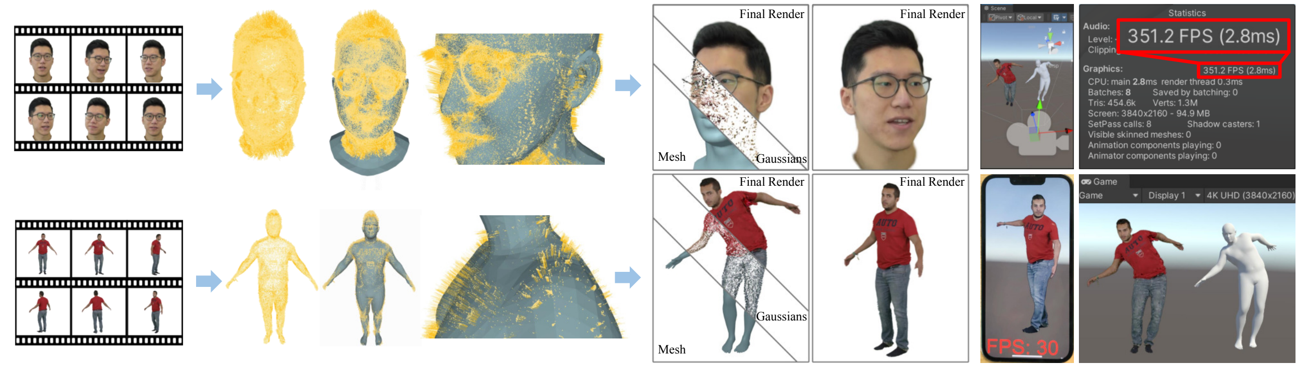
# SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting

Zhijing Shao[1,2] Zhaolong Wang[2] Zhuang Li[2] Duotun Wang[1]
Xiangru Lin[2] Yu Zhang[2] Mingming Fan[1,3] Zeyu Wang[1,3]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Prometheus Vision Technology Co., Ltd.
[3]The Hong Kong University of Science and Technology



```
Graphics:                         351.2 FPS (2.8ms)
  CPU: main 2.8ms  render thread 0.3ms
  Batches: 8        Saved by batching: 0
  Tris: 454.6k      Verts: 1.3M
  Screen: 3840x2160 - 94.9 MB
  SetPass calls: 8            Shadow casters: 1
  Visible skinned meshes: 0
  Animation components playing: 0
  Animator components playing: 0
```

(a) Monocular Video Input (b) Learned Gaussian Splatting Embedding on Mesh (c) Animatable Avatar (d) Real-Time Rendering in Unity

Figure 1. **Overview of SplattingAvatar featuring Mesh-Embedded Gaussian Splatting.** Our method takes (a) monocular videos as input, while employing (b) a trainable embedding technique for Gaussian-Mesh association. (c) Animated by mesh through the learned embedding, the Gaussians render into high-fidelity human avatars. (d) SplattingAvatar demonstrates real-time rendering capabilities in Unity, achieving over 300 FPS on an NVIDIA RTX 3090 GPU and 30 FPS on an iPhone 13 (images captured in action).

## Abstract

*We present SplattingAvatar, a hybrid 3D representation of photorealistic human avatars with Gaussian Splatting embedded on a triangle mesh, which renders over 300 FPS on a modern GPU and 30 FPS on a mobile device. We disentangle the motion and appearance of a virtual human with explicit mesh geometry and implicit appearance modeling with Gaussian Splatting. The Gaussians are defined by barycentric coordinates and displacement on a triangle mesh as Phong surfaces. We extend lifted optimization to simultaneously optimize the parameters of the Gaussians while walking on the triangle mesh. SplattingAvatar is a hybrid representation of virtual humans where the mesh represents low-frequency motion and surface deformation, while the Gaussians take over the high-frequency geometry and detailed appearance. Unlike existing deformation methods that rely on an MLP-based linear blend skinning (LBS) field for motion, we control the rotation and translation of the Gaussians directly by mesh, which empowers its compatibility with various animation techniques, e.g., skeletal animation, blend shapes, and mesh editing. Trainable from monocular videos for both full-body and head avatars, SplattingAvatar shows state-of-the-art rendering quality across multiple datasets. Code and data are available at https://github.com/initialneil/SplattingAvatar.*

## 1. Introduction

The demand for personalized, photorealistic, and animatable human avatars that render in real-time spans a wide array of applications, including gaming [48], extended reality (XR) storytelling [10, 19], and tele-presentation [22, 24]. As the quest for digital realism intensifies, practitioners face a growing challenge: improving the quality of 3D human models often means increasing the complexity of these models. This is typically achieved by adding more polygons, layering skin textures [5], and integrating advanced hair systems [42]. However, these enhancements invariably lead to higher computational demands, creating obstacles in achieving efficiency and portability in avatar rendering.

In our approach, we categorize the representation of mesh-based virtual humans into three distinct levels of detail. The first two levels encompass body motion and surface deformation, both of which are effectively captured by a mesh [11, 16, 39]. The third level, however, focuses on geometric details that are crucial for enhancing realism but challenging to represent with traditional meshes. This level is not only computationally demanding to render [31] but also faces limitations due to the rigid connectivity of mesh vertices, which hinders the adaptability to topological changes and complex or thin structures.

Recent advances in the field have seen a shift towards using Neural Radiance Fields (NeRF) [32], especially for capturing high-frequency details in 3D avatar modeling [3, 18, 21, 26, 27, 34, 35, 53]. A typical process involves constructing NeRF in a canonical space and then performing volume rendering in the posed space. This is done by tracing ray samples backward from their posed positions to their canonical origins [21, 34, 35, 53]. However, this reverse mapping process introduces ambiguities, as a single point in the posed space might correspond to multiple points in the canonical space [8, 9], leading to challenges in accurately rendering details. Additionally, the prevalent use of multilayer perceptron (MLPs) for motion control [21, 39, 51] tends to overlook the advantages of mesh-based representations for capturing surface deformations, an aspect crucial for realistic avatar movement as highlighted in studies like DECA [12], CAPE [30], and TalkSHOW [45].

To address the challenges posed by the limitations of NeRF and MLP-based motion control in capturing high-frequency details and realistic surface deformations, we introduce a novel solution. Inspired by the recently proposed Gaussian Splatting technique [23], we propose explicit motion control of the Gaussians with trainable embeddings on a mesh. The embedding is described by $(k, u, v, d)$ on the mesh as Phong surface [38], where $(u, v)$ represents the local barycentric coordinates of the embedding triangle $k$, and $d$ is the displacement along the interpolated normal vector. The pose-dependent rotation and scaling adjust dynamically in response to the mesh warping, while the pose-invariant properties, i.e., canonical rotation and scaling, color, and opacity, remain stable and consistent across various poses. Because the embedding point defined in barycentric coordinates is differentiable only inside the corresponding triangle, cross-triangle updates must be handled properly [40, 41]. During training, we conduct lifted optimization [38] with the embedding points walking on the triangle mesh.

Our hybrid representation, Gaussians embedded on a mesh, can be trained from a monocular video and efficiently port to Unity that runs in real time (Figure 1) by bringing together three key advantages. First, the use of the mesh for representing body motion and surface deformation not only proves efficient but also allows for high editability. This flexibility is crucial for adapting the avatar to various scenarios and movements. Second, the application of Gaussian Splatting enriches this model by providing a robust means to capture high-frequency geometry and appearance details. This is vital for achieving a level of realism that conventional meshes alone cannot offer. Third, the embedding technique empowers the Gaussians to be explicitly controlled by the mesh movements. This integration results in an efficient, clear, and non-ambiguous method for motion control, significantly reducing the computational load compared to MLP-based methods.

Furthermore, our approach is distinct from existing hybrid models such as AvatarReX [52] and DELTA [14], which typically segment avatars into body parts like hair, hands, clothes, and face. Instead, our method achieves a disentanglement of motion and appearance. In the SplattingAvatar framework, although different parts may have specific motion control, the rendering is uniformly conducted through Gaussian Splatting. This uniformity achieved by our method ensures a cohesive and harmonious appearance across all parts of the avatar.

We summarize our main contributions as follows:

- We introduce a framework that integrates Gaussian Splatting with meshes, offering a new avatar representation that achieves realism and computational efficiency.
- Our approach applies lifted optimization to avatar modeling, allowing for joint optimization of Gaussian parameters and mesh embeddings for accurate reconstruction.
- We demonstrate the capability of real-time rendering and adaptability to creating diverse avatars through comprehensive evaluation and a Unity implementation.

## 2. Related Work

**Mesh-based avatar.** The rise of free-viewpoint video in sequences of textured meshes has shown the expressiveness of detailed texture atlas along with as few as 10k triangles [11]. Many efforts [18, 20] have been put into extending this line of work to build controllable avatars. With the help of human shape models with strong prior [6, 25, 28, 33] that unwrap to a unified UV space, texture atlas can be obtained by 2D image generation supervised through differentiable rendering [31, 43]. Such prior models provide consistency across large motions and can be recovered from monocular videos or even a single image. To cope with the shape details of identities and clothes, CAPE [30] predicts displacements on the vertices with pose-conditioned VAE. Due to the limitation of the base model to topological changes, some treat the textured mesh as input conditions [31, 36] for image rendering, while others resort to implicit representations of

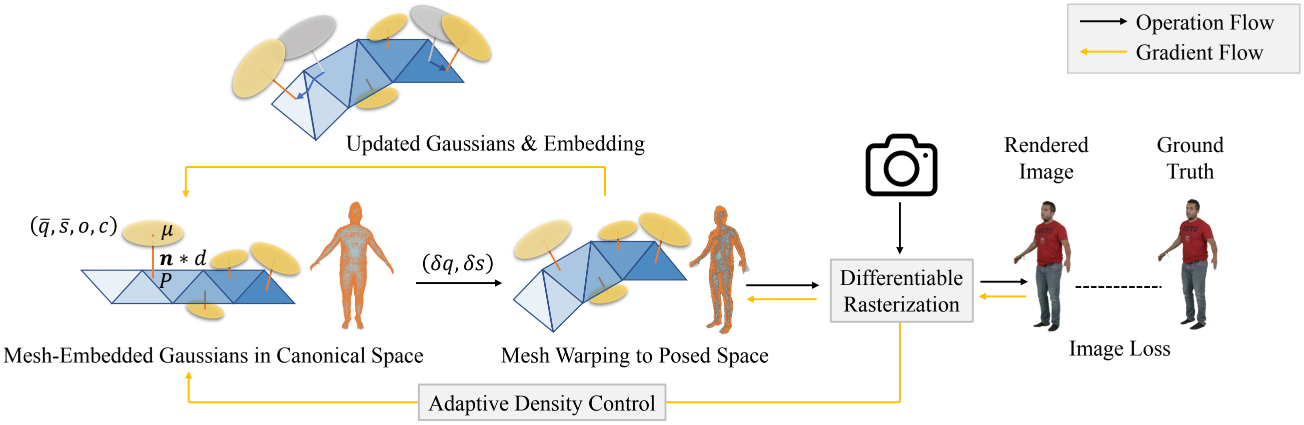

Operation Flow
Gradient Flow
Updated Gaussians & Embedding
$(\bar{q}, \bar{s}, o, c)$
$\mu$
$\boldsymbol{n} * d$
$P$
$(\delta q, \delta s)$
Rendered Image
Ground Truth
Differentiable Rasterization
Image Loss
Mesh-Embedded Gaussians in Canonical Space
Mesh Warping to Posed Space
Adaptive Density Control

Figure 2. **The pipeline of our method.** SplattingAvatar learns 3D Gaussians with trainable embedding on the canonical mesh. The motion and deformation of the mesh explicitly bring the Gaussians to the posed space for differentiable rasterization. Both the Gaussians and embedding parameters are optimized during training. The position $\mu$ is the barycentric point $P$ plus a displacement $d$ along the interpolated normal vector $\boldsymbol{n}$. Pose-dependent quaternion and scaling $(\delta q, \delta s)$ and pose-invariant quaternion, scaling, opacity, and color $(\overline{q}, \overline{s}, o, c)$ together define the properties of the Gaussians.

the mesh [8, 9, 20, 39], color [16, 20, 39], or materials [4].

**Implicit neural avatar.** To achieve convincing rendering beyond the limitation of triangle mesh, especially on the hair, glasses, and clothes, some recent works [3, 16, 21, 26, 34, 35, 44, 47, 53] focus on constructing NeRF in the canonical space (usually T-pose of SMPL [28] or neutral expression of FLAME [25]) and conduct volume rendering at the posed space. The required backward tracing from pose to canonical is non-trivial and raises an ambiguity issue. Existing works propose to adopt pose conditioned inverse LBS field [17, 34] or to optimize a root-finding loop with multiple initialization [8, 9, 21]. The increased computational load upon volume rendering prohibits the potential real-time applications.

PointAvatar [51], with explicit point primitives, takes advantage of forward rasterization that only requires non-ambiguous forward deformation from canonical to pose, producing photo-realistic appearance and detailed challenging geometries such as hair and glasses. In transforming to Gaussian Splatting, we further increase the efficiency and compatibility with our mesh embedding mechanism instead of the LBS-based deformation field and achieve two magnitude faster rendering speed with on-par quality.

**Hybrid avatar representation.** First attempts have been proposed to disentangle human avatar modeling into separate parts with varying properties. AvatarRex [52] learns disentangled models for face, body, and hands. SCARF [13] and DELTA [14] propose hybrid modeling with textured mesh for body, and NeRF for hair and clothing. In contrast, our method handles the disentanglement in terms of motion and appearance to explicit mesh geometry and implicit Gaussian Splatting rendering. Different from existing works [3, 20] that attach features to fixed locations on mesh like mesh vertices, our trainable embedding enables the Gaussians to optimize their locations on mesh and distribute unevenly according to the texture complexity.

## 3. Method

**Overview.** Given a sequence of monocular images, each with a registered mesh template, i.e., the deformed mesh of SMPL [28] or FLAME [25], we train a hybrid representation of human avatar as 3D Gaussians [23] embedded on the canonical mesh. The Gaussians, parameterized by position, rotation, scale, color, and opacity, are semi-transparent 3D particles that render into camera views through splatting-based rasterization.

Each 3D Gaussian is embedded on one triangle of the canonical mesh in its local $(u, v, d)$ coordinates. The embedding directly defines the position of the Gaussians in both canonical and posed space. Other than position, each Gaussian has its own parameters of rotation, scaling, color, and opacity. With the mesh deformed by animation, the embedding also provides additional rotation and scaling upon each Gaussian. The additional pose-dependent rotation is defined by barycentric interpolated per-vertex quaternion while the additional scaling is defined by the area change of the embedded triangle.

During optimization, the Gaussian parameters and the embedding parameters are updated simultaneously. When the update of $(u, v)$ moves the embedding across the triangle boundary, the barycentric update is re-expressed in the neighboring triangle as if the Gaussian is walking on the mesh. To support embedding, we adapt the clone and split scheme of 3D Gaussians [23] to better suit our needs.

**Embedding on mesh.** Inspired by the Phong shading in computer graphics, Phong surface [38] defines the position and normal of a point inside a triangle. For the point $P$ on triangle $k$ with barycentric coordinate $(u, v)$, its position and normal is a linear interpolation of the triangle's vertices $\{V_1, V_2, V_3\}$ and per-vertex normals $\{\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3\}$:

$$P = \mathcal{V}(k, u, v) = u * V_1 + v * V_2 + (1 - u - v) * V_3 \quad (1)$$

$$\boldsymbol{n} = \mathcal{N}(k, u, v) = u * \boldsymbol{n}_1 + v * \boldsymbol{n}_2 + (1 - u - v) * \boldsymbol{n}_3 \quad (2)$$

where $\mathcal{V}$ maps triangle index $k$ and barycentric coordinates $(u, v)$ to a point on the mesh and $\mathcal{N}$ the interpolated normal.

We define the position of a Gaussian, i.e., the mean $\mu$, by a displacement $d$ along the interpolated normal vector:

$$\mu = P + d * \boldsymbol{n} \quad (3)$$

Embedding $E = \{k, u, v, d\}$ approximates a first-order continuous space around the mesh surface.

As proposed by Zielonka et al. [53], for the corresponding triangle in the canonical and posed space at frame $t$ we compute the matrix $\{R_{cano}, R_{pose}\}$ based on the triangle's tangent, bitangent, and normal to track the triangle rotation from canonical to pose, noted that the notation $t$ is skipped. The rotation matrix is then converted to a quaternion, and we calculate the per-vertex quaternion $q_V$ by area-weighted average from surrounding neighbor triangles:

$$R_k = R_{cano} R_{pose}^{-1} \quad (4)$$

$$q_V = \frac{\sum_{k \in \Omega_V} A_k q_k}{\sum_{k \in \Omega_V} A_k} \quad (5)$$

where $\Omega_V$ is the neighbor triangles of vertex $V$, $A_k$ and $q_k$ are the triangle's area and quaternion respectively. For an embedding $E_i$ with quaternions $\{q_1, q_2, q_3\}$ calculated on the corresponding triangle vertices at frame $t$, the barycentric interpolated rotation $\delta q_{i,t}$ is multiplied to the canonical rotation $\overline{\boldsymbol{q}}_i$ of the Gaussian in the canonical space:

$$\delta q_{i,t} = u * q_1 + v * q_2 + (1 - u - v) * q_3 \quad (6)$$

$$q_{i,t} = \delta q_{i,t} * \overline{q}_i \quad (7)$$

The same applies to scaling where the area change of the embedded triangle is used to represent the scaling caused by deformation: $s_{i,t} = (A_{pose}/A_{cano})\overline{s}_i$. While the original implementation of Gaussian Splatting [23] represents color in view-dependent spherical harmonics, we choose to turn it off for simplicity [29].

We perform initialization by randomly selecting 10k pairs of triangle indices and barycentric coordinates on the canonical mesh. We set the barycentric coordinates to be the current $(u, v)$ of embeddings and initialize all the $d$ to be zero. With the position of the Gaussians calculated from the embeddings, we initialize other properties of the Gaussians according to their original definitions [23]. Initially, the Gaussians are positioned on the surface of the mesh. With the training proceeds with more poses, the embeddings generally bring the Gaussians to approximate the actual geometry and densify in the regions with rich texture. Figure 3 illustrates the development of the embeddings.

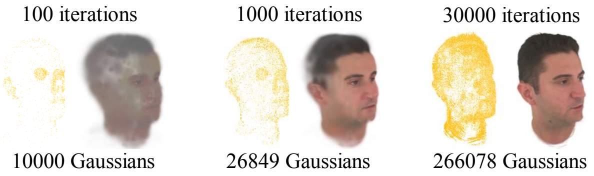


Figure 3. **The development of Gaussian embeddings on mesh.** Each line segment indicates the position of one Gaussian displaced from its embedding point on mesh. Gaussians for off-surface geometries like the hair have positive displacements while others turn to have negative displacements because when the mesh surface is correctly aligned to the geometry like in the facial area, the means for the Gaussians will be inside the mesh.

**Differentiable rendering of Gaussian Splatting.** With the position, rotation, and scaling of the Gaussians updated by the mesh deformation at frame $t$, we perform differentiable Gaussian rendering [23] to the observed camera view(s). The Gaussian in space is defined by its mean $\mu$ and a 3D covariance matrix $\Sigma$.

$$G_{i,t}(x) = e^{-\frac{1}{2}(x)^T \Sigma_{i,t}^{-1}(x)} \quad (8)$$

$$\Sigma_{i,t} = R_{i,t} S_{i,t} S_{i,t}^T R_{i,t}^T \quad (9)$$

where $R_{i,t}$ is the rotation matrix constructed from $q_{i,t}$, and $S_{i,t}$ the scaling matrix from $s_{i,t}$. Given the world-to-camera view matrix $W$ and the Jacobian $J$ of the point projection matrix. The influence of the Gaussian is splatted to 2D [54]:

$$\Sigma' = J W \Sigma W^T J^T \quad (10)$$

The image formation of Gaussian Splatting is akin to NeRF, where the same volume rendering formula is applied to the blending from near to far. The color $C$ of a pixel rendered by $N$ Gaussians is given by a series of $\alpha$-*blending*:

$$C = \sum_{i=1}^{N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (11)$$

with $\alpha_i$ evaluated from the 2D covariance, and an opacity in logit $o_i$ with sigm() being the standard sigmoid function:

$$\alpha_i(P) = \text{sigm}(o_i) \exp(-\frac{1}{2}(P - \mu_i)(\Sigma_i)^{-1}(P - \mu_i)) \quad (12)$$

The Equation 11 is implemented in CUDA with a for loop for each pixel, while in our Unity implementation, each Gaussian is drawn by a front-parallel Quad primitive based on the projection and 2D covariance. We resort to the standard rasterization pipeline of the rendering engine to enable $\alpha$-*blending* with these semi-transparent Gaussians.

Due to limited viewing angle and pose variations from monocular video, we propose a scaling regularization term to prevent Gaussians from growing long and thin. A random background color is generated every iteration to mix with the rendered image $I$ and ground truth image $I_{gt}$, providing important cues for the silhouette. The loss is the sum of $\mathcal{L}_1$, $\mathcal{L}_{mse}$, a perceptual loss [49], and a scaling regularization.

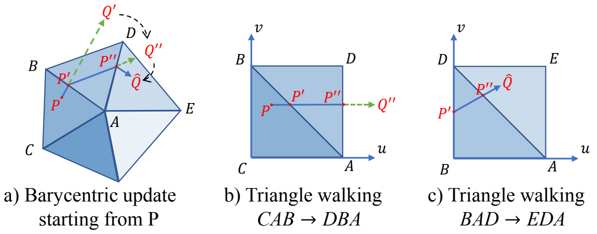
$$\mathcal{L} = \mathcal{L}_1 + \lambda_{l_{mse}}\mathcal{L}_{mse} + \lambda_l\mathcal{L}_{lpips} + \lambda_s\mathcal{L}_{scaling} \tag{13}$$

$$\mathcal{L}_{scaling}(i) = \begin{cases} |\hat{s_i}|, & \hat{s_i} > \max(T_s, T_r\check{s_i}) \\ 0, & otherwise \end{cases} \tag{14}$$

With $s_i \in \mathbb{R}^3$ being the scaling of a Gaussian, $\hat{s_i}$ and $\check{s_i}$ are the maximum and minimum scaling values respectively. The scaling regularization is posed on $\hat{s_i}$ when it is both long (larger than $T_s$) and thin (larger than $T_r$ times $\check{s_i}$). Please see Section 4.3 for an ablation study on the regularization term.

**Walking on a triangle mesh.** The notion *Lifted Optimization* arises in the model-point registration for hand tracking [38, 40, 41] in contrast to *Iterative Closest Point (ICP)*, where the solve for model pose and correspondences are *lifted* to be *simultaneous*. We extend this notion to our avatar training, where the properties of the Gaussians and the trainable embeddings are optimized simultaneously. The barycentric coordinate of a point $P$ is $(k, u, v)$ defined within triangle $k$. When the learned update $Q = (k, u, v) + (\delta u, \delta v)$ is outside triangle $k$, we find the intersection $P'$ on the shared edge of the adjacent triangle $k'$ and re-express the remaining update in $k'$ as $Q' = P' + (\delta u', \delta v')$. Because the barycentric coordinates are agnostic to the triangle shape, without loss of generality, the re-expression is conducted by conceptually treating two adjacent triangles as right triangles with the intersection on the hypotenuse. The update is iteratively re-expressed until it ends inside the final triangle. We show the re-expression process in Figure 4. The detailed steps are presented in Algorithm 1. Noted that we omit the conceptual re-ordering of the vertices.

**Optimization.** We use Adam to optimize the Gaussian parameters and the embedding parameters. The original learning rate attenuation on position [23] is instead applied to the embedding parameters. We record the current barycentric $(u, v)$ and optimize for $(\delta u, \delta v, d)$. The triangle walking in Algorithm 1 is implemented as a *pybind11* module in C++. When an embedding is being transferred to another triangle, we reset its corresponding optimizer state of the $(\delta u, \delta v, d)$.



a) Barycentric update starting from P
b) Triangle walking *CAB* → *DBA*
c) Triangle walking *BAD* → *EDA*

Figure 4. **Walking on triangles for embedding update.** a) The recursion process of walking on a triangle mesh. b) The update $P + \delta$ starting from triangle *CAB* is re-expressed as $P' + \delta'$ in triangle *DBA*, and c) re-expressed again in *EDA*. The re-expression between two triangles is conducted by conceptually treating them as two right triangles adjacent to each other on the hypotenuse.

**Algorithm 1** Walking on triangles

```
Input: $k, u, v, \delta u, \delta v$
Output: $\hat{k}, \hat{u}, \hat{v}$
  function WalkOnTriangles($k, u, v, \delta u, \delta v$)
      $P \leftarrow (u, v)$
      $Q \leftarrow (u + \delta u, v + \delta v)$
      if Q is inside triangle then
          Return $(k, Q.u, Q.v)$
      end if
      Intersect $P$-$Q$ with hypotenuse* on $(u', v')$
                                      ▷ *reorder vertices if needed
      $\delta u' \leftarrow \delta u - (u' - u)$
      $\delta v' \leftarrow \delta v - (v' - v)$
      Return ReExpress($k, u', v', \delta u', \delta v'$)
  end function
  function ReExpress($k, u', v', \delta u', \delta v'$)
      $\hat{k} \leftarrow$ adjacent of $k$
      $\hat{u} \leftarrow 1 - u'$, $\hat{v} \leftarrow 1 - v'$
      $\delta\hat{u} \leftarrow -\delta u'$, $\delta\hat{v} \leftarrow -\delta v'$
      Return WalkOnTriangles($\hat{k}, \hat{u}, \hat{v}, \delta\hat{u}, \delta\hat{v}$)
  end function
$(\hat{k}, \hat{u}, \hat{v}) \leftarrow$ WalkOnTriangles($k, u, v, \delta u, \delta v$)
```

The densification process [23] plays an important role in allocating more Gaussians to where in need. In the clone and prune process, the embedding parameters are copied or deleted in the same way as Gaussian parameters. In the split process, when a new position $\hat{\mu}$ is sampled from the Gaussian, we solve a mini problem with triangle walking to find the new embedding:

$$\hat{E} = \underset{k,u,v,d}{\arg\min} \|\mathcal{V}(k, u, v) + d * \mathcal{N}(k, u, v) - \hat{\mu}\|_2^2 \tag{15}$$

**Unity implementation for mobile device.** With maximum compatibility in mind, we made SplattingAvatar solely rely on the warped mesh. Before exporting to Unity, we uploaded the canonical mesh in *.obj* format to Mixamo [1] for auto-rigging. In total, we exported one *.ply* file of the Gaussians, one *.json* file describing the embedding, and one *.fbx* file from Mixamo to Unity. Note that the *.fbx* file can be rigged

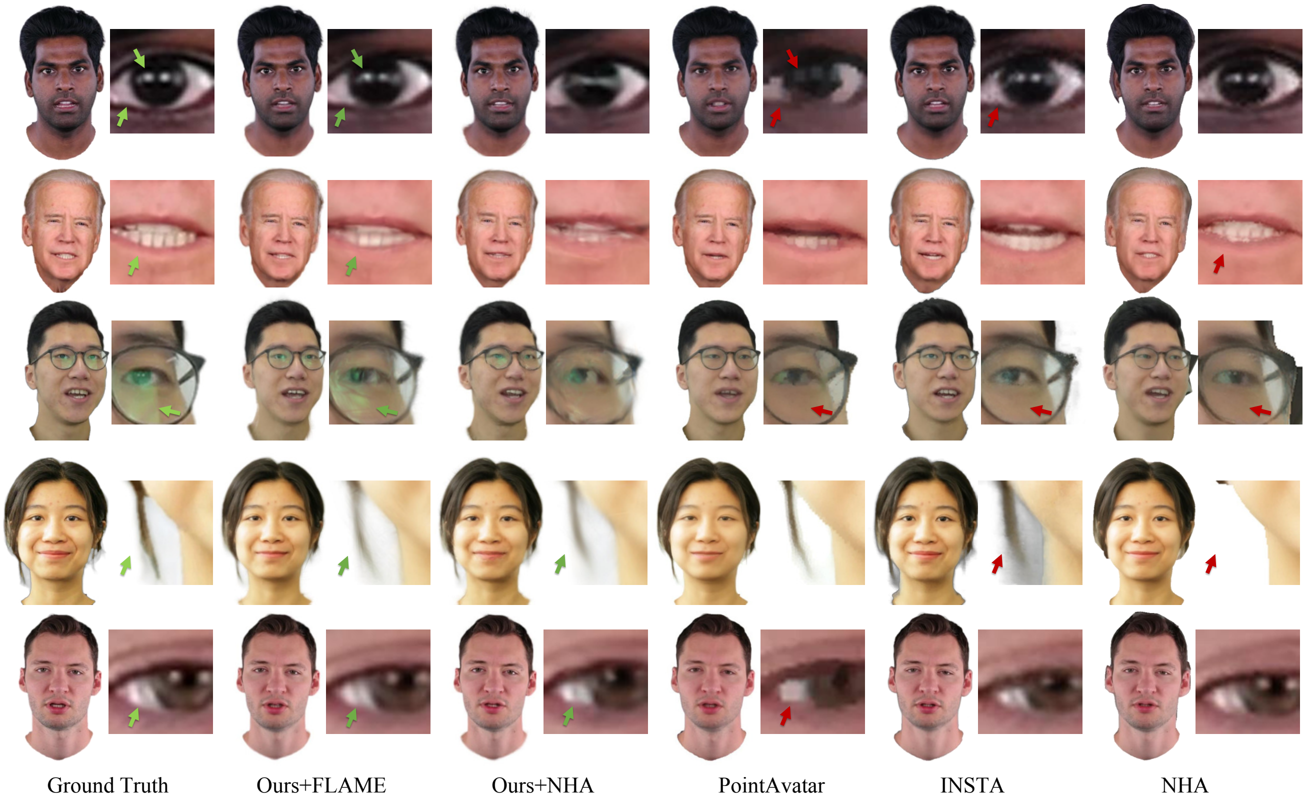


Figure 5. **Qualitative comparison on head avatars.** SplattingAvatar produces photorealistic rendering for avatars with high-quality details especially in the eye and hair regions. Even the light reflection on the glasses is well reconstructed. Both PointAvatar [51] and NHA [16] can reconstruct good geometries but the rendering quality is limited by their underlying representations, i.e., points and texture atlas respectively. Compared to INSTA [53], our trainable embedding scheme produces better quality for off-surface geometries, especially for the glasses. The green arrows highlight where our results have better consistency with Ground Truth, while the red arrows point to where other methods show significant artifacts or noise. Please see the supplemental materials for illustrations of the error map.

by any other software for customized needs, as long as the triangle order is maintained.

We implemented the Gaussian renderer in Unity's compute shaders, starting from sorting all Gaussians by the z-axis in camera coordinates from near to far. Based on the calculated 2D covariance $\Sigma'$, one front-parallel quad primitive is drawn for every visible Gaussian centered at its position. This one-primitive-one-Gaussian strategy is important for the game engine to properly handle the occlusion of other regular objects. For every pixel to draw in the fragment shader, our implementation emits color with alpha pre-multiplication and sets the blend function to *(ONE, ONE_MINUS_SRC_ALPHA)*. Our Unity program achieves a high performance of over 300 FPS on a modern GPU while maintaining a steady 30 FPS on an iPhone 13.

## 4. Experiments

To demonstrate the effectiveness of SplattingAvatar, we compared it with state-of-the-art (SoTA) methods in two different types of datasets for head and full-body avatars.

| Method | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| NHA [16] | 20.29 | 0.883 | 0.145 |
| INSTA [53] | 26.42 | 0.924 | 0.080 |
| PointAvatar [51] | 27.84 | 0.913 | 0.067 |
| Ours+FLAME | <u>28.19</u> | **0.931** | <u>0.063</u> |
| Ours+NHA | **28.86** | **0.931** | **0.060** |

Table 1. **Quantitative comparison on head avatars.** Both variations of our method outperform existing methods in terms of average photometric errors. With detailed meshes from NHA [16], *Ours+NHA* performs the best based on the metrics. However, we observe better visual quality with *Ours+FLAME* in the inner regions of the rendered image.

### 4.1. Datasets

**Monocular video for head avatar.** Taking a single monocular video to construct a head avatar for the given subject, our method takes as input images, masks, camera parameters, and tracked FLAME meshes, denoting ***Ours+FLAME***. We

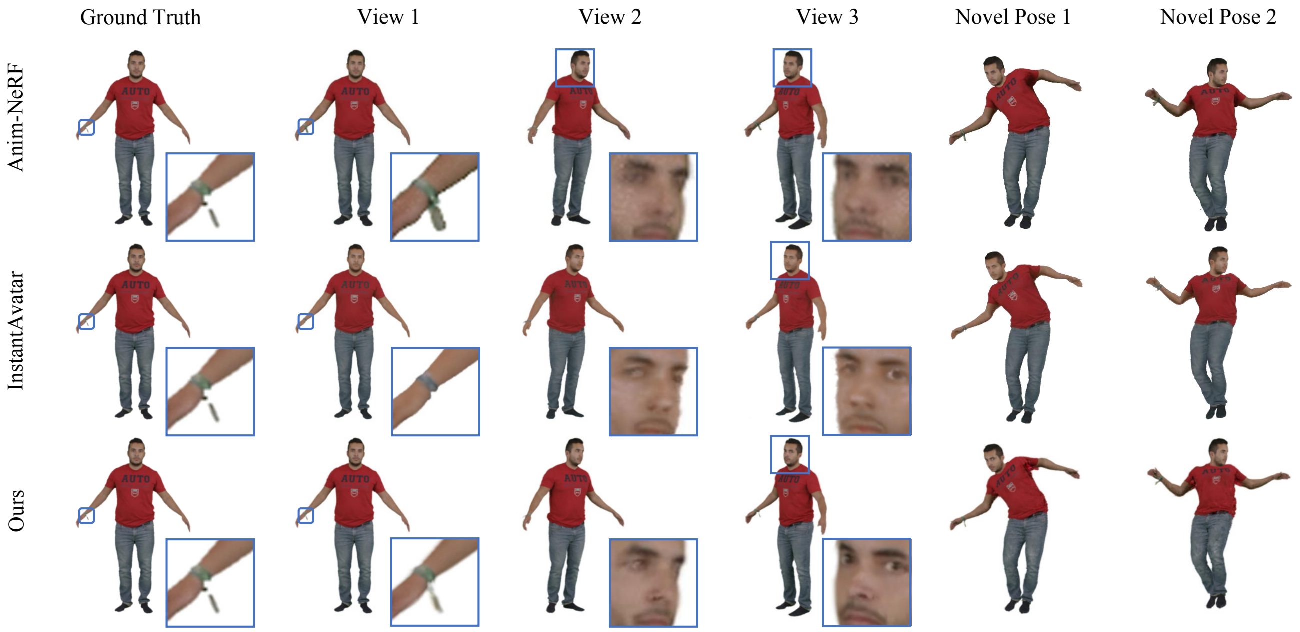


Figure 6. **Qualitative comparison on PeopleSnapshot [2].** We show the results on PeopleSnapshot (columns 2–4) and novel pose animation (columns 5–6). SplattingAvatar produces photorealistic rendering for full-body avatars, especially in the facial area, and captures thin structures like the accessory on the wrist.

| | male-3-casual | | | male-4-casual | | | female-3-casual | | | female-4-casual | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| Anim-NeRF [7] | 29.37 | 0.970 | **0.017** | 28.37 | 0.960 | 0.027 | 28.91 | 0.974 | **0.022** | 28.90 | 0.968 | **0.017** |
| InstantAvatar [21] | 30.91 | 0.977 | 0.022 | 29.77 | 0.980 | **0.025** | 29.73 | 0.975 | 0.025 | 30.92 | 0.977 | 0.021 |
| Ours | **33.01** | **0.982** | 0.020 | **30.99** | **0.982** | 0.029 | **30.81** | **0.978** | 0.028 | **32.57** | **0.981** | 0.018 |

Table 2. **Quantitative comparison on PeopleSnapshot.** Compared to two SoTA methods, we achieve significant improvements in pixel-wise quality with PSNR and SSIM. All three methods achieve good perceptual quality in terms of LPIPS where the metrics are close.

evaluated our approach with several SoTA methods on a combined dataset from NHA [16], NerFace [15], INSTA [53] and PointAvatar [51], including 10 subjects covering different videos captured with DSLR, smartphones and from the Internet. The pre-processing pipeline of IMavatar [50] and INSTA [53] was altered to apply DECA [12] for face tracking, RVM [37] for segmentation, and BisenetV2 [46] for face parsing. For each video, the last 350 frames were used as testing samples. Because our method can directly be animated by the given mesh, we further unleashed its potential by training and testing on the generated meshes from NHA [16] which have more geometry details. This variation is referred to as *Ours+NHA*.

**PeopleSnapshot.** We conducted a quantitative evaluation of the rendering quality of full-body avatars on the PeopleSnapshot [2] dataset. Following the protocol of InstantAvatar [21], we used SMPL meshes refined by Anim-NeRF [7]. Our method demonstrates the generalizability to novel poses through qualitative analysis in Section 4.2.

## 4.2. Comparison with SoTA

**Head avatar.** To evaluate the rendering quality of the learned avatars, we animated SplattingAvatar with the registered meshes of testing images. For *Ours+NHA*, we trained NHA [16] on the training set and extracted the final meshes for both the training and testing images, which were further used for the training and testing of our method respectively.

We conducted a comparative analysis of SplattingAvatar against INSTA [53], PointAvatar [51], and NHA [16]. As depicted in Figure 5, our method achieves superior quality in terms of improved details in the eye and hair regions, and even being able to capture the light reflection on the glasses. For *Ours+FLAME*, though the off-surface geometries like hair and glasses are not fully represented by meshes, our method can handle the rendering decently because the embeddings are optimized to find correct motions from nearby triangles. Please see Table 1 for quantitative evaluations.

**Full-body avatar.** We made a comparison to InstantAvatar [21] and Anim-NeRF [7] on PeopleSnapshot. For InstantAvatar [21], a complete training was performed for 200 epochs as suggested in the most recent version of the author's code. Image quality metrics in Table 2 demonstrate the effectiveness of our method in terms of the lowest pixel-wise errors. We qualitatively show the comparison of rendering quality of testing images in Figure 6, together with the demonstration of generalizability to novel poses. Our representation is friendly to thin structures like the accessory on the wrist. Our approach produced better quality overall and especially in the facial area compared to InstantAvatar [21], but slightly more artifacts under the shoulder due to very limited pose variations in the training set. We believe this can be much improved with more training poses.

### 4.3. Ablation Study

**Trainable embedding.** The key component of our method is the trainable embedding on the mesh. We conducted an ablation experiment by replacing it with fixed embedding on mesh and a trainable local shift $\Delta x \in \mathbb{R}^3$ per Gaussian. Without trainable embedding, the Gaussians encountered difficulties in following the mesh correctly. The right column of Figure 7 shows the irregular rendering artifacts without trainable embedding.

**Regularization.** In the optimization process of Gaussian Splatting, some Gaussians turn to become long and thin, generating artifacts when rendered into novel poses. We show the results without the scaling regularization in the middle column of Figure 7.

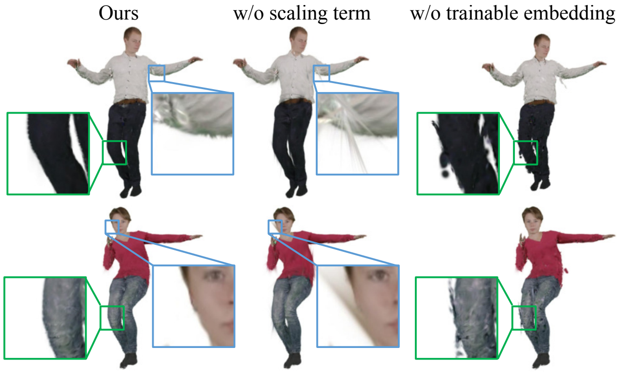


Figure 7. **Ablation study.** Without the scaling regularization term, Gaussians that are long and thin cause needle-like artifacts. Without trainable embedding, Gaussians do not follow the movement of the mesh tightly, leading to irregular rendering results. The application of our trainable embedding and the scaling term successfully removes most of the artifacts when rendered into novel poses.

### 4.4. Discussion

**Discussion on driving mesh.** Considering efficiency, compatibility, and portability, SplattingAvatar is designed to tightly rely on the motion and surface deformation of the underlying mesh. In the comparison between *Ours+FLAME* and *Ours+NHA*, we observe that the driving mesh should focus on the motion instead of fully reconstructing the exact geometry. In Figure 8 we show that when the mesh with vertex offsets from NHA [16] is applied, the detailed surface deformation improves the generalizability of SplattingAvatar to large poses. However, in the second and third row of Figure 5, the mesh from FLAME that captures the correct motion of the glasses rather than the shape is driving the best rendering quality of SplattingAvatar. To perform textured mesh rendering, the mesh of NHA [16] is seamed in the mouth region and deformed to fit the shape of the glasses, yet both being unhelpful to the quality of *Ours+NHA*.

**Limitations and future work.** As discussed above, our method depends on the motion representation ability of the driving mesh. With current FLAME and SMPL models, we do not have separate motion representations for clothes and hair. We believe SplattingAvatar can support future works on human avatars with disentangled mesh representations, e.g., separate meshes for clothes and hair stands.

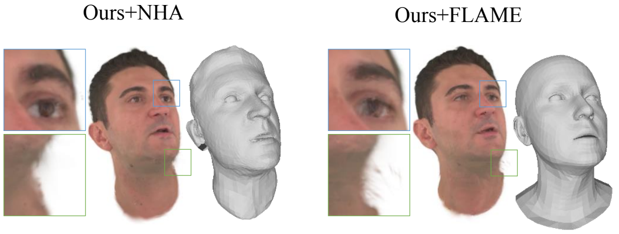


Figure 8. **Comparison between *Ours+FLAME* and *Ours+NHA*.** The better aligned mesh from NHA [16] improves the generalizability of SplattingAvatar to large pose variations.

## 5. Conclusion

In this paper, we have proposed a hybrid representation for human avatar modeling featuring Gaussian Splatting with trainable embeddings on a mesh. We extend lifted optimization to simultaneously optimize the parameters of the Gaussians and their embeddings. Our method leverages the advantages of the explicit motion representation with a mesh and implicit rendering capability of Gaussian Splatting. Compared with SoTA methods, our approach achieves the best rendering quality for both head and full-body avatars reconstructed from monocular videos and runs at real-time frame rates on a mobile device. Our method lays a foundation for future work in Gaussian Splatting manipulation with mesh-based motion control.

## Acknowledgments

This work was partially supported by Guangzhou-HKUST(GZ) Joint Funding #2023A03J0670 and Guangzhou Basic Research Scheme #2024A04J4229.

## References

[1] Mixamo. https://www.mixamo.com/. Accessed: November 10, 2023. 5

[2] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Detailed Human Avatars from Monocular Video. In *International Conference on 3D Vision*, pages 98–109, 2018. 7

[3] Ziqian Bai, Feitong Tan, Zeng Huang, Kripasindhu Sarkar, Danhang Tang, Di Qiu, Abhimitra Meka, Ruofei Du, Mingsong Dou, Sergio Orts-Escolano, Rohit Pandey, Ping Tan, Thabo Beeler, Sean Fanello, and Yinda Zhang. Learning Personalized High Quality Volumetric Head Avatars From Monocular RGB Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16890–16900, 2023. 2, 3

[4] Shrisha Bharadwaj, Yufeng Zheng, Otmar Hilliges, Michael J. Black, and Victoria Fernandez Abrevaya. FLARE: Fast learning of animatable and relightable mesh avatars. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, page 15, 2023. 3

[5] George Borshukov and J. P. Lewis. Realistic Human Face Rendering for "The Matrix Reloaded". In *ACM SIGGRAPH 2005 Courses*, page 13–es, New York, NY, USA, 2005. Association for Computing Machinery. 1

[6] Zenghao Chai, Haoxian Zhang, Jing Ren, Di Kang, Zhengzhuo Xu, Xuefei Zhe, Chun Yuan, and Linchao Bao. REALY: Rethinking the Evaluation of 3D Face Reconstruction. In *European Conference on Computer Vision*, pages 74–92. Springer, 2022. 2

[7] Jianchuan Chen, Ying Zhang, Di Kang, Xuefei Zhe, Linchao Bao, Xu Jia, and Huchuan Lu. Animatable Neural Radiance Fields from Monocular RGB Videos, 2021. 7

[8] Xu Chen, Yufeng Zheng, Michael J. Black, Otmar Hilliges, and Andreas Geiger. SNARF: Differentiable Forward Skinning for Animating Non-Rigid Neural Implicit Shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11594–11604, 2021. 2, 3

[9] Xu Chen, Tianjian Jiang, Jie Song, Max Rietmann, Andreas Geiger, Michael J. Black, and Otmar Hilliges. Fast-SNARF: A Fast Deformer for Articulated Neural Fields. *Pattern Analysis and Machine Intelligence (PAMI)*, 2023. 2, 3

[10] Kun-Hung Cheng and Chin-Chung Tsai. Children and Parents' Reading of An Augmented Reality Picture Book: Analyses of Behavioral Patterns and Cognitive Attainment. *Computers & Education*, 72:302–312, 2014. 1

[11] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4), 2015. 2

[12] Yao Feng, Haiwen Feng, Michael J Black, and Timo Bolkart. Learning an Animatable Detailed 3D Face Model from In-The-Wild Images. *ACM Transactions on Graphics (ToG)*, 40 (4):1–13, 2021. 2, 7

[13] Yao Feng, Jinlong Yang, Marc Pollefeys, Michael J. Black, and Timo Bolkart. Capturing and Animation of Body and Clothing from Monocular Video. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3

[14] Yao Feng, Weiyang Liu, Timo Bolkart, Jinlong Yang, Marc Pollefeys, and Michael J. Black. Learning Disentangled Avatars with Hybrid 3D Representations. *arXiv*, 2023. 2, 3

[15] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, 2021. 7

[16] Philip-William Grassal, Malte Prinzler, Titus Leistner, Carsten Rother, Matthias Nießner, and Justus Thies. Neural Head Avatars From Monocular RGB Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18653–18664, 2022. 2, 3, 6, 7, 8

[17] Chen Guo, Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. Vid2Avatar: 3D Avatar Reconstruction from Videos in the Wild via Self-supervised Scene Decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[18] Marc Habermann, Lingjie Liu, Weipeng Xu, Gerard Pons-Moll, Michael Zollhoefer, and Christian Theobalt. HDHumans: A Hybrid Approach for High-Fidelity Digital Humans. *Proc. ACM Comput. Graph. Interact. Tech.*, 6(3), 2023. 2

[19] Jennifer Healey, Duotun Wang, Curtis Wigington, Tong Sun, and Huaishu Peng. A Mixed-Reality System to Promote Child Engagement in Remote Intergenerational Storytelling. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 274–279, 2021. 1

[20] Hsuan-I Ho, Lixin Xue, Jie Song, and Otmar Hilliges. Learning Locally Editable Virtual Humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21024–21035, 2023. 2, 3

[21] Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. InstantAvatar: Learning Avatars From Monocular Video in 60 Seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16922–16932, 2023. 2, 3, 7, 8

[22] Redouane Kachach, Pablo Perez, Alvaro Villegas, and Ester Gonzalez-Sosa. Virtual Tour: An Immersive Low Cost Telepresence System. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 504–506, 2020. 1

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 3, 4, 5

[24] Nianlong Li, Zhengquan Zhang, Can Liu, Zengyao Yang, Yinan Fu, Feng Tian, Teng Han, and Mingming Fan. VMirror: Enhancing the Interaction with Occluded or Distant Objects in VR with Virtual Mirrors. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021. Association for Computing Machinery. 1

[25] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a Model of Facial Shape and Expression from 4D Scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. 2, 3

[26] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural Actor: Neural Free-View Synthesis of Human Actors with Pose Control. *ACM Trans. Graph.*, 40(6), 2021. 2, 3

[27] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of Volumetric Primitives for Efficient Neural Rendering. *ACM Trans. Graph.*, 40(4), 2021. 2

[28] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.*, 34(6), 2015. 2, 3

[29] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. *arXiv*, 2023. 4

[30] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J. Black. Learning to Dress 3D People in Generative Clothing. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[31] Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De la Torre, and Yaser Sheikh. Pixel Codec Avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 64–73, 2021. 2

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 405–421, Berlin, Heidelberg, 2020. Springer-Verlag. 2

[33] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[34] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable Neural Radiance Fields for Modeling Dynamic Human Bodies. In *ICCV*, 2021. 2, 3

[35] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans. In *CVPR*, 2021. 2, 3

[36] Sergey Prokudin, Michael J Black, and Javier Romero. SMPLpix: Neural Avatars from 3D Human Models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1810–1819, 2021. 2

[37] Shanchuan Lin and Linjie Yang and Imran Saleemi and Soumyadip Sengupta. Robust high-resolution video matting with temporal guidance. *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3132–3141, 2021. 7

[38] Jingjing Shen, Thomas J. Cashman, Qi Ye, Tim Hutton, Toby Sharp, Federica Bogo, Andrew Fitzgibbon, and Jamie Shotton. The Phong Surface: Efficient 3D Model Fitting Using Lifted Optimization. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 687–703, Berlin, Heidelberg, 2020. Springer-Verlag. 2, 4, 5

[39] Kaiyue Shen, Chen Guo, Manuel Kaufmann, Juan Jose Zarate, Julien Valentin, Jie Song, and Otmar Hilliges. X-Avatar: Expressive Human Avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16911–16921, 2023. 2, 3

[40] Jonathan Taylor, Richard Stebbing, Varun Ramakrishna, Cem Keskin, Jamie Shotton, Shahram Izadi, Aaron Hertzmann, and Andrew Fitzgibbon. User-Specific Hand Modeling from Monocular Depth Sequences. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 644–651, 2014. 2, 5

[41] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, and Jamie Shotton. Efficient and Precise Interactive Hand Tracking through Joint, Continuous Optimization of Pose and Correspondences. *ACM Trans. Graph.*, 35(4), 2016. 2, 5

[42] Keyu Wu, Yifan Ye, Lingchen Yang, Hongbo Fu, Kun Zhou, and Youyi Zheng. Neuralhdhair: Automatic High-fidelity Hair Modeling from a Single Image Using Implicit Neural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1526–1535, 2022. 1

[43] Donglai Xiang, Fabian Prada, Timur Bagautdinov, Weipeng Xu, Yuan Dong, He Wen, Jessica Hodgins, and Chenglei Wu. Modeling Clothing as a Separate Layer for an Animatable Human Avatar. *ACM Trans. Graph.*, 40(6), 2021. 2

[44] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-NeRF: Neural Radiance Fields for Rendering and Temporal Reconstruction of Humans in Motion. In *Advances in Neural Information Processing Systems*, pages 14955–14966. Curran Associates, Inc., 2021. 3

[45] Hongwei Yi, Hualin Liang, Yifei Liu, Qiong Cao, Yandong Wen, Timo Bolkart, Dacheng Tao, and Michael J Black. Generating holistic 3d human motion from speech. In *CVPR*, 2023. 2

[46] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. BiSeNet V2: Bilateral Network with Guided Aggregation for Real-Time Semantic Segmentation. *Int. J. Comput. Vision*, 129(11):3051–3068, 2021. 7

[47] Zhengming Yu, Wei Cheng, Xian Liu, Wayne Wu, and Kwan-Yee Lin. MonoHuman: Animatable Human Neural Field from Monocular Video. *CVPR*, 2023. 3

[48] Peter Zackariasson and Timothy L Wilson. *The Video Game Industry: Formation, Present State, and Future*. Routledge, 2012. 1

[49] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018. 5

[50] Yufeng Zheng, Victoria Fernández Abrevaya, Marcel C. Bühler, Xu Chen, Michael J. Black, and Otmar Hilliges. I M

Avatar: Implicit Morphable Head Avatars From Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13545–13555, 2022. 7

[51] Yufeng Zheng, Wang Yifan, Gordon Wetzstein, Michael J. Black, and Otmar Hilliges. PointAvatar: Deformable Point-Based Head Avatars From Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21057–21067, 2023. 2, 3, 6, 7

[52] Zerong Zheng, Xiaochen Zhao, Hongwen Zhang, Boning Liu, and Yebin Liu. AvatarRex: Real-time Expressive Full-body Avatars. *ACM Transactions on Graphics (TOG)*, 42(4), 2023. 2, 3

[53] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Instant Volumetric Head Avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4574–4584, 2023. 2, 3, 4, 6, 7

[54] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface Splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page 371–378, New York, NY, USA, 2001. Association for Computing Machinery. 4