SAILOR: Synergizing Radiance and Occupancy Fields for Live Human Performance Capture

ZHENG DONG, State Key Laboratory of CAD&CG, Zhejiang University, China
KE XU, City University of Hong Kong, China
YAOAN GAO, State Key Laboratory of CAD&CG, Zhejiang University, China
QILIN SUN, The Chinese University of Hong Kong, Shenzhen and Point Spread Technology, China
HUJUN BAO, State Key Laboratory of CAD&CG, Zhejiang University, China
WEIWEI XU∗, State Key Laboratory of CAD&CG, Zhejiang University, China
RYNSON W.H. LAU, City University of Hong Kong, China

Fig. 1. We propose SAILOR, a novel method for human free-view rendering and reconstruction from very sparse (e.g., 4) RGBD streams with low latency. Our approach learns a hybrid representation of radiance and occupancy fields, which can handle unseen performers without fine-tuning and generate high-quality appearance details in the novel view. In addition, it naturally supports portrait rendering and reconstruction without re-training on the corresponding datasets.

Immersive user experiences in live VR/AR performances require a fast and accurate free-view rendering of the performers. Existing methods are mainly based on Pixel-aligned Implicit Functions (PIFu) or Neural Radiance Fields (NeRF). However, while PIFu-based methods usually fail to produce photorealistic view-dependent textures, NeRF-based methods typically lack local geometry accuracy and are computationally heavy (e.g., dense sampling of 3D points, additional fine-tuning, or pose estimation). In this work, we propose a novel generalizable method, named SAILOR, to create high-quality human free-view videos from very sparse RGBD live streams. To produce view-dependent textures while preserving locally accurate geometry, we integrate PIFu and NeRF such that they work synergistically by conditioning the PIFu on depth and then rendering view-dependent textures through NeRF.

Specifically, we propose a novel network, named SRONet, for this hybrid representation. SRONet can handle unseen performers without fine-tuning. Besides, a neural blending-based ray interpolation approach, a tree-based voxel-denoising scheme, and a parallel computing pipeline are incorporated to reconstruct and render live free-view videos at 10 fps on average. To evaluate the rendering performance, we construct a real-captured RGBD benchmark from 40 performers. Experimental results show that SAILOR outperforms existing human reconstruction and performance capture methods.

CCS Concepts: • Computing methodologies → Image-based rendering; Mesh geometry models.

Additional Key Words and Phrases: human performance capture, high-quality human free-view videos, occupancy and radiance fields, hybrid representation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. 0730-0301/2023/12-ART205 $15.00
https://doi.org/10.1145/3618370

ACM Reference Format:
Zheng Dong, Ke Xu, Yaoan Gao, Qilin Sun, Hujun Bao, Weiwei Xu, and Rynson W.H. Lau. 2023. SAILOR: Synergizing Radiance and Occupancy Fields

ACM Trans. Graph., Vol. 42, No. 6, Article 205. Publication date: December 2023.
1 INTRODUCTION

The creation of free-viewpoint videos featuring humans is an actively researched topic in the fields of computer graphics and vision. It serves as a critical component for a wide range of applications, including virtual and augmented reality, distance education, and telecommunications. To provide immersive experiences to the users, applications like remote presence and teleconferencing [Orts-Escolano et al. 2016; Zhang et al. 2022b] require capturing high-fidelity 3D human models from consumer-affordable capture rigs in real-time with low latency from live video streams.

Recently, neural implicit representations have been widely used in human performance capture. Pixel-aligned implicit functions (PIFu) can reconstruct dynamic 3D human body surface meshes with details and textures [Dong et al. 2022; Feng et al. 2022; Li et al. 2020a,b; Saito et al. 2019, 2020; Yu et al. 2021b], where the surface meshes are extracted from a reconstructed occupancy field, and the surface textures are obtained using a trained network for predicting the RGB colors of surface points. Neural radiance fields (NeRF) are another increasingly popular family of techniques that leverage coordinate-based networks to encode volumetric density and color fields. It may synthesize photorealistic novel-view images with highly detailed 3D space sampling [Gafni et al. 2021; Mildenhall et al. 2020; Pumarola et al. 2021; Tretschk et al. 2021]. However, both two lines of methods still have weaknesses. First, the surface-texture-based rendering method of PIFu may lead to blurred rendering results in some cases, and PIFu cannot handle view-dependent effects or the transparency of human hairs. Second, NeRF suffers from slow rendering speed and weak generalization ability. Latest generalizable NeRF methods may fail to handle unseen subjects nor novel image rendering from sparsely captured views [Chen et al. 2021b; Gao et al. 2022; Jiang et al. 2022; Kwon et al. 2021; Peng et al. 2021b]. Typically, fine-tuning is necessary to achieve high-quality rendering results for a new subject [Gafni et al. 2021; Lin et al. 2022; Shao et al. 2022b; Wang et al. 2021a; Yu et al. 2021a]. Hence, developing a generalizable method that can create live and photorealistic human free-viewpoint videos with sparse capture rigs is still challenging.

In this work, we aim to address the above challenge with two observations. First, we observe that the PIFu and NeRF representations can be synergized through depth information in such a way that while NeRF uses global radiance information to synthesize high-quality views, its shape ambiguity can be reduced by incorporating the occupancy field in PIFu, which helps guide surface reconstruction. Second, we observe that synergizing the PIFu and NeRF representations demands for accurate depth information, as relying on image features solely may still yield unreliable shape estimation results in the occupancy fields, especially under sparse capture settings. If we can obtain accurate depths, we may constrain the PIFu surface field and align image features better with surfaces to model colors in the radiance fields, resulting in a generalization of the learned human model to novel poses and appearances.

Based on the above two observations, we propose a novel human performance capture method, SAILOR, for creating high-quality free-view videos from sparse (e.g., 4) RGBD video streams. It is a generalizable method that can handle unseen performers without fine-tuning (see Fig. 1). Our method has three main steps to process the RGBD video inputs, at the core of which is a novel neural 3D human representation that takes both advantages of PIFu and NeRF for high-quality geometry and photorealistic appearance reconstruction. First, we train a U-Net-like [Ronneberger et al. 2015] depth denoising network to reduce noise and fill possible holes in the raw depth maps. We condition our method on depth denoising, as it provides readily robust geometry cues for correcting topological errors caused by unseen performers/gestures in the live streams. Second, we propose a novel network (called SRONet) to model neural performers through a combination of occupancy and neural radiance fields. Specifically, SRONet constructs 3D human surfaces in the soft occupancy field based on the pixel-aligned denoised depth features, and renders high-quality appearances in the color field conditioned on both image-aligned color features and geometry features. We also construct a tree structure [Liu et al. 2020; Lombardi et al. 2021] from denoised depths, based on which a novel voxel-denosing scheme is proposed to constrain the sampling points inside voxels falling on the body surface during inference. Third, a neural blending-based ray interpolation scheme is proposed to render novel-view images in 1K resolution with small computational overheads.

We note that the Unisurf [Oechsle et al. 2021] may be closely related to ours, as we both combine the occupancy and radiance fields in one model. Specifically, Unisurf [Oechsle et al. 2021] is proposed for solid object reconstruction, which essentially adopts a coarse-to-fine strategy to locate and refine solid object surfaces via volume rendering of NeRF and surface rendering in the occupancy field, respectively. However, as Unisurf uses the occupancy field for rendering, it does not incorporate the pixel-aligned features of PIFu and therefore is a non-generalizable method. In practice, our method runs two magnitudes faster than Unisurf in rendering and can handle unseen performers without any fine-tuning.

To summarize, this work makes the following contributions:

- A novel human performance capture method (called SAILOR) with a hybrid network (SRONet), which synergizes occupancy and radiance fields conditioned on a depth denoising process and its resulting pixel-aligned RGBD features. SAILOR is generalizable to handle unseen performers under a sparse RGBD camera setting without fine-tuning.

- An applicable system that incorporates a tree-based structure, a voxel denoising scheme, a neural blending-based ray interpolation approach, and a parallel computing pipeline. It creates free-view rendering results in 1K resolution at 10 fps on average.

- A real-captured human benchmark, which contains multi-view RGBD videos captured from 40 performers (with ~4,000 frames per person), covering various actions.

Extensive experiments on performers with diverse gestures, motions, and clothing, verify the effectiveness of SAILOR against existing human performance capture methods in terms of reconstruction and rendering accuracy.

2 RELATED WORK

2.1 Monocular Human Performance Capture

A line of methods is proposed to use monocular videos for human performance capture. Xu et al. [2018] propose the first markerless...
deep method, which computes a textured template mesh of static T-pose for each performer, and models the articulated motions and non-rigid surface deformations via a combination of 2D/3D pose estimations and silhouette-based surface refinement, respectively. The T-pose (or A-pose) mesh is then widely adopted [Dou et al. 2017, 2016; Habermann et al. 2021, 2019, 2020; Li et al. 2021; Newcombe et al. 2015a, 2011; Su et al. 2020; Yu et al. 2018], based on which motions are modeled by estimating the non-rigid deformations from the template mesh. Xiang et al. [2020] propose the statistical deformation models for clothing capturing. Zhao et al. [2022c] propose a dynamic surface network to predict dynamic offsets and texture maps based on the SMPL [Loper et al. 2015] template and a reference-based rendering network that combines the predicted offsets and texture maps to render novel human avatar images.

**Pixel-aligned implicit functions (PIFu)** [Saito et al. 2019, 2020] show promising high-resolution reconstruction results of textured objects compared with previous 3D representations (e.g., SMPL [Loper et al. 2015; Pavlakos et al. 2019], voxels [Zheng et al. 2019], points [Yifan et al. 2019], and meshes [Allieck et al. 2019; Zhu et al. 2022]). Li et al. [2020a] propose an octree-based surface localization method and a mesh-free rendering method to apply PIFu for monocular human performance capture. Later methods incorporate pre-computed template meshes [Li et al. 2020b], human parsing maps [Chan et al. 2022b], 3DMM [Cao et al. 2022], and SMPL [Chan et al. 2022a; Feng et al. 2022; Xiu et al. 2022; Zheng et al. 2021] with implicit functions to represent 3D humans with motions.

**Neural radiance field (NeRF)** [Mildenhall et al. 2020] is another popular 3D implicit representation that utilizes classic volumetric rendering to produce free-view images. To handle dynamic scenes, some methods [Park et al. 2021a; Peng et al. 2023; Pumarola et al. 2021; Tretschk et al. 2021] extend NeRF by constructing continuous deformation fields. The deformation fields typically map the observed coordinates to canonical coordinates of a template of the target, following the non-rigid reconstruction-and-tracking scheme [Newcombe et al. 2015b]. Peng et al. [2021b] construct the deformation field by leveraging a set of structured latent codes to represent the performer’s local geometry and appearance. In [Chen et al. 2021b], 3D positions, shapes, and poses are incorporated to guide the construction of the deformation field. These methods [Chen et al. 2021b; Jiang et al. 2022; Peng et al. 2021b] rely on parametric human models [Joo et al. 2018; Kocabas et al. 2020; Loper et al. 2015] to handle human topology changes under motions. Recently, Weng et al. [2022] proposed to model skeletal rigid and non-rigid motions via a discrete grid and a continuous field, respectively.

Some other methods [Gafni et al. 2021; Hu et al. 2023; Su et al. 2022; Xian et al. 2021] handle dynamic scenes by conditioning the NeRF on additional inputs to change the radiance field of the scene directly. Xian et al. [2021] condition the NeRF on the timestamps of the input RGBD video (where D is estimated by a video depth estimation method), and use depth as supervision to refine the scene geometry. Gafni et al. [2021] condition the NeRF on a set of latent codes (computed from video frames and the background image) and a 3D morphable model (for tracking facial expressions and poses). The HyperNeRF method [Park et al. 2021b] combines the deformation field and the conditioning networks on latent deformation and appearance codes. Recently, Kim et al. [2023] extended the HumanNeRF [Weng et al. 2022] to support rendering of multiple performers, by introducing a set of latent identity codes and pose-conditioned codes.

While monocular videos are convenient and of lower cost, a fundamental limitation of monocular methods is the shape-radiance ambiguity caused by partial occlusions. The difference is that our method utilizes a sparse (e.g., 4) set of RGBD cameras for full-body human performance capture and can effectively reduce this ambiguity by integrating PIFu and NeRF representations.

### 2.2 Volumetric Human Performance Capture

Volumetric capture methods [De Aguiar et al. 2008; Vlasic et al. 2008] typically leverage multiple cameras to cover the whole capture volume of performers. A group of methods [Collet et al. 2015; Guo et al. 2019; Işık et al. 2023; Jiakai et al. 2021; Liu et al. 2009; Vlasic et al. 2009; Wang et al. 2021b, 2022; Zhang et al. 2022a; Zhao et al. 2022a] leverage high-end studio (tens up to hundreds of) cameras for accurate 3D reconstruction. Multi-view RGB stereo information is used in [Işık et al. 2023; Wang et al. 2021b, 2022], while RGB and infrared (IR) are combined with silhouette [Collet et al. 2015] and depth [Guo et al. 2019]. These methods are typically unaffordable for novice users.

Recently, a set of methods has been proposed to capture human performance from a sparse set (less than ten views) of RGB(D) cameras. Wu et al. [2020] use PointNet+ [Qi et al. 2017] to extract 3D point cloud features and design a CNN to render novel images, while the newly rendered images are used to help further improve the visual hull reconstruction [Matusik et al. 2000]. This method is limited by the low-resolution noisy point cloud representation. A few methods [Dong et al. 2022; Saito et al. 2019, 2020; Shao et al. 2022a; Yu et al. 2021b] combine multi-view RGB(D) information with the PIFu representation to produce accurate geometry reconstruction results. However, learning the colors of surface points from image features of sparse views makes these PIFu-based methods difficult to produce novel view-dependent and photorealistic appearances.

Another line of methods is built upon the generalizable NeRF [Chen et al. 2021a; Yu et al. 2021a], which conditions the NeRF on pixel-aligned image features. Some methods [Gao et al. 2022; Kwon et al. 2021] factorize NeRF into a canonical NeRF and a deformation field, and model the deformation field through learning mappings from the surfaces of 3D body parametric models to the 3D volume. In [Peng et al. 2021a], a neural blend weight field in canonical space is combined with the skeleton-driven deformation [Lewis et al. 2000] to generate deformation fields. Wang et al. [2021a] combine NeRF with image-based rendering [Chen and Williams 1993;Debevec et al. 1998; Hedman et al. 2018], in which the colors and densities of target view are computed by aggregating the image features of neighboring source views. Mihajlovic et al. [2022] condition the NeRF on 3D keypoints to encode robust spatial 3D information. Some methods also condition NeRF with SMPL or pose [Liu et al. 2021], and texel-aligned (pose, image, and camera position) features [Remelli et al. 2022] for drivable volumetric avatar rendering.

Recent endeavors are made to mitigate the geometry ambiguities of NeRF. Zhao et al. [2022b] construct a pose-based deformation field for modeling geometry under motions. Shao et al. [2022b] propose to regress both occupancy and densities from multi-view RGB features.
in which ground-truth occupancy can be involved for geometry supervision. Lin et al. [2022] propose to estimate the depth probability distribution (i.e., depth and confidence maps) for constraining the spatial sampling of NeRF near the surfaces. Nonetheless, deriving geometry proxies (i.e., body parametric models, surface occupancy, and depth distribution in [Lin et al. 2022; Shao et al. 2022b,c; Zhao et al. 2022b]) based on RGB information is often not reliable. The inaccurate local geometry further results in visual blurriness and artifacts on the appearances, which makes the fine-tuning for unseen performers inevitable in these methods.

In this work, we propose a depth-conditioned hybrid representation of PIFu and NeRF to address this geometry/appearance ambiguity problem of unseen performers. By incorporating accurate depth, we show that pixel-aligned RGBD features enable accurate and generalizable surface reconstructions and can guide NeRF to produce high-fidelity appearances in near-real-time.

3 OUR METHOD

Our method aims to generate high-quality, and high-resolution free-view videos in near-real time, given $N$ RGB-D streams of $\{I^i, D^i\}_{i=1,...,N}$ captured by a sparse set of Kinect-V4 sensors, where I and D represent the RGB and depth images, respectively, and $N$ is set to 4 in our implementation.

As illustrated in Fig. 2, our method contains five steps: (1) **Image-conditioned Depth Denoising** $F_d$ removes noise and completes the holes in the noisy multi-view depth images. (2) **Two-layer Tree Construction** $T$ divides the human-body volume into two levels, and stores parent-child voxels of the volume as two sequences of nodes in the GPU, based on the denoised depths of $F_d$. (3) For a ray emitted from the target view, **Ray-Voxel Intersection** records the indexes of the two-level voxels that intersected with the ray, along with the depths of the intersected points. **Points Sampling** then records the depths in the target view of the sampled points within each voxel. These sampled points along the ray are located near the 3D human surface for efficient appearance rendering. (4) For each sampled point on the ray, our SRONet predicts its RGB and soft occupancy values, based on the denoised depths and RGB images. The color and depth of the sampled pixel in the target view are computed via a blending function. This step processes the rays for an image at 1/4

![Fig. 2. Given RGBD streams captured by 4 Azure Kinect sensors as inputs, (1) a Depth Denoising Module first removes noise and fills the holes of raw depths conditioned on the RGB images. (2) With the denoised depths, a Two-Layer Tree Structure is constructed to store the global geometry in discretization. (3) Efficient Ray-Voxel Intersection and Points Sampling are performed for rendering. (4) A novel SRONet network is proposed to synergize the radiance and occupancy fields, for 3D reconstruction and free-view rendering. (5) The outputs of SRONet are then upsampled via Ray Upsampling and Neural Blending to produce the final results in 1k resolution.]

![Fig. 3. Visualization of our depth denoising results and their fused point clouds on our real captured RGBD images. Our depth denoising network can reduce noise and fill in the missing regions (e.g., hair and hand regions marked in black dashed boxes) for the full-body and portrait inputs.](image-url)
Accurate geometry information from the depth plays a vital role in rendering the human body, we exploit the geometric cue (i.e., denoised depths) by computing the fused point cloud \( P_{rf} \) from \( D_{rf} \) and leveraging \( P_{rf} \) to construct a Two-Layer Tree (denoted as \( T \)) in the GPU end.

Construction of \( T \). Fig. 4 shows the two-layer tree construction process. First, we adopt the TSDF-Fusion [Newcombe et al. 2011] to convert the denoised depths \( D_{rf}^{i} (i = 1, \ldots, N) \) into a full-body point cloud \( P_{rf} \). Second, based on the fused TSDF volume \( V_{tsdf} \), we binarize \( V_{tsdf} \) to generate an occupied volume \( V_{occ} \). Third, we merge the valid voxels with a value 1 into large voxels in a ratio of \( 4^3 : 1 \) (i.e., each large voxel can have up to 64 child voxels). Finally, we store all the valid voxels as a global list \( L_{v} \) in the GPU, where each node in \( L_{v} \) records the index, size, and position (in world coordinate) of the corresponding voxel. We have implemented \( T \) with CUDA acceleration (~6ms), which supports the storage of multiple batches (or performers) simultaneously.

Voxel Denoising via Post-merging. The raw point cloud \( P_{rf} \) often has undesirable floating voxels (red circle in Fig. 4). Hence, we apply a post-merging step to eliminate these voxels during inference. Specifically, we first use our SRONet (Sec. 3.3) to construct a soft occupied volume \( V_{occ}^{m} \) where voxel values are in \([0, 1]\). We then fuse the two volumes \( V_{occ}^{m} \) and \( V_{occ} \) with a union operation, as:

\[
V_{occ}^{m}(x) = \begin{cases} B(V_{occ}^{m}(x), \beta) & |V_{occ}(x) - V_{occ}^{m}(x)| \geq \gamma \\ 0 & \text{else} \\ \end{cases}, \tag{1}
\]

where \( B(\cdot, \beta) \) is a binarization function with threshold \( \beta \), and \( \gamma \) is an occupancy threshold to eliminate external voxels.

To render a novel-view image, we project rays from pixels of the target view, and perform Ray-Voxel Intersection to identify voxels in both levels of \( T \) that are intersected with the rays, and perform Points Sampling to sample the points inside the intersected voxels on the rays (See the supplemental for details).

---

3.1 Image-conditioned Depth Denoising: \( T_{d} \)

Accurate geometry information from the depth plays a vital role in our method for rendering accuracy and stability. However, the raw depths acquired by Kinect cameras are often noisy and incomplete. To refine the raw depth, we train an UNet-like [Ronneberger et al. 2015] depth denoising network (denoted as \( F_{d} \)) to perform the denoising process as \( D_{rf}^{i} = F_{d}(I^{i}, D^{i}), \) where \( I^{i} \) and \( D^{i} \) are the input RGB and Depth images of view \( i \), respectively. \( F_{d} \) helps remove high-frequency noise, fill the missing parts, and output a reliable depth map \( D_{rf}^{i} \) for our rendering system. To train our depth denoising network \( F_{d} \), we employ a training dataset with high-quality 3D human scans and simulate the depth noise on the ground-truth depth maps (see Sec. 5). The designed loss functions, network structure, and training details of \( F_{d} \) are provided in the supplemental material.

Fig. 3 shows that our depth denoising module performs well on our real captured data, i.e., removing noise and filling in the missing regions (e.g., black dashed boxes). However, due to the sparse capture setting, full-body fused points from \( D_{rf}^{i} \) may still have holes in some invisible areas (e.g., yellow dashed boxes). We construct the two-layer tree structure to cover such regions as described next.

3.2 Two-layer Tree Construction: \( T \)

To constrain the rendering range to be near the 3D surface of the human body, we exploit the geometric cue (i.e., denoised depths), by computing the fused point cloud \( P_{rf} \) from \( D_{rf}^{i} \) and leveraging \( P_{rf} \) to construct a Two-Layer Tree (denoted as \( T \)) in the GPU end.

ACM Trans. Graph., Vol. 42, No. 6, Article 205. Publication date: December 2023.
Fig. 6. Overview of SRONet. (1) OccNet takes denoised depths and a 3D point as inputs, and predicts the soft occupancy \( \alpha_{\text{o}} \in [0,1] \) value of the point. (2) ColorNet predicts the color value of the point based on the RGB images, view directions, and the geometric features. (3) We use the soft occupancy values to compute the weights for blending colors of the sampling points on a ray, to produce the final pixel color of the novel view.

**Depth-conditioned Occupancy Field.** For a sampled point \( x \) on the emitted ray \( i \), we first predict its soft occupancy value \( \alpha_{\text{o}} \in [0,1] \) by aggregating the pixel-aligned depth features of \( D_{\text{rf}}(i) \leftarrow f_{D}(i) \), \( i = 1, ..., N \), where \( \alpha_{\text{o}} \) is the probability of the point \( x \) locating inside the human body (\( \alpha_{\text{o}} = 0.5 \) indicates that the point is on the surface). We use a sub-network, named OccNet, to model this occupancy field \( F_{\text{o}} \) as:

\[
F_{\text{o}}(x, D_{\text{rf}}) = f_{2}(\text{Avg}(f_{1}(W^{i}(x), c^{i}(x)))_{i=1, ..., N}) = \alpha_{\text{o}},
\]

where \( W^{i} = \text{E}_{d}(D_{\text{rf}}(i)) \) represents the depth feature map of the \( i \)-th view, and \( \text{E}_{d}() \) is the depth encoder. For the projected 2D image coordinate \( \pi(x) \) and depth \( z^{i} \) of \( x \) in view \( i \), \( W^{i}(x) \) is the fetched depth feature vector at \( \pi(x) \) and \( c^{i}(x) = [z^{i}, p^{i}(x)] \), where \( p^{i}(x) \in [-\delta_{p}, \delta_{p}] \) is the truncated PSDF value computed based on \( D_{\text{rf}} \) and \( z^{i} \), similar to [Dong et al. 2022]. In Eq. 2, \( W^{i}(x) \) along with \( c^{i}(x) \) are fed into the first implicit function \( f_{1} \) to obtain the geometric features. These features are then processed by an average pooling operator \( \text{Avg} \) and further fed into the second implicit function \( f_{2} \) for occupancy querying. The queried value \( \alpha_{\text{o}} \) is used for both reconstruction and rendering.

**Geometry-conditioned Color Field.** We predict the view-dependent color value \( c_{\text{i}} \in \mathbb{R} \) by aggregating the pixel-aligned RGB features of \( \Gamma(i) \leftarrow \Gamma(i) \), \( i = 1, ..., N \), conditioned on the local view direction \( d^{i} \) and geometric features \( \hat{f}_{\text{geo}} \), where \( d^{i} = R^{i}d \) with \( d \) being the view direction in world coordinate, and \( \hat{f}_{\text{geo}} = f_{3}(W^{i}(x), c^{i}(x)) \). We use another sub-network, named ColorNet, to model color field \( F_{\text{c}} \) as:

\[
F_{\text{c}}(x, l, d) = f_{3}(\mathcal{H}(\{f_{3}(W^{i}(x), \hat{f}_{\text{geo}}(d^{i}, \text{rgb}^{i}))\}_{i=1, ..., N}) = c_{\text{i}}.
\]

where \( M^{i} = E_{c}(l^{i}) \) is the rgb feature map in view \( i \) and \( E_{c}(\cdot) \) is the rgb encoder. \( M^{i}(x) \) and \( \text{rgb}^{i} \in \mathbb{R}^{C} \) are the fetched rgb features of the point, respectively. \( f_{3} \) and \( f_{3} \) are both implicit functions to process features. We implement the feature fusion process \( \mathcal{H} \) as a transformer encoder [Vaswani et al. 2017] with hydra attention blocks [Bolya et al. 2023] and adopt the fully fused scheme in [Müller et al. 2021] to accelerate this process.

**Rendering.** To produce the final color \( \hat{C}(i) \) for the emitted ray \( i \), we use the unified surface and volume rendering function in [Oechsle et al. 2021] to blend color vector \( c_{\text{i}} \) for each sampled point \( x \), as:

\[
\hat{C}(i) = \sum_{i=1}^{M} \alpha_{\text{o}}(i) \prod_{j<i} (1 - \alpha_{\text{o}}(j)) c_{\text{x}}(i),
\]

where \( \alpha_{\text{o}}(i) = \sum_{j<i} (1 - \alpha_{\text{o}}(j)) \) is the blending weight for the \( i \)-th point \( x_{i} \) sampled on ray \( I \), and \( M \) is the number of sampled points. When \( x_{i} \) is far from the body surface, the occupancy value \( \alpha_{\text{o}}(i) \) is close to 0 or 1. Hence, the weight \( \alpha_{\text{o}} \) tends to have high response values only for the points near the surface. This aligns with our motivation of sampling points inside the surface voxels. Similarly, we compute depth \( \hat{D}(i) \) of the surface intersected with the emitted ray by blending depth \( d(i) \) of points, as:

\[
\hat{D}(i) = \sum_{i=1}^{M} \omega_{\text{o}}(i)d(i).
\]

**Optimization of SRONet.** We adopt two loss functions to supervise the reconstruction and rendering process of SRONet.

(1) **Geometry and Color Synergistic Loss.** We first sample point \( y \) around the body surface (bottom part in Fig. 6), and then measure the difference between the predicted occupancy value \( \alpha_{\text{o}} \) and the ground-truth value \( \alpha_{\text{gt}} \) to train our OccNet to learn global geometric information. Meanwhile, we penalize the per-ray error between \( \hat{C}(i) \) and the ground-truth color \( C^{*}(i) \) to train both our ColorNet and OccNet for learning textures and enhancing geometric details. The two losses work in a synergistic manner, as:

\[
L_{\text{sync}} = \mu_{\text{o}} \sum_{x \epsilon S} L_{B}(\alpha_{\text{o}}, \alpha_{\text{gt}}) + \mu_{\text{c}} \sum_{i \epsilon R} L_{1}(\hat{C}(i), C^{*}(i)),
\]

where \( S \) and \( R \) denote the sampled points and rays set, respectively. \( L_{B} \) and \( L_{1} \) are the BCE loss and the smooth L1 loss, while \( \mu_{\text{o}} \) and \( \mu_{\text{c}} \) are the balancing weights.

(2) **Depth Consistency Loss.** We enhance the consistency between the predicted depth value \( \hat{D}(i) \) and the GT depth value \( D^{*}(i) \) to improve reconstruction and rendering details, as:

\[
L_{D^{*}} = \sum_{i \epsilon R} L_{2}(\hat{D}(i), D^{*}(i)),
\]

where \( L_{2} \) is the L2 loss. \( D^{*}(i) \) is fetched from the rendered GT depth map. The complete loss function for SRONet is then a combination of \( L_{\text{sync}} \) and \( L_{D^{*}} \), as:

\[
L_{\text{syn}} + \lambda_{D^{*}} L_{D^{*}}, \quad \text{where } \lambda_{D^{*}} \text{ is a balance term.}
\]

### 3.4 Appearance Upsampling

We propose a fast ray upsampling scheme to further enhance the rendered image of SRONet with higher resolution and richer details. Compared to LookinGood [Martin-Brualla et al. 2018], which enhances the rendered images, we interpolate each emitted ray into four sub-rays during the rendering. The color of each sub-ray is predicted based on the shared color, depth, and features of the emitted ray, and two high-resolution adjacent RGB inputs, via a neural blending method.
As shown in Fig. 7, for an emitted ray $r_{a} = \hat{r}_{\text{color}}, \hat{r}_{\text{rgb}}$ into each sub-ray $r_{i}$, where $i \in \{0, 1, 2, 3\}$, we first interpolate it into four sub-rays (denoted as $r_0, r_1, r_2, r_3$), whose emitted source points are consistent with $l$, but the sub-pixel positions are assigned as $(x, y), (x + 0.5, y), (x, y + 0.5), (x + 0.5, y + 0.5)$. Then, we obtain the ray color $C(I)$ and the intersected depth $D(I)$ using SRONet. We also blend the point features $\hat{f}_{\text{color}}$ (output by the hydra attention blocks $\mathcal{H}$ for each sampled point $x$) for the ray $l$, via $\sum_{i=1}^{M} \alpha(x, i) \hat{f}_{\text{color}}(i)$, to obtain the ray color features $\hat{f}_{\text{ray}}$. At last, we scatter $\hat{C}(I), \hat{D}(I)$ and $\hat{f}_{\text{ray}}$ into each sub-ray $r_{i}(i = 0, 1, 2, 3)$ to obtain the coarse sub-ray features. This ray upsampling scheme enables a $2\times$ increase in spatial resolution, while simply using more rays ($4\times$) takes about $5\times$ more inference time.

**Neural Blending Operation.** We leverage two adjacent RGB inputs (in 1K resolution) to refine the ray upsampling features, similar to [Zhao et al. 2022b]. Specifically, we first use a UNet to encode the two adjacent raw RGB images into two RGB feature maps. Given the intersected depth value $\hat{D}(I)$ of each sub-ray $r_{i}$, we compute the surface position $x \in \mathbb{R}^{3}$ and back-project the surface point to the adjacent two views to fetch the colors $C_{\text{rgb}}$, $C_{\text{rgb}}$, and the RGB features $\hat{f}_{\text{rgb}}, \hat{f}_{\text{rgb}}$. We then back-project the surface point to the adjacent two refined depth maps $D_{\text{rgb}}$, $D_{\text{rgb}}$ to calculate the soft visibility values $O_{\text{rgb}}, O_{\text{rgb}}$, which can be written as:

$$O_{i}^{j} = \exp(-\sigma_{a} \cdot (z_{i}^{j} - d_{i}^{f}))^{2}, \tag{7}$$

where $i$ is located in $(n_{0}, n_{1})$, and $z_{i}^{j}$ is the projected depth of the surface point in view $i$. $d_{i}^{f}$ is the fetched depth value from $D_{i}^{f}$ in the 2D coordinate $p^i(x)$ of view $i$. $\sigma_{a}$ is a weight coefficient determined by depth units. Hence, $O_{i}^{j}$ tends to be 1 when $x$ is visible in view $i$, and 0 otherwise. Finally, we feed the RGB features ($\hat{f}_{\text{rgb}}, \hat{f}_{\text{rgb}}$), two local view directions ($d_{\text{rgb}}$ and $d_{\text{rgb}}$), two visibility values ($O_{\text{rgb}}$ and $O_{\text{rgb}}$), along with the ray color features ($\hat{f}_{\text{ray}}$) into our neural blending network $\mathcal{F}_{\text{b}}$ to obtain the blending weights $W_{x}$, as:

$$\mathcal{F}_{\text{b}}(x, l, D_{\text{rgb}}, d) = f_{\text{b}}(\hat{C}(I), \hat{D}(I)) = \hat{f}_{\text{ray}}(\hat{C}(I), \hat{D}(I)) = \hat{W}_{x}, \tag{8}$$

where $f_{\text{b}}$ is the implicit function, and $W_{x} \in \mathbb{R}^{3}$ is used to blend the two adjacent colors and the ray color for point $x$. The final color $\hat{C}_{r}$ for a sub-ray $r$ can then be obtained via: $W_{x} \cdot [C_{\text{rgb}}, C_{\text{rgb}}, \hat{C}(I)]$. See Fig. 7 for illustration and refer to supplemental for training details.

### 3.5 Parallel Acceleration

We design a parallel acceleration method to leverage multiple GPUs and a single CPU (2 Nvidia RTX 3090 and an Intel i9-13900k in this work) to accelerate the rendering. It aims to distribute the workload regarding input views and ray computations among GPUs, and build a pipeline to reduce the processing latency for each operation.

<table>
<thead>
<tr>
<th>Stages</th>
<th>Operations</th>
<th>Time w/o acc.</th>
<th>Time w/ acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{d}}$</td>
<td>Depth denoising</td>
<td>$\approx 74$ms</td>
<td>$\approx 101$ms</td>
</tr>
<tr>
<td>$T_{\text{f}}$</td>
<td>Encoding RGBD images in SRONet</td>
<td>$\approx 76$ms</td>
<td>$\approx 91$ms</td>
</tr>
<tr>
<td>$T_{\text{u}}$</td>
<td>High-resolution RGB encoding in neural blending</td>
<td>$\approx 25$ms</td>
<td>$\approx 6$ms</td>
</tr>
<tr>
<td>$T_{\text{r}}$</td>
<td>Building two-layer tree</td>
<td>$\approx 6$ms</td>
<td>-</td>
</tr>
<tr>
<td>$b$</td>
<td>Soft Visibility</td>
<td>$\approx 6$ms</td>
<td>-</td>
</tr>
<tr>
<td>$b$</td>
<td>Intersection Detection of voxels intersected by rays.</td>
<td>$\approx 10$ms</td>
<td>-</td>
</tr>
<tr>
<td>$b$</td>
<td>Points Sampling Sampling points within voxels along the rays.</td>
<td>$\approx 2$ms</td>
<td>-</td>
</tr>
<tr>
<td>$b$</td>
<td>Ray Upsampling Ray upsampling and neural blending</td>
<td>$\approx 27$ms</td>
<td>$\approx 3$ms</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>$\approx 483$ms</td>
<td>$\approx 10$ms</td>
</tr>
</tbody>
</table>

Table 1. The running time for each stage of our pipeline w/o and w/ acceleration is reported. Note that we use the three alternative streams for w/ acc., which further reduces the sum time of $\leq 105$ms by 9.5%-19%.

![Fig. 8. The parallel computing pipeline of acceleration.](https://example.com/fig8.png)

The final color $\hat{C}_{r}$ for a sub-ray $r$ can then be obtained via: $W_{x} \cdot [C_{\text{rgb}}, C_{\text{rgb}}, \hat{C}(I)]$. See Fig. 7 for illustration and refer to supplemental for training details.
Specifically, we divide the SAILOR pipeline into three groups of operations, and in each GPU we accelerate these groups of operations with three alternative data streams: (1) I/O (CPU to GPU) and depth denoising; (2) two-layer tree building and RGBD images encoding (including RGBDs encoding in SRONet and the RGBs encoding in $T_d$); and (3) ray querying in SRONet and neural blending-based ray upsampling. Finally, the post-processing and IO operations are used to reshape the color vectors of the emitted rays into images for display. We allocate half of the workload of each group to one GPU for parallel inference acceleration.

As illustrated in Fig. 8, GPU-0 handles the RGBD data of views 1 and 3, while GPU-1 handles those of views 2 and 4. Each $T_d$ in the GPU predicts two refined depths, all of which are sent to the CPU for data synchronization. The CPU then sends the four reduced depth maps back to two GPUs for the two-layer tree construction. The interaction between the CPU and GPUs for image encoding performs in the same way as $T_d$. We allocate half of the total rays (i.e., $512^2/2$) to each GPU, which are led into SRONet for ray querying and subsequent upsampling. Moreover, we utilize the surface rendering scheme to accelerate ray querying. After ray synchronization, we obtain the color vector of $4 \times 512^2$ logits, which is reshaped to an RGB image in 1K resolution as the final rendering result. We also use TensorRT with half-precision to accelerate $T_d$, our SRONet, and the neural blending module. Besides, we adopt the fully fused scheme [Müller et al. 2021] to accelerate all the implicit functions $f_i(i = 1, \ldots, 6)$ and the hydra attention operation $H$. Tab. 1 reports the time cost of each main operation in SAILOR. The accelerated SAILOR can finally render the free-view video in 1K resolution at a single timestamp (upper row) and over a time period (bottom row). The system performs undistortion (∼1.5ms), background-matting (∼4ms using TensorRT), image deformation (∼1.2ms, including cropping, padding, and rotation), and depth-to-color alignment (∼0.13ms) with acceleration to obtain the RGBD inputs for SAILOR.

4 OUR DATASET

We construct a real-captured human dataset, consisting of 160k+ frames of multi-view RGBD dynamic human motions, captured by Azure Kinect-V4 from 40 performers (20 female and 20 male actors). Each actor performed approximately 4,000 frames of action, wearing daily clothing. Typical actions are listed in Fig. 13.

The dataset for each performer contains captured RGBD sequences in 8 views, the pre-calibrated camera internal and external parameters, and the foreground segmentation RGB images (produced by background-matting-v2 [Lin et al. 2021]). The resolutions of the captured RGB and depth data are $2,560 \times 1,440$ and $1,024 \times 1,024$, respectively. For novel-view rendering evaluation, we use RGBD images of 4 fixed perspective views (the interval between two adjacent views is 90 degrees, and the indexes of cameras are 0, 4, 6, 7, respectively) as inputs. RGBD images of the other four views (i.e., indexes of 1, 2, 3, 5) are used to evaluate rendering quality.

Our dataset contains various actions, diverse facial expressions, and complex geometries. Fig. 10 shows some examples. Our dataset can be considered a challenging human performance capture benchmark for evaluating SAILOR and other rendering methods. Fig. 11 shows some rendering results from SAILOR on this dataset.

5 RESULTS

Training and Evaluation. We train and evaluate our method using the public available THuman2.0 [Yu et al. 2021b] dataset, which contains 500 high-quality 3D human scans. We split the dataset into
We compare our method to five state-of-the-art human reconstruction methods (with available codes), including PixelNeRF [Yu et al. 2021a], IBRNet [Wang et al. 2021a], MPSNeRF [Gao et al. 2022], NHP [Kwon et al. 2021], KeypointNeRF [Mihajlovic et al. 2022], NPBG++ [Rakhimov et al. 2022] and PIFu(RGBD) [Saito et al. 2019]. For fair comparisons, we either re-train (unavailable pre-trained weights) or fine-tune (available pre-trained weights) these methods on the training set of THuman2.0 dataset [Yu et al. 2021b]. We also report comparisons on the real-captured examples (Fig. 13). The average L1(cm) distances to depth maps for 4 holdout views are 7.040 (PIFuHD), 6.385 (IPNet), 0.9051 (GTPIFu), and 0.9040 (Ours), showing that our method plays favorably against them.

**Visual Comparison.** Fig. 12 shows comparisons between our results and those of the PIFuHD [Saito et al. 2020], IPNet [Bhatnagar et al. 2020], and GTPIFu [Dong et al. 2022]. While PIFuHD [Saito et al. 2020] and IPNet [Bhatnagar et al. 2020] tend to produce obvious geometric artifacts under sparse views (Fig. 12(b,c)), the results of GTPIFu [Dong et al. 2022] (Fig. 12(d)) and ours (Fig. 12(e)) are more accurate, as we both exploit the robust geometric cues from the depth denoising. The face regions of GTPIFu [Dong et al. 2022] may contain more high-frequency details than ours, as they leverage another PIFu to model the face regions separately. However, this is computationally heavy. Our results contain more accurate details on some body regions (e.g., wrinkles of clothes) than those of GTPIFu, since the local high-frequency body geometry information may be suppressed by the joint optimization of depth denoising and occupancy prediction in [Dong et al. 2022]. In contrast, the joint optimization of the occupancy and color fields in our SRONet exploits the ground-truth 3D and RGBD signals for capturing more high-frequency geometric details.

### 5.2 Comparisons of Rendering

We compare SAILOR to six state-of-the-art generalizable rendering methods (with available codes), including PixelNeRF [Yu et al. 2021a], IBRNet [Wang et al. 2021a], MPSNeRF [Gao et al. 2022], NHP [Kwon et al. 2021], KeypointNeRF [Mihajlovic et al. 2022], NPBG++ [Rakhimov et al. 2022] and PIFu(RGBD) [Saito et al. 2019]. For fair comparisons, we either re-train (unavailable pre-trained weights) or fine-tune (available pre-trained weights) these methods on the training set of THuman2.0 dataset [Yu et al. 2021b]. PSNR, SSIM, and MAE are used to measure the rendering accuracy. We also report the Learned Perceptual Image Patch Similarity (LPIPS) [Richard Zhang and Wan 2018] for reference. We generate noise of 5 different degrees (i.e., 0.25cm, 0.5cm, 1.0cm, 1.5cm, and 2.0cm of Gaussian standard deviation) on the depth maps of the 3D meshes to evaluate our method against NPBG++ [Rakhimov et al. 2022] and PIFu(RGBD) [Saito et al. 2019].
Datasets:

- Kung Fu
- Stretching_1
- Lifting Legs
- Stretching_2
- Rocking & Walking
- THuman2.0

Fig. 13. Visualization of rendering comparisons on our real-captured dataset (row 1-5) and the THuman2.0 dataset [Yu et al. 2021b] (row 6), between our results and those of PixelNeRF [Yu et al. 2021a], IBRNet [Wang et al. 2021a], MPSNeRF [Gao et al. 2022], KeypointNeRF [Mihajlovic et al. 2022], NPBG++ [Rakhimov et al. 2022] and PIFu(RGBD) [Saito et al. 2019].
three objective metrics (i.e., PSNR, SSIM, and MAE) on the THuman2.0 dataset, and along with LPIPS on the real-captured dataset. Tab. 3 also summarizes the average rendering time of existing methods. We re-train or fine-tune all competing methods on our training dataset for a fair evaluation. The rendering time (for images in 1k resolution) is calculated using a single RTX 3090 GPU as their public codes do not include a multi-card accelerated rendering manner. The best and second best results are marked in bold and underline, respectively. Refer to the supplemental for detailed comparisons of the real-captured data on 10 independent performers.

Fig. 14. Ablation Study on our real captured dataset. 1 of 4 input RGBD views (a). Reconstruction and rendering results of different ablated versions (b to g). Our reconstruction results (h). Our rendering results without upsampling (i). Our final rendering results (j).

Fig. 15. Our method can render the geometric parts of the clothing correctly, while SMPL-based methods MPSNeRF [Gao et al. 2022] and NHP [Kwon et al. 2021] tend to fail, and PIFu(RGBD) [Saito et al. 2019] tends to render unrealistic details especially in the facial region due to the textured mesh.

Table 3. Comparisons of rendering results on the THuman2.0 dataset and our real-captured dataset, produced by our method and existing generalizable methods. We re-train or fine-tune all competing methods on our training dataset for a fair evaluation. The rendering time (for images in 1k resolution) is calculated using a single RTX 3090 GPU as their public codes do not include a multi-card accelerated rendering manner. The best and second best results are marked in bold and underline, respectively. Refer to the supplemental for detailed comparisons of the real-captured data on 10 independent performers.

Visual Comparison. Fig. 13 visualizes the qualitative comparisons on both our real-captured dataset (first five rows) and the
5.3 Ablation Study

We conduct ablation studies on both the THuman2.0 dataset [Yu et al. 2021b] (upper part) and our real captured data (lower part). The best and second best results are marked in **bold** and **underline**, respectively.

In addition, PIFu(RGBD) [Saito et al. 2019] takes advantage of the raw depth and may somewhat model clothing changes. However, it tends to generate undesirable results under novel views. In contrast, our method exploits the denoised depths with a hybrid representation of surface and color fields, thus effectively reconstructing and rendering the clothing (e.g., clothing in the bottom part of Fig. 9).

### Table 4. Ablation Study on the THuman2.0 dataset [Yu et al. 2021b] (upper part) and our real captured data (lower part). The best and second best results are marked in **bold** and underline, respectively.

<table>
<thead>
<tr>
<th>Models</th>
<th>THuman2.0 [Yu et al. 2021b] Dataset</th>
<th>Our Real Captured Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>SSIM</td>
<td>LPIPS $\times 10^{-1}$</td>
</tr>
<tr>
<td>w/o GT Depth</td>
<td>33.171 0.961 0.348 0.495</td>
<td>30.225 0.955 0.459 0.647</td>
</tr>
<tr>
<td>w/o GT Occ.</td>
<td>32.871 0.959 0.356 0.512</td>
<td>29.837 0.950 0.553 0.712</td>
</tr>
<tr>
<td>w/o Denoised Depth</td>
<td>32.588 0.955 0.425 0.548</td>
<td>29.540 0.950 0.622 0.655</td>
</tr>
<tr>
<td>Soft Occ. → Density</td>
<td>32.441 0.961 0.382 0.494</td>
<td>30.085 0.947 0.549 0.712</td>
</tr>
<tr>
<td>OccMLP → DkMLP</td>
<td>33.499 0.959 0.351 0.512</td>
<td>32.441 0.961 0.382 0.494</td>
</tr>
<tr>
<td>w/o Upsampling</td>
<td>34.865 0.967 0.291 0.467</td>
<td>30.085 0.947 0.549 0.712</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>34.882</strong> 0.969 0.354 0.392</td>
<td><strong>30.228</strong> 0.968 0.454 0.634</td>
</tr>
</tbody>
</table>

(1) We replace the soft occupancy field of our OCCNet with the density field, and replace the color blending function with the volume rendering function in NeRF (denoted as “Soft Occ. → Density”). In such a way, the hybrid representation degrades back to NeRF, and the results of both datasets show that the performance degrades accordingly (4th rows). Fig. 14(c) shows that geometric and texture errors both occur in the reconstructed and rendering results, and the denoised depths cannot effectively correct these errors.

(2) We investigate the way of DoubleField [Shao et al. 2022b] by making our OCCNet predict both occupancy and density values, for reconstruction and rendering, respectively (denoted as “OccMLP→DkMLP”). In this case, the occupancy and radiance fields are combined in an implicit manner by sharing the features in OCCNet. However, quantitative results (5th rows) show that the performance is inferior to ours. By observing Fig. 14(g) we find that this strategy suffers from local geometry errors (e.g., floating surfaces and holes in the reconstructed portrait) and produce color artifacts (e.g., greenish colors on the rendered face). These results suggest that such an implicit integration of occupancy and radiance fields may not be ideal, as the density field may affect the surface field negatively (shape-color ambiguities) in the early stage. In contrast, by discarding the density field and explicitly combining the occupancy field with the color field, SAILOR is able to avoid this issue.

### Table 5. Comparisons between the volume rendering and surface rendering on the THuman2.0 dataset [Yu et al. 2021b]. “Volume→surface” indicates replacing the color blending (Eq. 4) with the surface rendering in our SRONet. The best results are marked in **bold**. The reported time (Ours) corresponds to the process of Ray Querying in Tab. 1.

<table>
<thead>
<tr>
<th>Models</th>
<th>PSNR</th>
<th>SSIM</th>
<th>LPIPS $\times 10^{-1}$</th>
<th>MAE $\times 10^{-2}$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume → Surface</td>
<td>34.889 0.964 0.464 0.450</td>
<td>≈ 112ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>34.882 0.969 0.354 0.392</td>
<td>≈ 27ms</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

shows that $T_i$ is able to correct the colors by integrating colors of neighboring views, especially for portrait rendering.

**Rendering Scheme.** We note that surface rendering may result in a fast ray querying in SRONet, as only the surface points calculated using the depths $\hat{D}(i)$ are involved in ColorNet, which is suitable for rendering acceleration. We investigate the rendering scheme by replacing the volume rendering in our SRONet with surface rendering, denoted as "Volume→Surface". Results on the THuman2.0 dataset [Yu et al. 2021b] are reported in Tab. 5. While we can see that volume rendering and surface rendering yield close performance in terms of PSNR and SSIM, volume rendering performs better when measured with the LPIPS and MAE metrics. This provides two choices, i.e., users may switch to using surface rendering for further acceleration or adopting the volume rendering in our SRONet for a more accurate rendering result.

5.4 Generalization in New Settings

**New Camera Setting.** Our method does not require the cameras to be put exactly as the setting (refer to sec 1.4 in the supplemental) of our capture system. To verify this, we test our method on the real-captured data in Fig. 13 using a single input view (front view). The PSNR/SSIM/LPIPS values of two adjacent test views (45 degrees) are 25.356/0.939/0.0588. Moreover, the ablational studies of sensor numbers are included in our supplemental.

**New Clothing Types.** We show that our method can handle scenarios where clothing topology changes (e.g., hats, loose pants, putting on and taking off coats, in Fig. 9), long hair, as well as handling accessories such as glasses and phones to some extent. Please refer to our provided third-person videos for more details.

**Portrait Reconstruction and Rendering.** Fig. 16 shows our novel-view portrait rendering and reconstruction results of three performers using SAILOR. Here, we use 3 Azure Kinect-V4 sensors for capturing RGBDs. It shows that SAILOR can handle some sudden expression changes and complex geometries such as long hair (1st row), appearance changes such as wearing glasses and different clothes (2nd row), and can track the subject with diverse expressions over a long time period (3rd row).

6 CONCLUSION

In this paper, we have proposed a novel method (SAILOR) for creating high-quality human free-view videos from very sparse RGBD videos with low latency. The core of SAILOR is a depth-conditioned hybrid representation of PIFu and NeRF, capable of preserving locally accurate geometry and producing vivid view-dependent textures. We have proposed a novel network (SRONet) for this hybrid representation. In addition, we have designed a neural blending-based ray interpolation scheme, a tree-based voxel denoising scheme, and a parallel computing pipeline for acceleration. To evaluate rendering performance, we have constructed a real-captured RGBD benchmark of 40 performers. Experiments show that SAILOR can handle unseen performers without fine-tuning, outperform existing human reconstruction and performance capture methods, and can be applied to human portrait reconstruction and rendering.

Our method does have some limitations. First, our rendering results may exhibit temporal color/illumination flickering, overlay artifacts and fail to capture fine details (e.g., hands and fingers), due to inaccurate matting, camera calibration, and a lack of temporal modeling. Second, our dataset capture assumes a uniform illumination. Modeling temporal constraints and complex lighting can be interesting for future research.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their professional and constructive comments. This work was partially supported by a GRF grant from RGC of Hong Kong (Ref.: 11205620). Weixui Xu is supported by NSFC (No. 61732016). Qilin Sun would like to acknowledge the support from NSFC (No. 62302423). Besides, this work was supported by Ant Group, and this paper is supported by the Information Technology Center and State Key Lab of CAD&CGB, Zhejiang University.

REFERENCES


Peter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. 2015. Kinect v2 for mobile robot navigation: Evaluation and modeling. In ICAR.


ACM Trans. Graph., Vol. 42, No. 6, Article 205. Publication date: December 2023.


Yizhong Zhang, Jiasheng Yang, Zhen Liu, Ruicheng Wang, Guojun Chen, Xin Tong, and Baining Guo. 2022b. VirtualCube: An Immersive 3D Video Communication System. In *IEEE VR.*