction. Besides, our technique cannot yet handle self-contact between distant body parts. In future work, we plan to explore a more elegant Octree indexing scheme and employ more temporal regularization to handle the flickering. Moreover, our current NGI animal can only generate results within the baked lighting environment while rendering the CGI asset and thus cannot quickly adopt the lighting in a new environment compared to traditional rendering engines. It is interesting to enable more explicit lighting control on top of the current NGI animal design.

From ARTEMIS system side, our current interactions with NGI animals are still limited by the heavy reliance on human motion capture and pre-defined rule-based strategies. Fundamentally, current ARTEMIS design is based on only several relatively basic interactive modes. Despite the unprecedented interaction experience of ARTEMIS, a more advanced and knowledge-based strategy for the interaction between NGI animals and us humans are in urgent need. Moreover, more engineering effort is needed to provide a more immersive experience for current VR applications, especially for supporting the rendering of multiple animals.

Discussion. ARTEMIS has the potential to create the neural animal directly from captured real animal data if some obstacles are conquered: it requires high-quality multi-view videos of furry animals with rich fur details while obtaining such videos of real animals remains difficult. Animals occupy a much smaller portion within the images than human performers, and therefore their captured videos are generally of a low resolution. Further, balancing the exposure setting (aperture vs. shutter vs. gain) to capture fast motion and shape images simultaneously can be very tricky. In contrast, skeleton extraction is generally robust if the camera setting is dense, as shown in the paper. Recent excellent work BANmo [Yang et al. 2021] also provides promising insights for animals from casual videos.

On the other hand, compared to rasterization-based schemes for fast rendering hair and fur (e.g., EEVEE), our work, instead, aims to show that neural rendering provides a promising alternative that can tackle shape deformations similar to traditional meshes but implicitly. In addition, such neural representations can potentially

handle real animations without manual model building: by building a neural representation from a multi-view video sequence of a real animal, one can potentially conduct photo-realistic rendering of the animal under unseen poses without explicitly modeling the underlying geometry, let alone fur. Besides, furry objects are challenging to model using meshes: it requires extensive labor by artists and the use of ultra-dense meshes where implicit representation such as NeRF and its extensions can relieve artists from such labor.

7 CONCLUSION

We have presented a neural modeling and rendering pipeline called ARTEMIS for generating articulated neural pets with appearance and motion synthesis. Our ARTEMIS system has interactive motion control, realistic animation, and high-quality rendering of furry animals, all in real-time. At the core of our ARTEMIS, our neural-generated (NGI) animals renew the rendering process of traditional CGI animal characters, which can generate real-time photo-realistic rendering results with rich details of appearance, fur, and opacity with notably three to four orders of magnitude speed-up. Our novel dynamic neural representation with a tailored neural rendering scheme enables highly efficient and effective dynamic modeling for our NGI animal. Our robust motion capture scheme for real animals, together with the recent neural character control scheme, provides the controllable ability to interact with our NGI animals with more realistic movements. Moreover, our hybrid rendering engine enables the integration of ARTEMIS into existing consumer-level VR headset platforms so as to provide a surreal and immersive experience for users to intimately interact with virtual animals as if in the real world. Extensive experimental results and VR showcases demonstrate the effectiveness of our approach for neural animal modeling and rendering, supporting immersive interactions unseen before. We believe that our approach renews the way humans perceive and interact with virtual animals, with more immersive, realistic, and responsive interaction experiences, with many potential applications for animal digitization and protection or fancy human-animal interactions in VR/AR, gaming, or entertainment.

ACKNOWLEDGMENTS

The authors would like to thank Junyu Zhou and Ya Gao from DGene Digital Technology Co., Ltd. for processing the CGI animals models and motion capture data. Besides, we thank Zhenxiao Yu and Heyang Li from ShanghaiTech University for producing a supplementary video and figures.

This work was supported by NSFC programs (61976138, 61977047), the National Key Research and Development Program (2018YFB2100500), STCSM (2015F0203-000-06) and SHMEC (2019-01-07-00-01-E00003).

REFERENCES

Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. 2020. Neural point-based graphics. In *European Conference on Computer Vision*. Springer, 696–712.

Pang Anqi, Chen Xin, Luo Haimin, Wu Minye, Yu Jingyi, and Xu Lan. 2021. Few-shot Neural Human Performance Rendering from Sparse RGBD Videos. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI-21*.

Yongtang Bao and Yue Qi. 2016. Realistic Hair Modeling from a Hybrid Orientation Field. *Vis. Comput.* 32, 6–8 (jun 2016), 729–738. https://doi.org/10.1007/s00371-016-1240-1

Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824* (2020).

Benjamin Biggs, Oliver Boyne, James Charles, Andrew Fitzgibbon, and Roberto Cipolla. 2020. Who left the dogs out? 3d animal reconstruction with expectation maximization in the loop. In *European Conference on Computer Vision*. Springer, 195–211.

Benjamin Biggs, Thomas Roddick, Andrew Fitzgibbon, and Roberto Cipolla. 2018. Creatures great and smal: Recovering the shape and motion of animals from video. In *Asian Conference on Computer Vision*. Springer, 3–19.

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12684–12694.

Thomas J Cashman and Andrew W Fitzgibbon. 2012. What shape are dolphins? building 3d morphable models from 2d images. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 232–244.

Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. 2020. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*. Springer, 608–625.

Menglei Chai, Linjie Luo, Kalyan Sunkavalli, Nathan Carr, Sunil Hadap, and Kun Zhou. 2015. High-Quality Hair Modeling from a Single Portrait Photo. *ACM Trans. Graph.* 34, 6, Article 204 (oct 2015), 10 pages. https://doi.org/10.1145/2816795.2818112

Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, and Kun Zhou. 2016. AutoHair: Fully Automatic Hair Modeling from a Single Image. *ACM Trans. Graph.* 35, 4, Article 116 (jul 2016), 12 pages. https://doi.org/10.1145/2897824.2925961

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. arXiv:2103.15595 [cs.CV]

Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. 2019. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*. 9609–9619.

Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2015. A Practical and Controllable Hair and Fur Model for Production Path Tracing. In *ACM SIGGRAPH 2015 Talks* (Los Angeles, California) *(SIGGRAPH '15)*. Association for Computing Machinery, New York, NY, USA, Article 23, 1 pages. https://doi.org/10.1145/2775280.2792559

Eugene d'Eon, Guillaume Francois, Martin Hill, Joe Letteri, and Jean-Marie Aubry. 2011. An Energy-Conserving Hair Reflectance Model. In *Proceedings of the Twenty-Second Eurographics Conference on Rendering* (Prague, Czech Republic) *(EGSR '11)*. Eurographics Association, Goslar, DEU, 1181–1187. https://doi.org/10.1111/j.1467-8659.2011.01976.x

Alexandre Derouet-Jourdan, Florence Bertails-Descoubes, and Joëlle Thollot. 2013. Floating Tangents for Approximating Spatial Curves with G1 Piecewise Helices. *Comput. Aided Geom. Des.* 30, 5 (jun 2013), 490–520. https://doi.org/10.1016/j.cagd.2013.02.007

Zhipeng Ding, Yongtang Bao, and Yue Qi. 2016. Single-View Hair Modeling Based on Orientation and Helix Fitting. In *2016 International Conference on Virtual Reality and Visualization (ICVRV)*. 286–291. https://doi.org/10.1109/ICVRV.2016.54

Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14346–14355.

Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. 2021. Real-Time Deep Dynamic Characters. *ACM Trans. Graph.* 40, 4, Article 94 (jul 2021), 16 pages. https://doi.org/10.1145/3450626.3459749

Tong He, Yuanlu Xu, Shunsuke Saito, Stefano Soatto, and Tony Tung. 2021. ARCH++: Animation-ready clothed human reconstruction revisited. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11046–11056.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. 2021. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5875–5884.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2015. Single-View Hair Modeling Using a Hairstyle Database. *ACM Trans. Graph.* 34, 4, Article 125 (jul 2015), 9 pages. https://doi.org/10.1145/2766931

Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. 2020. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3093–3102.

Eldar Insafutdinov and Alexey Dosovitskiy. 2018. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. 2020. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6001–6010.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*. Springer, 694–711.

J. T. Kajiya and T. L. Kay. 1989. Rendering Fur with Three Dimensional Textures. *SIGGRAPH Comput. Graph.* 23, 3 (jul 1989), 271–280. https://doi.org/10.1145/74334.74361

Angjoo Kanazawa, Shahar Kovalsky, Ronen Basri, and David Jacobs. 2016. Learning 3d deformation of animals from 2d images. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 365–374.

Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. 2018. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 371–386.

Tero Karras. 2012. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics*. 33–37.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3907–3916.

Sinead Kearney, Wenbin Li, Martin Parsons, Kwang In Kim, and Darren Cosker. 2020. RGBD-Dog: Predicting Canine Pose from RGBD Sensors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. arXiv:http://arxiv.org/abs/1312.6114v10 [stat.ML]

Maria Kolos, Artem Sevastopolsky, and Victor Lempitsky. 2020. TRANSPR: Transparency Ray-Accumulating Neural 3D Scene Point Renderer. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 1167–1175.

Zhengfei Kuang, Kyle Olszewski, Menglei Chai, Zeng Huang, Panos Achlioptas, and Sergey Tulyakov. [n.d.]. NeROIC: Neural Object Capture and Rendering from Online Image Collections. ([n. d.]).

Youngjoong Kwon, Dahun Kim, Duygu Ceylan, and Henry Fuchs. 2021. Neural human performer: Learning generalizable radiance fields for human performance rendering. *Advances in Neural Information Processing Systems* 34 (2021).

Christoph Lassner and Michael Zollhofer. 2021. Pulsar: Efficient Sphere-based Neural Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1440–1449.

Ruilong Li, Kyle Olszewski, Yuliang Xiu, Shunsuke Saito, Zeng Huang, and Hao Li. 2020a. Volumetric Human Teleportation. In *ACM SIGGRAPH 2020 Real-Time Live!* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 1 pages. https://doi.org/10.1145/3407662.3407756

Ruilong Li, Yuliang Xiu, Shunsuke Saito, Zeng Huang, Kyle Olszewski, and Hao Li. 2020b. Monocular real-time volumetric performance capture. In *European Conference on Computer Vision*. Springer, 49–67.

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.

Chen-Hsuan Lin, Chen Kong, and Simon Lucey. 2018. Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020a. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020), 15651–15663.

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 Trans. Graph.(ACM SIGGRAPH Asia)* (2021).

Lingjie Liu, Weipeng Xu, Marc Habermann, Michael Zollhöfer, Florian Bernard, Hyeongwoo Kim, Wenping Wang, and Christian Theobalt. 2020b. Neural Human Video Rendering by Learning Dynamic Textures and Rendering-to-Video Translation. *IEEE Transactions on Visualization and Computer Graphics* PP (05 2020), 1–1. https://doi.org/10.1109/TVCG.2020.2996594

Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. 2019. Neural rendering and reenactment of human actor videos. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–14.

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 (jul 2019), 14 pages. https://doi.org/10.1145/3306346.3323020

Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. 2021. Mixture of Volumetric Primitives for Efficient Neural Rendering. *ACM Trans. Graph.* 40, 4, Article 59 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459863

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

H. Luo, A. Chen, Q. Zhang, B. Pang, M. Wu, L. Xu, and J. Yu. 2021. Convolutional Neural Opacity Radiance Fields. In *2021 IEEE International Conference on Computational Photography (ICCP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–12. https://doi.org/10.1109/ICCP51581.2021.9466273

Linjie Luo, Hao Li, and Szymon Rusinkiewicz. 2013. Structure-Aware Hair Capture. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32, 4 (July 2013).

Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. 2003. Light Scattering from Human Hair Fibers. *ACM Trans. Graph.* 22, 3 (jul 2003), 780–791. https://doi.org/10.1145/882262.882345

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. 2018. LookinGood: Enhancing Performance Capture with Real-Time Neural Re-Rendering. *ACM Trans. Graph.* 37, 6, Article 255 (dec 2018), 14 pages. https://doi.org/10.1145/3272127.3275099

Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7210–7219.

Alexander Mathis and Richard A. Warren. 2018. On the inference speed and video-compression robustness of DeepLabCut. *bioRxiv* (2018). https://doi.org/10.1101/457242 arXiv:https://www.biorxiv.org/content/early/2018/10/30/457242.full.pdf

Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. 2002. Image-Based 3D Photography Using Opacity Hulls. *ACM Trans. Graph.* 21, 3 (jul 2002), 427–437. https://doi.org/10.1145/566654.566599

Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. 2021. Gnerf: Gan-based neural radiance field without posed camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6351–6361.

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 *European conference on computer vision*. Springer, 405–421.

Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum* 40, 4 (2021). https://doi.org/10.1111/cgf.14340

Sylvain Paris, Hector M. Briceño, and François X. Sillion. 2004. Capture of Hair Geometry from Multiple Images. *ACM Trans. Graph.* 23, 3 (aug 2004), 712–719. https://doi.org/10.1145/1015706.1015784

Sylvain Paris, Will Chang, Oleg I. Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. 2008. Hair Photobooth: Geometric and Photometric Acquisition of Real Hairstyles. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–9. https://doi.org/10.1145/1360612.1360629

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, Article 238 (dec 2021).

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. 2021a. Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14314–14323.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 523–540.

Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. 2021b. 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.

Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14335–14345.

Riccardo Roveri, Lukas Rahmann, Cengiz Oztireli, and Markus Gross. 2018. A network architecture for point cloud classification via automatic depth images generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4176–4184.

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. 2019. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2304–2314.

Andrew Selle, Michael Lentine, and Ronald Fedkiw. 2008. A Mass Spring Model for Hair Simulation. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–11. https://doi.org/10.1145/1360612.1360663

Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, et al. 2019. Textured neural avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2387–2397.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems* 33 (2020).

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. 2019. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2437–2446.

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.

Sebastian Starke, Yiwei Zhao, Fabio Zinno, and Taku Komura. 2021. Neural animation layering for synthesizing martial arts movements. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–16.

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).

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (2020).

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2021. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12959–12970.

Cen Wang, Minye Wu, Ziyu Wang, Liao Wang, Hao Sheng, and Jingyi Yu. 2020. Neural Opacity Point Cloud. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 7 (2020), 1570–1581. https://doi.org/10.1109/TPAMI.2020.2986777

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021a. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4690–4699.

Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021b. NeRF–: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021).

Mark Weber, Huiyu Wang, Siyuan Qiao, Jun Xie, Maxwell D Collins, Yukun Zhu, Liangzhe Yuan, Dahun Kim, Qihang Yu, Daniel Cremers, et al. 2021. DeepLab2: A TensorFlow Library for Deep Labeling. *arXiv preprint arXiv:2106.09748* (2021).

Yichen Wei, Eyal Ofek, Long Quan, and Heung-Yeung Shum. 2005. Modeling Hair from Multiple Views. In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 816–820. https://doi.org/10.1145/1186822.1073267

Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. 2020. Multi-view neural human rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1682–1691.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2021. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9421–9431.

Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. 2021. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. *Advances in Neural Information Processing Systems* 34 (2021).

Lan Xu, Zhuo Su, Lei Han, Tao Yu, Yebin Liu, and Lu Fang. 2020. UnstructuredFusion: Realtime 4D Geometry and Texture Reconstruction Using Commercial RGBD Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 10 (2020), 2508–2522. https://doi.org/10.1109/TPAMI.2019.2915229

Ling-Qi Yan, Chi-Wei Tseng, Henrik Wann Jensen, and Ravi Ramamoorthi. 2015. Physically-Accurate Fur Reflectance: Modeling, Measurement and Rendering. *ACM Trans. Graph.* 34, 6, Article 185 (oct 2015), 13 pages. https://doi.org/10.1145/2816795.2818080

Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. 2021. BANMo: Building Animatable 3D Neural Models from Many Casual Videos. *arXiv preprint arXiv:2112.12761* (2021).

Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. 2021. iNeRF: Inverting Neural Radiance Fields for Pose Estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. 2019. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. PlenOctrees for Real-time Rendering of Neural Radiance Fields. In *ICCV*.

Cem Yuksel, Scott Schaefer, and John Keyser. 2009. Hair Meshes. *ACM Trans. Graph.* 28, 5 (dec 2009), 1–7. https://doi.org/10.1145/1618452.1618512

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-Adaptive Neural Networks for Quadruped Motion Control. *ACM Trans. Graph.* 37, 4, Article 145 (jul 2018), 11 pages. https://doi.org/10.1145/3197517.3201366

Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. 2021. Editable Free-Viewpoint Video Using a Layered Neural Representation. *ACM Trans. Graph.* 40, 4, Article 149 (jul 2021), 18 pages. https://doi.org/10.1145/3450626.3459756

Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J Black. 2019. Three-D Safari: Learning to Estimate Zebra Pose, Shape, and Texture from Images" In the Wild". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5359–5368.

Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 2017. 3D menagerie: Modeling the 3D shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6365–6373.