HYBRIDAVATAR: PHOTOREAL ANIMATABLE HUMANS AS MESH-NATIVE ASSETS FROM MULTI-VIEW CAMERAS

Anonymous authors
Paper under double-blind review

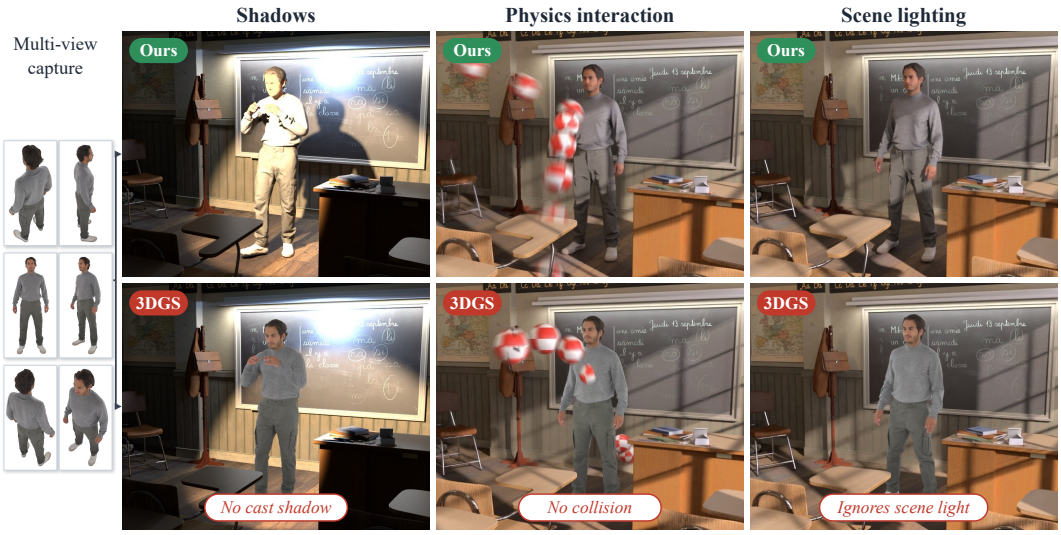

Figure 1: **HybridAvatar is photoreal and mesh-native.** Top: from a multi-view capture we build an animatable avatar that is compatible with classical graphics engines. Placed in a scene, it casts shadows (left), collides with a ball (middle), and is lit by the scene (right). Bottom: a 3DGS avatar rendered in the same scene from the same camera does none of these: it casts no shadow, lets the ball pass into its body, and does not respond to the scene light.

ABSTRACT

Mesh-based avatars have enjoyed decades of graphics engine support: physics, shadows, and relighting all assume mesh geometry. Recent 3D Gaussian Splatting (3DGS) avatars deliver higher rendering quality and can be reconstructed from multi-view captures with as few as 10–16 cameras, but their splat representation is incompatible with conventional graphics engines. We present a hybrid pipeline that bridges these two regimes. We train a pose-conditioned avatar with mesh-anchored Gaussian Splatting from multi-view video, then, for every training pose, extract three asset maps (texture, normal, displacement) with a UV-space rasterizer applied to the trained Gaussians. At inference time, geometry for a novel pose is produced by the trained network as a posed mesh, while appearance is produced by a pose-space mixture-of-Gaussians blend over the per-pose asset maps. The output is a textured mesh asset that can be loaded directly into any engine, supports physics and dynamic shadows, and responds to scene lighting, while approaching the rendering quality of 3DGS avatars. The same model can also be rendered directly as Gaussians at quality comparable to 3DGS avatars. In a *hybrid mode* it is rendered with Gaussians while its mesh, which shares their geometry, serves as a physics and shadow proxy. We evaluate the pipeline on multi-view human captures and show that it produces engine-ready assets whose photometric quality exceeds that of meshes extracted from 3DGS avatars and approaches that of direct renderings of animatable 3DGS avatars. Our project page is available at `https://anonymous.4open.science/w/HybridAvatarICLR27/`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Animatable digital humans are essential for a wide range of applications, from games and virtual production to telepresence, AR/VR, and embodied AI. In all of these settings, a human avatar must do two things: it must look right under a wide range of viewpoints and lighting conditions, and it must integrate with the rest of the production stack, including skeletal rigs, physics, shadow casting, relighting, post-processing, and content authoring. The de facto representation that satisfies the second requirement is a *posed mesh paired with UV-mapped texture, normal, and displacement maps* (Cohen et al., 1998; Akenine-Möller et al., 2018). Decades of graphics engine development assume that geometry is mesh-based: rigid-body physics, shadow mapping, screen-space global illumination, deferred relighting, and stylization shaders all consume meshes, not point clouds or splats (Gregory, 2018; Akenine-Möller et al., 2018).

Recent work on 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has produced a parallel ecosystem of human avatars (Li et al., 2024c; Kocabas et al., 2024) that pushes rendering quality far beyond what was previously achievable for animatable humans. These methods reconstruct from multi-view captures using as few as 10–16 cameras, a capture rig that is now within reach of independent studios, and they produce striking photometric realism. Crucially, 3DGS avatars are also *drivable*: at inference time, they can be animated to adopt novel poses, and be composed into virtual scenes. Their visual quality, modest capture footprint, and animation support make them an attractive replacement for the mesh-based asset pipeline.

But the splat representation is not free. A 3DGS avatar is a point cloud of anisotropic Gaussians, not a surface. It does not have a watertight mesh, well-defined surface normals, a texture atlas, or a displacement field. As a result, it cannot be rendered by standard real-time engines without bespoke rasterizers; it cannot take part in physics simulation; it cannot cast shadows against mesh geometry; and it does not respond to scene lights. The splat pipeline gives up the entire mesh-native production stack in exchange for its rendering quality.

This raises the central question of our work: *can we have both?* Can an animatable human avatar inherit, to a large extent, the photometric quality and capture friendliness of 3DGS while exporting as a clean mesh asset, with texture, normal, and displacement maps, that drops into existing engines and unlocks physics, shadow casting, and shading under scene lights?

We answer this question with a hybrid pipeline, summarized in Fig. 2. The pipeline trains a pose-conditioned avatar using mesh-anchored 3D Gaussian Splatting from multi-view video, where each Gaussian is rigidly attached to a triangle of an underlying SMPL-X-derived template mesh. Pose conditioning is provided to a UV-space neural network (a StyleUNet) that predicts both per-vertex geometric residuals and per-pixel Gaussian attributes in the canonical UV atlas. After training, we exploit the fact that every Gaussian sits on a known surface position to extract three *asset maps* (texture, normal, displacement) for every training pose.

At inference time, a single forward pass of the network supports three rendering modes. In *mesh mode*, only the predicted posed mesh is used, and appearance is reconstructed by a real-time pose-space mixture-of-Gaussians blend over the per-pose asset maps. The end product is a posed mesh and its three asset maps per query pose: a clean, engine-ready asset that can be rasterized with any standard renderer, lit by any environment, and integrated with mesh-based physics and shadow systems, while approaching the rendering quality of 3DGS avatars. In *3DGS mode*, the Gaussians predicted for the query pose are rendered directly by splatting. Because every Gaussian is anchored to the predicted mesh, both share exactly the same geometry, which enables a third, *hybrid mode*: the avatar is rendered with Gaussians while its mesh acts as a proxy for collisions, physics, and shadow casting. Experiments on multi-view human captures show that our avatars clearly outperform meshes extracted from splat avatars and a static mesh-anchored baseline, and support mesh-native operations such as shadow casting and relighting that splat-based methods cannot (Fig. 1).

Contributions. Our contributions are threefold. First, a pose-conditioned UV-space network that predicts both the posed mesh geometry and the mesh-anchored Gaussians attached to it, which together form a *HybridAvatar*. Second, texture, normal, and displacement maps extracted from the trained Gaussians for every training pose, and a pose-space blend that combines them into the maps of a novel pose. The result is a standard textured mesh, produced at interactive rates, which any conventional engine can load with no Gaussian rendering at run-time. Third, three rendering modes from one trained model: a 3DGS mode at quality comparable to dedicated splat-based avatars, a

mesh mode for engine-ready assets, and a hybrid mode in which the Gaussians are rendered while the mesh, which shares their geometry, serves as a physics and shadow proxy.

2 Related Work

Neural Rendering. NeRF (Mildenhall et al., 2020) sparked wide interest in neural rendering (Xie et al., 2022), yet remains limited by computational cost despite follow-up improvements (Müller et al., 2022; Barron et al., 2022; 2023; Tancik et al., 2023). 3DGS (Kerbl et al., 2023) addresses this by representing scenes as explicit 3D Gaussian primitives, extending sphere-based approaches (Lassner & Zollhöfer, 2021), and rasterizing them via splatting (Westover, 1991). Originally designed for static scenes, 3DGS has been extended to dynamic scenes (Shaw et al., 2024; Luiten et al., 2024; Wu et al., 2024; Lee et al., 2024; Li et al., 2024a), SLAM-based reconstruction (Keetha et al., 2024), mesh reconstruction (Huang et al., 2024; Guédon & Lepetit, 2024), and NVS from sparse (Mihajlovic et al., 2024) and embodied views (Wang et al., 2025).

Human Reconstruction and Neural Rendering. Mesh-based templates (Pavlakos et al., 2019; Loper et al., 2015) recover 3D human shape and pose (Bogo et al., 2016; Kanazawa et al., 2018), but despite improvements (Alldieck et al., 2018; 2019; Mir et al., 2020; Shin et al., 2024) do not yield photorealistic renderings. Implicit functions (Mescheder et al., 2019; Park et al., 2019) reconstruct detailed clothed humans (Chen et al., 2021; Alldieck et al., 2021; Saito et al., 2020; He et al., 2021; Huang et al., 2020; Deng et al., 2020), yet also lack photorealism and are often not reposable. NeRF-based methods (Peng et al., 2021; Guo et al., 2023; Weng et al., 2022; Jiang et al., 2022; Habermann et al., 2023; Zhu et al., 2024; Li et al., 2022; Liu et al., 2021; Xu et al., 2021) produce photorealistic controllable humans but inherit NeRF's speed limitations. Recent 3DGS-based approaches (Kocabas et al., 2024; Qian et al., 2024; Moreau et al., 2024; Abdal et al., 2024; Zielonka et al., 2025; Moon et al., 2024; Li et al., 2024c; Pang et al., 2024; Lei et al., 2024; Hu et al., 2024; Li et al., 2024b; Zheng et al., 2024a; Jiang et al., 2024; Dhamo et al., 2024; Xu et al., 2024; Guo et al., 2025; Qiu et al., 2025; Junkawitsch et al., 2025; Moreau et al., 2025; Mir et al., 2026a;b; 2025) build controllable human and face avatars with strong rendering quality, but their splat representation is not directly compatible with conventional graphics engines, which expect mesh-based geometry for physics, shadow casting, and relighting. Even when such avatars carry a mesh template internally, as in UMA (Zhu et al., 2026), which aligns Gaussians to a tracked template with 2D point-track supervision to reach ultra-fine detail from 40 cameras, the mesh serves only as a scaffold and appearance is still rendered by splatting.

Hybrid 3DGS-Mesh Representations. A complementary line of work constrains Gaussian primitives to lie on or near a mesh surface to recover surface-consistent geometry. Surface-aligned formulations (Guédon & Lepetit, 2024; Huang et al., 2024) flatten Gaussians into surfels for efficient mesh extraction. Another line of work (Held et al., 2026a;b) uses the 3DGS optimization framework to reconstruct 3D meshes or assets based on triangles from RGB images. Recent human-specific methods anchor Gaussians directly to a body-model surface for animation (Wen et al., 2024; Zhou et al., 2025). DeMapGS (Zhou et al., 2025) additionally extracts canonical UV maps from the trained representation. MeshAvatar (Chen et al., 2024) learns an explicit triangle mesh, extracted from a signed distance field by marching tetrahedra and posed by LBS, but decodes albedo and roughness at each surface point with a pose-conditioned network and shades them with a Monte Carlo physically based renderer. Its material decomposition supports relighting, but the avatar has no UV parameterization or texture maps, and its appearance at a new pose requires evaluating this network, so it cannot be loaded into a conventional engine as a textured mesh without an additional per-pose baking step. It also has no Gaussian representation, so it offers neither the photometric quality of splatting nor a hybrid mode in which splat rendering is paired with a matching mesh proxy. Physics-based avatars (Zheng et al., 2024b; Lee et al., 2025) also attach Gaussians to a tracked garment mesh, but use the mesh to drive a cloth simulator and keep appearance static, so they require dense capture and per-subject garment preparation and still render through a Gaussian rasterizer. Our asset instead exports the mesh itself with pose-dependent asset maps, so physics, shadows, and object interaction are provided by the target engine. Our pipeline builds on the anchoring strategy of DeMapGS and extends it to a pose-conditioned, multi-frame setting in which we extract per-pose asset maps (texture, normal, displacement) usable with a standalone mesh, decoupled from the underlying Gaussian representation at inference time.

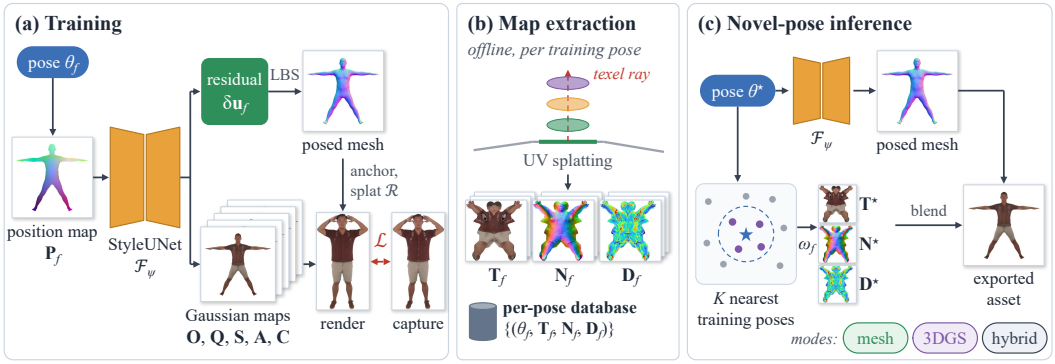

Figure 2: **Method overview.** (a) Training: for each training frame f, the StyleUNet $\mathcal{F}_\psi$ maps the posed position map $\mathbf{P}_f$ to a vertex residual $\delta\mathbf{u}_f$, which gives the posed mesh after LBS, and to the Gaussian maps $\mathbf{O}_f, \mathbf{Q}_f, \mathbf{S}_f, \mathbf{A}_f, \mathbf{C}_f$; the Gaussians are anchored to the posed mesh, splatted with rasterizer $\mathcal{R}$, and compared with the capture by loss $\mathcal{L}$. (b) Per-pose asset map extraction: offline, the UV-space rasterizer applied to the trained Gaussians of each training pose gives its three asset maps $\mathbf{T}_f, \mathbf{N}_f, \mathbf{D}_f$, stored in a per-pose database. (c) Novel-pose inference: the network gives the posed mesh for the query pose $\boldsymbol{\theta}^\star$, and its asset maps $\mathbf{T}^\star, \mathbf{N}^\star, \mathbf{D}^\star$ are blended from the K nearest training poses with weights ω_f. The same forward pass supports the mesh, 3DGS, and hybrid rendering modes.

3 Method

Our pipeline, illustrated in Fig. 2, has three stages. During *training*, a pose-conditioned UV-space network predicts, for each frame, a posed template mesh and a set of Gaussians anchored to it, which are splatted into the training cameras and supervised photometrically (Secs. 3.1–3.3). After training, *per-pose asset map extraction* rasterizes the Gaussians of every training pose into three *asset maps* in the canonical UV atlas: a texture, a normal, and a displacement map (Sec. 3.4). At *inference*, a single forward pass gives both the posed mesh and its anchored Gaussians for an input pose, which supports three rendering modes (Sec. 3.5): a mesh mode, whose asset maps are blended from the per-pose asset maps in pose space; a 3DGS mode; and a hybrid mode that renders the Gaussians and uses the mesh as a physics and shadow proxy. Further implementation details are given in App. A.

3.1 Mesh-Anchored Gaussian Avatar

Canonical Template. We build a subject-specific template by evaluating SMPL-X at the subject's identity shape $\boldsymbol{\beta}$ and zero pose, with the single-island UV unwrap of Zhou et al. (2025) for the body-without-eyes topology. This gives a canonical mesh with N_V template vertices $\bar{\mathbf{v}}^{\mathsf{C}}$ and a UV atlas of $N_{\text{uv}} \times N_{\text{uv}}$ pixels, with $N_{\text{uv}} = 1024$, whose topology is shared by all later stages. Skinning weights $\mathbf{W} \in \mathbb{R}^{N_V \times N_J}$ over the N_J SMPL-X joints are inherited from SMPL-X (App. A), and for the pose $\boldsymbol{\theta}_f$ of training frame $f \in \{1, \ldots, F\}$, SMPL-X forward kinematics yields per-bone transforms $\mathbf{G}_f \in \mathbb{R}^{N_J \times 4 \times 4}$, into which we fold the global orientation and root translation of the frame. These transforms pose a canonical mesh by linear blend skinning (LBS):

$$\mathbf{v}_m^{\mathsf{P}}(\boldsymbol{\theta}_f) = \sum_{j=1}^{N_J} w_{m,j}\, \mathbf{G}_{f,j}\, \mathbf{v}_m^{\mathsf{C}}, \tag{1}$$

where $\mathbf{v}_m^{\mathsf{C}}$ is canonical vertex m in homogeneous coordinates and $w_{m,j}$ is its weight to joint j. We apply Eq. 1 both to the template $\bar{\mathbf{v}}^{\mathsf{C}}$, to build the network input (Sec. 3.2), and to the pose-corrected canonical mesh $\mathbf{v}^{\mathsf{C}}(\boldsymbol{\theta}_f)$ of Eq. 4, to obtain the posed mesh.

Gaussian-to-Mesh Anchoring. Following Zhou et al. (2025), we anchor every Gaussian to a triangle of the template. Unlike their splats, which carry learnable barycentric coordinates and can move across triangles, we place exactly one Gaussian at each foreground pixel of the UV atlas and store its parameters as UV maps, as in Li et al. (2024c) (Fig. 3a). We precompute a static, pose-independent lookup that associates each foreground UV pixel p with the vertex indices $(m_{p,1}, m_{p,2}, m_{p,3})$ of the triangle containing it and the corresponding barycentric weights $\mathbf{b}_p$. For a posed frame, the Gaussian at pixel p is placed by blending the three posed vertices of that triangle

and adding a small along-normal offset o_p predicted by the network:

$$\boldsymbol{\mu}_p^{\mathsf{P}}(\boldsymbol{\theta}_f) = \sum_{i=1}^{3} b_{p,i}\, \mathbf{v}_{m_{p,i}}^{\mathsf{P}}(\boldsymbol{\theta}_f) \;+\; o_p\, \mathbf{n}_p^{\mathsf{P}}(\boldsymbol{\theta}_f), \tag{2}$$

where $\mathbf{n}_p^{\mathsf{P}}$ is the normal of the posed triangle containing p. This triangle also defines a local frame $\mathbf{E}_p^{\mathsf{P}}(\boldsymbol{\theta}_f) \in SO(3)$, whose third column is $\mathbf{n}_p^{\mathsf{P}}$ and whose first column is the direction of increasing u in the UV parameterization, projected onto the triangle plane. Rotation, scale, opacity, and color are read from the same UV pixel, and the rotation is expressed in this frame, $\mathbf{R}_p^{\mathsf{P}}(\boldsymbol{\theta}_f) = \mathbf{E}_p^{\mathsf{P}}(\boldsymbol{\theta}_f)\,\mathrm{Rot}(\bar{\mathbf{q}}_p)$, where $\bar{\mathbf{q}}_p$ is the unit quaternion predicted at p, $\mathrm{Rot}(\cdot)$ converts a quaternion to a rotation matrix, and $\mathbf{q}_p^{\mathsf{P}}$ denotes the quaternion of $\mathbf{R}_p^{\mathsf{P}}$. This anchoring ties every Gaussian to the surface (Fig. 3b), so the trained representation is consistent with a textured mesh at every pose, a property we exploit in Sec. 3.4.

3.2 Pose-Conditioned UV Network

Pose Conditioning. Following Li et al. (2024c), the network input is a *posed position map* $\mathbf{P}_f = \mathbf{P}(\boldsymbol{\theta}_f) \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 3}$: we pose the template with Eq. 1 and rasterize the posed vertex positions $\mathbf{v}^{\mathsf{P}}(\boldsymbol{\theta}_f)$ into the UV atlas with the same lookup, leaving background pixels at zero. Each UV texel thus sees the posed configuration of its own mesh region, a spatially local pose signal.

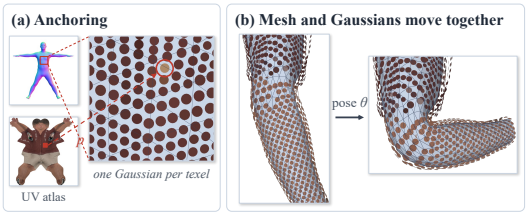

Figure 3: **Mesh-anchored Gaussians.** (a) Each foreground texel p of the UV atlas owns one Gaussian, placed on the triangle that contains it. (b) Tied to their triangles, the Gaussians follow the mesh as the pose changes.

Network Architecture. A StyleUNet $\mathcal{F}_\psi$ with weights ψ (Li et al., 2024c), a UV-space encoder/decoder with style modulation, maps the posed position map to a multi-channel canonical UV output:

$$\mathcal{F}_\psi(\mathbf{P}_f) \;\rightarrow\; (\delta\mathbf{u}_f,\, \mathbf{O}_f,\, \mathbf{Q}_f,\, \mathbf{S}_f,\, \mathbf{A}_f,\, \mathbf{C}_f). \tag{3}$$

The first block $\delta\mathbf{u}_f \in \mathbb{R}^{N_V \times 3}$ is a per-vertex residual, read from the output map at the UV pixels of the template vertices. It lives in the differential parameterization of Nicolet et al. (2021): writing $\mathbf{L}$ for the mesh Laplacian, the canonical vertex positions for the current pose are recovered by a linear solve rather than by addition,

$$\mathbf{v}^{\mathsf{C}}(\boldsymbol{\theta}_f) \;=\; \bar{\mathbf{v}}^{\mathsf{C}} \;+\; \kappa \left(\mathbf{I} + \lambda_{\mathrm{diff}}\mathbf{L}\right)^{-1} \delta\mathbf{u}_f, \tag{4}$$

where λ_{diff} sets the strength of the parameterization, κ is a fixed residual scale, and the inverse is applied through a cached Cholesky factorization. Since this inverse acts as a low-pass filter on the mesh, the vertex displacement stays smooth for any network output, and the explicit regularizer of Sec. 3.3 complements this filtering. The remaining outputs are dense $N_{\mathrm{uv}} \times N_{\mathrm{uv}}$ maps of per-Gaussian attributes, which we read at each foreground pixel p: the along-normal offset $\mathbf{O}_f$, giving the o_p of Eq. 2; the rotation map $\mathbf{Q}_f$, whose normalized value gives the quaternion $\bar{\mathbf{q}}_p$ of Sec. 3.1; the log scale $\mathbf{S}_f$, with $\mathbf{s}_p = \exp(\mathbf{S}_f(p))$; the opacity logit $\mathbf{A}_f$, with $\alpha_p = \mathrm{sigmoid}(\mathbf{A}_f(p))$; and the view-independent color $\mathbf{C}_f$, giving $\mathbf{c}_p$. In the 2D stages of Sec. 3.3, the Gaussians are planar and only the first two scale channels are used. From these outputs, we assemble the posed Gaussian set

$$\mathcal{G}^{\mathsf{P}}(\boldsymbol{\theta}_f) \;=\; \left\{(\boldsymbol{\mu}_p^{\mathsf{P}},\, \mathbf{q}_p^{\mathsf{P}},\, \mathbf{s}_p,\, \alpha_p,\, \mathbf{c}_p)\right\}_{p \in \mathcal{U}_{\mathrm{fg}}}, \tag{5}$$

where $\mathcal{U}_{\mathrm{fg}}$ is the set of foreground UV pixels, and the posed mean is given by Eq. 2. During training, $\mathcal{G}^{\mathsf{P}}$ is rendered into each training camera by a differentiable Gaussian rasterizer $\mathcal{R}$, which is the surfel rasterizer of Huang et al. (2024) in the 2D stages and that of Kerbl et al. (2023) in the 3D stage.

3.3 Training Schedule and Losses

Three-Stage Schedule. We follow the 2DGS→3DGS→2DGS schedule of Zhou et al. (2025): 2D Gaussians (Huang et al., 2024) first give stable surface-aligned coverage, 3D Gaussians (Kerbl et al., 2023) then recover volumetric appearance detail, and a final 2D stage returns to surface-aligned Gaussians before asset extraction. The same network $\mathcal{F}_\psi$ is optimized across all stages.

Losses. The loss for a sampled (frame, camera) pair (f, c) combines a masked photometric term, a structural similarity term, an alpha-mask term, and a Laplacian regularizer on the vertex residual:

$$\mathcal{L} \;=\; \lambda_{\text{rgb}}\,\mathcal{L}_{\text{rgb}} \;+\; \lambda_{\text{ssim}}\,\mathcal{L}_{\text{ssim}} \;+\; \lambda_{\text{mask}}\,\mathcal{L}_{\text{mask}} \;+\; \lambda_{\text{lap}}\,\mathcal{L}_{\text{lap}}, \tag{6}$$

where $\hat{\mathbf{I}}_{f,c}$ and $\mathbf{I}_{f,c}$ are the rendered and captured images, $\mathcal{L}_{\text{rgb}} = \|(\hat{\mathbf{I}}_{f,c} - \mathbf{I}_{f,c}) \odot \mathbf{M}_{f,c}\|_1$, $\mathcal{L}_{\text{ssim}} = 1 - \text{SSIM}(\hat{\mathbf{I}}_{f,c}, \mathbf{I}_{f,c})$, $\mathcal{L}_{\text{mask}} = \|\hat{\mathbf{M}}_{f,c} - \mathbf{M}_{f,c}\|_1$ compares the rendered alpha $\hat{\mathbf{M}}_{f,c}$ with the foreground mask $\mathbf{M}_{f,c}$, and $\mathcal{L}_{\text{lap}} = \|\mathbf{L}\,(\mathbf{v}^{\mathsf{C}}(\boldsymbol{\theta}_f) - \bar{\mathbf{v}}^{\mathsf{C}})\|_{\text{F}}^2$ penalizes the Laplacian of the pose-dependent vertex displacement of Eq. 4. This term keeps pose-dependent mesh deformations smooth and the per-pose asset maps consistent across nearby poses. Loss weights $\lambda_\cdot$ are listed in App. B.

3.4 Per-Pose Asset Map Extraction

After training, we extract the three asset maps for each of the F training poses. Because every Gaussian is anchored to a known triangle of the posed mesh (Eq. 2), each posed Gaussian projects deterministically onto the canonical UV atlas. We accumulate per-Gaussian contributions into the atlas with a UV-space rasterizer, which sorts the Gaussians covering a texel by height along the face normal and composites them front-to-back with transmittance, as in volume rendering. This gives, per pose, a texture map $\mathbf{T}_f \in \mathbb{R}^{N_{\text{uv}} \times N_{\text{uv}} \times 3}$ together with the accumulated opacity $\hat{\boldsymbol{\alpha}}_f \in \mathbb{R}^{N_{\text{uv}} \times N_{\text{uv}}}$; a normal map $\mathbf{N}_f \in \mathbb{R}^{N_{\text{uv}} \times N_{\text{uv}} \times 3}$; and a displacement map $\mathbf{D}_f \in \mathbb{R}^{N_{\text{uv}} \times N_{\text{uv}} \times 1}$, which encodes mesoscale surface detail. For the normal map, the normal of each Gaussian, the third axis of its rotation, is rotated into the frame $\mathbf{E}^{\mathsf{P}}$ (Sec. 3.1) of the triangle containing the texel before compositing, and the result is renormalized. Because this frame is defined by the UV parameterization, it moves with the surface, so the normal maps of all poses live in a common frame and can be blended by Eq. 10; it is also the frame in which standard engines expect normal maps. The displacement map is an equal mix of the opacity-normalized expected offset and the median offset,

$$\mathbf{D}_f \;=\; \tfrac{1}{2}\,\mathbf{D}_f^{\text{med}} \;+\; \tfrac{1}{2}\,\mathbf{D}_f^{\text{exp}} \oslash \max\!\big(\hat{\boldsymbol{\alpha}}_f,\, \epsilon\big), \tag{7}$$

where $\mathbf{D}_f^{\text{exp}}$ is the transmittance-weighted sum of the offsets o_p, $\mathbf{D}_f^{\text{med}}$ is the offset of the Gaussian at which the transmittance falls below one half, $\oslash$ denotes element-wise division, and ϵ is a small constant that avoids division by zero. The result is a database $\{(\boldsymbol{\theta}_f,\, \mathbf{T}_f,\, \mathbf{N}_f,\, \mathbf{D}_f)\}_{f=1}^{F}$ of per-pose asset maps. Extraction requires no backpropagation and runs once per training pose, offline.

3.5 Novel-Pose Inference

At inference time, given a query pose $\boldsymbol{\theta}^\star$, the network is invoked once, and its output supports three rendering modes, summarized at the end of this subsection. In mesh mode, which we describe first, the network provides posed mesh geometry, and appearance is obtained by a pose-space mixture-of-Gaussians blend over the per-pose asset maps; no Gaussian rasterization is required.

Posed Mesh from the Network. We evaluate the network of Eq. 3 at the query pose, $\mathcal{F}_\psi(\mathbf{P}(\boldsymbol{\theta}^\star))$, and keep only its vertex residual $\delta\mathbf{u}^\star$. We map it to canonical positions $\mathbf{v}^{\mathsf{C},\star}$ with Eq. 4 and apply LBS (Eq. 1) to obtain the posed mesh $\mathbf{v}^{\mathsf{P},\star}(\boldsymbol{\theta}^\star)$. Mesh mode does not need the remaining outputs; they are used by the 3DGS mode below.

Why a Pose-Space Blend. The network output at $\boldsymbol{\theta}^\star$ also contains the Gaussian attributes, so asset maps could in principle be extracted from it directly. The cost this avoids is not the network forward pass, which mesh mode performs at every frame in any case, but the extraction itself. Producing asset maps for a query pose requires the custom CUDA UV-space rasterizer of Sec. 3.4, which depth-sorts and composites every Gaussian falling on every triangle. This rasterizer takes 25.5 s per pose on an H100 GPU, 99.7% of the total extraction time, and has no equivalent in a conventional engine. We therefore extract asset maps only for the training poses, offline, and obtain the maps of a query pose with the blend of Eq. 10. The blend is a weighted sum of $K = 16$ textures that takes 2.1 ms, four orders of magnitude less, and can be evaluated in any fragment shader.

Pose-Space Distance. We define a pose distance from the SMPL-X joint positions, which is robust to axis-angle wrap-around and exposes the same geometric structure that drives mesh deformation. For poses $\boldsymbol{\theta}, \boldsymbol{\theta}'$ let $\mathbf{J}(\boldsymbol{\theta}) \in \mathbb{R}^{N_J \times 3}$ denote the SMPL-X joint locations (no shape factor, identity removed); then

$$d(\boldsymbol{\theta},\, \boldsymbol{\theta}') \;=\; \frac{1}{\sqrt{N_J}}\,\big\|\mathbf{J}(\boldsymbol{\theta}) - \mathbf{J}(\boldsymbol{\theta}')\big\|_{\text{F}}. \tag{8}$$

Mixture-of-Gaussians Blend. For a query pose $\boldsymbol{\theta}^\star$, we form normalized kernel weights over its K nearest training poses and blend their asset maps:

$$\omega_f(\boldsymbol{\theta}^\star) \;=\; \frac{\exp\!\big(-d(\boldsymbol{\theta}^\star, \boldsymbol{\theta}_f)^2/2\sigma^2\big)}{\sum_{f' \in \mathcal{N}_K(\boldsymbol{\theta}^\star)} \exp(-d(\boldsymbol{\theta}^\star, \boldsymbol{\theta}_{f'})^2/2\sigma^2)}, \qquad f \in \mathcal{N}_K(\boldsymbol{\theta}^\star), \tag{9}$$

$$\mathbf{X}^\star \;=\; \sum_{f \in \mathcal{N}_K(\boldsymbol{\theta}^\star)} \omega_f(\boldsymbol{\theta}^\star)\, \mathbf{X}_f, \qquad \mathbf{X} \in \{\mathbf{T},\, \mathbf{N},\, \mathbf{D}\}, \tag{10}$$

where $\mathcal{N}_K(\boldsymbol{\theta}^\star)$ is the set of the $K = 16$ training poses nearest to $\boldsymbol{\theta}^\star$ under d, the bandwidth σ is set from the median nearest-neighbor pose distance in the training database, and the normal map is renormalized after blending. Restricting the blend to K poses keeps its cost independent of the database size: each query reads and sums only K sets of asset maps, while the neighbor search runs on the $N_J \times 3$ joint locations.

Asset Export. The asset for the query pose is the posed mesh with its three blended asset maps, $\big(\mathbf{v}^{\mathsf{P},\star},\, \mathbf{T}^\star,\, \mathbf{N}^\star,\, \mathbf{D}^\star\big)$, written as a standard OBJ with PNG/EXR maps (App. F). Because it is a textured mesh, it composes with mesh-native physics, shadow casting, and shading by scene lights, none of which the underlying Gaussian representation supports.

Three Rendering Modes. The same forward pass also yields the full posed Gaussian set $\mathcal{G}^{\mathsf{P}}(\boldsymbol{\theta}^\star)$ of Eq. 5, assembled from the remaining outputs of Eq. 3 exactly as during training. The avatar can therefore be rendered in *3DGS mode*, by splatting these Gaussians with $\mathcal{R}$, or in *mesh mode*, by rasterizing the exported asset. The first gives the highest rendering quality but requires a Gaussian renderer; the second is compatible with any conventional engine. Because every Gaussian is anchored to a triangle of the posed mesh (Eq. 2), both share the same geometry. This enables a third, *hybrid mode*: the avatar is rendered by splatting while the posed mesh of the same forward pass serves as a proxy for collisions, physics, and shadow casting, and the proxy matches the rendered surface by construction.

4 Experiments

We compare our mesh mode against animatable mesh-based baselines (Sec. 4.2), ablate each of our three components (Sec. 4.3), and evaluate the hybrid mode and the mesh-native operations (Sec. 4.4).

4.1 Setup

Data. We use two multi-view datasets with per-frame SMPL-X parameters: **AvatarReX** (Zheng et al., 2023) and **ActorsHQ** (Işık et al., 2023). All AvatarReX results are obtained from its 16-camera rig, of which 14 cameras are used for training. For ActorsHQ, we use 48 cameras for training and hold out 2–4 cameras for evaluation. For both datasets, we hold out the final 10% of frames for novel-pose evaluation.

Metrics. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) on foreground crops. Since our contributions concern the geometry of the asset as much as its appearance, we also report two metrics that need no ground-truth geometry: *silhouette IoU* against the captured masks, and *normal error*, the mean angle between rendered normals and normals predicted by Sapiens (Khirodkar et al., 2024) on the captured images.

4.2 Comparison with Baselines

Baselines. As splat-based references, we report **Animatable Gaussians** (Li et al., 2024c), which shares our pose-conditioning network but outputs only Gaussians, and our avatar in 3DGS mode. As mesh-asset baselines, we use **AG-Mesh**, Animatable Gaussians driven to the query pose and converted to a textured mesh by TSDF fusion, Marching Cubes, and projective texturing, the strongest post-hoc export of a splat avatar we could build; and **DeMapGS + LBS** (Zhou et al., 2025), trained on a single frame with our template and UV atlas, on the same cameras and without external normal supervision, and reposed by LBS with fixed asset maps. All mesh assets are rendered with OpenGL, unlit, with the same crops and masks as the splat renderings (details in App. F).

Table 1: **Quantitative comparison on held-out poses.** The upper block lists splat-based renderings (3DGS mode), which require a Gaussian renderer; the lower block lists textured-mesh assets (mesh mode). Within each block, the best result is highlighted in red and the second best in yellow. Anim.: the asset can be animated in a standard engine; Topo.: all poses share one mesh topology; UV: the asset carries all three asset maps (texture, normal, displacement). AG denotes Animatable Gaussians; AG-Mesh is its per-pose export, a new unrigged mesh for every pose.

	ActorsHQ			AvatarReX			Asset Properties		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	Anim.	Topo.	UV
3DGS mode: splat-based rendering									
AG (Li et al., 2024c)	28.85	0.938	0.058	29.71	0.975	0.044	–	–	–
Ours (3DGS mode)	28.49	0.935	0.061	29.36	0.973	0.046	–	–	–
Mesh mode: textured-mesh assets									
AG-Mesh	24.50	0.918	0.078	26.06	0.965	0.056	–	–	–
DeMapGS (Zhou et al., 2025) + LBS	22.23	0.911	0.098	21.66	0.947	0.099	•	•	•
Ours (mesh mode)	26.55	0.927	0.069	27.24	0.969	0.053	•	•	•

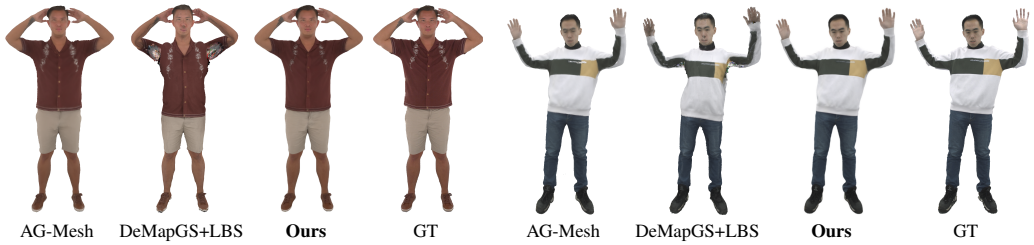

Figure 4: **Visual comparison of mesh assets on held-out poses.** Two held-out poses of two subjects. For each, from left to right: a mesh extracted from Animatable Gaussians (AG-Mesh), DeMapGS driven by linear blend skinning, our exported asset, and the captured reference.

Results. In 3DGS mode, our avatar is within 0.5 dB PSNR of Animatable Gaussians (Tab. 1), so the same model retains the quality of a dedicated splat avatar when a Gaussian renderer is available. Mesh mode costs 2.30 and 2.47 dB relative to Animatable Gaussians and 1.94 and 2.12 dB relative to our 3DGS mode. Among mesh-based methods (Tab. 1), ours is best on every metric on both datasets, improving over AG-Mesh by 2.05 and 1.18 dB PSNR and over DeMapGS + LBS by 4.32 and 5.58 dB. AG-Mesh is the closest competitor in image metrics, but it produces a new unrigged topology per pose, with a texture but no normal or displacement maps, so it is not a reusable asset. Visually (Fig. 4), AG-Mesh shows seams and blurred texture, and DeMapGS + LBS shows static wrinkles that do not follow the pose; our asset preserves pose-dependent wrinkles, with some loss of high-frequency detail. App. C compares against the splat renderings (Fig. 6).

4.3 Ablation Study

Tab. 2 and Fig. 5 ablate the various components of our method; further variants are in App. D.

MLP backbone. We replace the UV-space StyleUNet with an MLP that predicts the same per-texel outputs. Li et al. (2024c) compare backbones for Gaussian appearance only; in our pipeline the backbone also predicts the vertex residual that defines the exported mesh, so this row tests the convolutional UV design for both geometry and appearance.

LBS only. We remove the predicted vertex residual, so the mesh is the fixed template deformed by skinning alone (Eq. 1), without pose-dependent corrections. Garment folds are then left to the asset maps, which lowers silhouette IoU and raises normal error.

Static asset maps. We replace the per-pose asset maps with a single set averaged over all training poses. The appearance no longer depends on the pose, which raises LPIPS.

4.4 Hybrid Mode and Mesh-Native Operations

Proxy Consistency in Hybrid Mode. In hybrid mode, the avatar is rendered by splatting while a mesh provides collisions and shadows, which only works if the proxy coincides with the rendered surface: a larger proxy makes contacts and shadows float, and a smaller one lets objects penetrate the visible avatar. For every held-out pose we render the Gaussians and the proxy from the test cameras and report their silhouette IoU and mean absolute depth difference (Tab. 3). Our proxy (the posed



Figure 5: **Qualitative ablation on a held-out pose.** The first three columns each remove one component of the full model, as in Tab. 2; the full model and the captured reference are on the right. Without the vertex residual (LBS only), garment folds are missing from the geometry; with static asset maps, appearance no longer changes with the pose.

Table 2: **Ablation on held-out poses (AvatarReX).** Each row removes one component. NE: normal error (deg.). Best in red.

Variant	PSNR↑	LPIPS↓	IoU↑	NE↓
MLP backbone	25.61	0.066	0.958	22.9
LBS only	26.63	0.058	0.941	24.3
Static asset maps	26.38	0.063	0.970	20.1
Full	27.24	0.053	0.972	18.6

Table 3: **Proxy consistency in hybrid mode.** Agreement between the Gaussian rendering and its mesh proxy; depth error in mm.

Proxy	IoU↑	Depth↓
SMPL-X	0.883	27.4
AG-Mesh	0.956	8.9
Ours	0.984	3.2

mesh) agrees with the rendered surface up to the small offsets of Eq. 2, because every Gaussian is anchored to it. The posed SMPL-X body, the usual proxy for splat avatars, misses clothing, and AG-Mesh follows the surface more closely but must be recomputed for every pose.

Mesh-Native Operations. Fig. 1 places our avatar and a 3DGS avatar in the same classroom and renders both from the same camera. Our avatar casts its shadow onto the blackboard and floor, deflects a ball on contact, and receives the window light across its body. The 3DGS avatar does none of these: it casts no shadow, lets the ball pass into its body, and does not respond to the scene light. For more results, please see the project page.

5 Conclusion

We presented HybridAvatar, an animatable human avatar in which a single pose-conditioned UV-space network predicts both a posed mesh and the Gaussians anchored to its triangles, trained from multi-view video. Because the Gaussians are tied to the mesh, one forward pass supports three rendering modes. In mesh mode, texture, normal, and displacement maps extracted for every training pose are blended in pose space, which gives a standard textured mesh that conventional engines can load and that supports physics, shadow casting, and shading by scene lights. Its rendering quality approaches that of 3DGS avatars and clearly exceeds that of meshes extracted from splat avatars and of a static mesh-anchored baseline. In 3DGS mode, the same Gaussians are splatted directly, at quality comparable to dedicated splat-based avatars. In hybrid mode, the avatar is rendered with Gaussians while its mesh, which shares their geometry, serves as a physics and shadow proxy that matches the rendered surface by construction.

Limitations and Future Work. The pipeline inherits the geometric biases of the SMPL-X body model: topology is fixed, and loose clothing or hair beyond the surface envelope is poorly captured by the displacement field. Such garments therefore do not respond to contact with objects, although a simulation layer as in MPMAvatar (Lee et al., 2025) could be applied to our exported mesh, since it is rigged and engine-ready. The pose-space blend can over-smooth appearance details that are tied to specific poses, such as individual wrinkles. Finally, our asset stores appearance as static texture, normal, and displacement maps and does not separate albedo from shading: capture lighting remains baked into the texture, so the response to scene lighting is approximate, most visibly under strong directional light. These limitations suggest two directions for future work: (i) separating albedo from shading in the per-pose asset maps to enable physically based relighting, and (ii) replacing the kernel-based pose-space blend with a learned model, such as per-pose latents or a learned pose manifold, to better preserve high-frequency pose-dependent detail.

AI USE STATEMENT

We used AI-based tools in a supporting role only. A large language model was used for language editing of author-written text (grammar, phrasing, and shortening) and for routine code completion during implementation. The research question, method design, experimental protocol, and interpretation of results were produced by the authors without generative AI. All AI-assisted text and code were reviewed and verified by the authors, who take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

This work builds animatable photorealistic avatars of real people, which raises two concerns. First, data: we use publicly released multi-view capture datasets whose subjects consented to recording and redistribution under the terms of the respective dataset licenses, and we do not capture new human data. We follow those licenses, use the data only for research, and show only subjects whose datasets permit publication of renders. Second, misuse: photorealistic drivable avatars can be used to depict a person saying or doing something they did not do. Our pipeline needs a dense calibrated multi-view capture of a cooperating subject, which makes casual misuse from in-the-wild imagery impractical, but it does not eliminate the risk for subjects who have been captured. We recommend that any deployment couple avatar creation with explicit subject consent, retention limits, and provenance signaling on rendered output. The authors have read and adhere to the ICLR Code of Ethics.

REFERENCES

Rameen Abdal, Wang Yifan, Zifan Shi, Yinghao Xu, Ryan Po, Zhengfei Kuang, Qifeng Chen, Dit-Yan Yeung, and Gordon Wetzstein. Gaussian shell maps for efficient 3D human generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering*. A K Peters/CRC Press, fourth edition, 2018.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3D people models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Thiemo Alldieck, Hongyi Xu, and Cristian Sminchisescu. imGHUM: Implicit generative models of 3D human shape and articulated pose. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016.

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)*, 2021.

Yushuo Chen, Zerong Zheng, Zhe Li, Chao Xu, and Yebin Liu. MeshAvatar: Learning high-quality triangular human avatars from multi-view videos. In *European Conference on Computer Vision (ECCV)*, 2024.

Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *ACM SIGGRAPH Conference Proceedings*, 1998.

Boyang Deng, John P Lewis, Timothy Jeruzalski, Gerard Pons-Moll, Geoffrey Hinton, Mohammad Norouzi, and Andrea Tagliasacchi. NASA: Neural articulated shape approximation. In *European Conference on Computer Vision (ECCV)*, 2020.

Helisa Dhamo, Yinyu Nie, Arthur Moreau, Jifei Song, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. HeadGaS: Real-time animatable head avatars via 3D Gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.

Jason Gregory. *Game Engine Architecture*. A K Peters/CRC Press, third edition, 2018.

Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned Gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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.

Chen Guo, Junxuan Li, Yash Kant, Yaser Sheikh, Shunsuke Saito, and Chen Cao. Vid2Avatar-Pro: Authentic avatar from videos in the wild via universal prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Marc Habermann, Lingjie Liu, Weipeng Xu, Gerard Pons-Moll, Michael Zollhoefer, and Christian Theobalt. HDHumans: A hybrid approach for high-fidelity digital humans. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3), 2023.

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

Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. MeshSplatting: Differentiable rendering with opaque meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Daniel Rebain, Anthony Cioppa, Bernard Ghanem, Andrea Vedaldi, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle splatting for real-time radiance field rendering. In *International Conference on 3D Vision (3DV)*, 2026b.

Liangxiao Hu, Hongwen Zhang, Yuxiang Zhang, Boyao Zhou, Boning Liu, Shengping Zhang, and Liqiang Nie. GaussianAvatar: Towards realistic human avatar modeling from a single video via animatable 3D Gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH Conference Proceedings*, 2024.

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

Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. HumanRF: High-fidelity neural radiance fields for humans in motion. *ACM Transactions on Graphics*, 42(4), 2023.

Wei Jiang, Kwang Moo Yi, Golnoosh Samei, Oncel Tuzel, and Anurag Ranjan. NeuMan: Neural human radiance field from a single video. In *European Conference on Computer Vision (ECCV)*, 2022.

Yuheng Jiang, Zhehao Shen, Yu Hong, Chengcheng Guo, Yize Wu, Yingliang Zhang, Jingyi Yu, and Lan Xu. Robust dual Gaussian splatting for immersive human-centric volumetric videos. *ACM Transactions on Graphics*, 43(6), 2024.

Hendrik Junkawitsch, Guoxing Sun, Heming Zhu, Christian Theobalt, and Marc Habermann. EVA: Expressive virtual avatars from multi-view videos. In *ACM SIGGRAPH Conference Proceedings*, 2025.

Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. SplaTAM: Splat, track & map 3D Gaussians for dense RGB-D SLAM. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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.

Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Zhaoen Su, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models. In *European Conference on Computer Vision (ECCV)*, 2024.

Muhammed Kocabas, Jen-Hao Rick Chang, James Gabriel, Oncel Tuzel, and Anurag Ranjan. HUGS: Human Gaussian splats. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Changmin Lee, Jihyun Lee, and Tae-Kyun Kim. MPMAvatar: Learning 3D Gaussian avatars with accurate and robust physics-based dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Junoh Lee, ChangYeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic Gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiahui Lei, Yufu Wang, Georgios Pavlakos, Lingjie Liu, and Kostas Daniilidis. GART: Gaussian articulated template models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Ruilong Li, Julian Tanke, Minh Vo, Michael Zollhofer, Jurgen Gall, Angjoo Kanazawa, and Christoph Lassner. TAVA: Template-free animatable volumetric actors. In *European Conference on Computer Vision (ECCV)*, 2022.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

Zhe Li, Yipengjing Sun, Zerong Zheng, Lizhen Wang, Shengping Zhang, and Yebin Liu. Animatable and relightable Gaussians for high-fidelity human avatar modeling. *arXiv preprint arXiv:2311.16096*, 2024b.

Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable Gaussians: Learning pose-dependent Gaussian maps for high-fidelity human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024c.

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 Transactions on Graphics*, 2021.

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

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *International Conference on 3D Vision (3DV)*, 2024.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. SplatFields: Neural Gaussian splats for sparse 3D and 4D reconstruction. In *European Conference on Computer Vision (ECCV)*, 2024.

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 *European Conference on Computer Vision (ECCV)*, 2020.

Aymen Mir, Thiemo Alldieck, and Gerard Pons-Moll. Learning to transfer texture from clothing images to 3D humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Aymen Mir, Jian Wang, Riza Alp Guler, Chuan Guo, Gerard Pons-Moll, and Bing Zhou. Aha! animating human avatars in diverse scenes with gaussian splatting. *arXiv preprint arXiv:2511.09827*, 2025.

Aymen Mir, Riza Alp Guler, Xiangjun Tang, Peter Wonka, and Gerard Pons-Moll. Ahoy! animatable humans under occlusion from youtube videos with gaussian splatting and video diffusion priors. In *European Conference on Computer Vision (ECCV)*, 2026a.

Aymen Mir, Riza Alp Guler, Jian Wang, Peter Wonka, Bing Zhou, and Gerard Pons-Moll. Raga: Real time ray traced gaussian shadow casting for 3dgs avatar-scene interaction. In *European Conference on Computer Vision (ECCV)*, 2026b.

Gyeongsik Moon, Takaaki Shiratori, and Shunsuke Saito. Expressive whole-body 3D Gaussian avatar. In *European Conference on Computer Vision (ECCV)*, 2024.

Arthur Moreau, Jifei Song, Helisa Dhamo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Human Gaussian splatting: Real-time rendering of animatable avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Arthur Moreau, Mohammed Brahimi, Richard Shaw, Athanasios Papaioannou, Thomas Tanay, Zhensong Zhang, and Eduardo Pérez-Pellitero. Better together: Unified motion capture and 3D avatar reconstruction. *arXiv preprint arXiv:2503.09293*, 2025.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4), 2022.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics*, 40(6), 2021.

Haokai Pang, Heming Zhu, Adam Kortylewski, Christian Theobalt, and Marc Habermann. ASH: Animatable Gaussian splats for efficient and photoreal human rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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 of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable avatars via deformable 3D Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Lingteng Qiu, Shenhao Zhu, Qi Zuo, Xiaodong Gu, Yuan Dong, Junfei Zhang, Chao Xu, Zhe Li, Weihao Yuan, Liefeng Bo, Guanying Chen, and Zilong Dong. AniGS: Animatable Gaussian avatar from a single image with inconsistent Gaussian reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. PIFuHD: Multi-level pixel-aligned implicit function for high-resolution 3D human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Richard Shaw, Michal Nazarczuk, Jifei Song, Arthur Moreau, Sibi Catley-Chandar, Helisa Dhamo, and Eduardo Perez-Pellitero. SWinGS: Sliding windows for dynamic 3D Gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.

Jisu Shin, Junmyeong Lee, Seongmin Lee, Min-Gyu Park, Ju-Mi Kang, Ju Hong Yoon, and Hae-Gon Jeon. CanonicalFusion: Generating drivable 3D human avatars from multiple images. In *European Conference on Computer Vision (ECCV)*, 2024.

Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH Conference Proceedings*, 2023.

Xiaoyuan Wang, Yizhou Zhao, Botao Ye, Xiaojun Shan, Weijie Lyu, Lu Qi, Kelvin C. K. Chan, Yinxiao Li, and Ming-Hsuan Yang. HoliGS: Holistic Gaussian splatting for embodied view synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

Jing Wen, Xiaoming Zhao, Zhongzheng Ren, Alexander G. Schwing, and Shenlong Wang. GoMAvatar: Efficient animatable human modeling from monocular video using Gaussians-on-Mesh. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Lee Alan Westover. *Splatting: a parallel, feed-forward volume rendering algorithm*. PhD thesis, University of North Carolina at Chapel Hill, 1991.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

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 (NeurIPS)*, 2021.

Yuelang Xu, Benwang Chen, Zhe Li, Hongwen Zhang, Lizhen Wang, Zerong Zheng, and Yebin Liu. Gaussian head avatar: Ultra high-fidelity head avatar via dynamic Gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. GPS-Gaussian: Generalizable pixel-wise 3D Gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

Yang Zheng, Qingqing Zhao, Guandao Yang, Wang Yifan, Donglai Xiang, Florian Dubost, Dmitry Lagun, Thabo Beeler, Federico Tombari, Leonidas Guibas, and Gordon Wetzstein. PhysAvatar: Learning the physics of dressed 3D avatars from visual observations. In *European Conference on Computer Vision (ECCV)*, 2024b.

Zerong Zheng, Xiaochen Zhao, Hongwen Zhang, Boning Liu, and Yebin Liu. AvatarReX: Real-time expressive full-body avatars. *ACM Transactions on Graphics*, 42(4), 2023.

Shuyi Zhou, Shengze Zhong, Kenshi Takayama, Takafumi Taketomi, and Takeshi Oishi. DeMapGS: Simultaneous mesh deformation and surface attribute mapping via Gaussian splatting. In *ACM SIGGRAPH Asia Conference Papers*, 2025.

Heming Zhu, Fangneng Zhan, Christian Theobalt, and Marc Habermann. TriHuman: A real-time and controllable tri-plane representation for detailed human geometry and appearance synthesis. *ACM Transactions on Graphics*, 2024.

Heming Zhu, Guoxing Sun, Christian Theobalt, and Marc Habermann. UMA: Ultra-detailed human avatars via multi-level surface alignment. *ACM Transactions on Graphics*, 2026.

Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3D Gaussian avatars. In *International Conference on 3D Vision (3DV)*, 2025.

A Additional Method Details

In Sec. 3 of the main paper, we refer to the appendix for further implementation details of the template, the network outputs, and the training loop. We provide them here.

Template Construction and Skinning. Starting from SMPL-X evaluated at the subject's shape $\boldsymbol{\beta}$ and zero pose, we apply the single-island UV unwrap of Zhou et al. (2025) and prune degenerate triangles. Pruning merges some vertices, so the SMPL-X skinning weights cannot be copied directly. We track merged vertices with a union-find structure and assign each vertex of the final template the average of the SMPL-X weights of the vertices merged into it, which yields the weights $\mathbf{W} \in \mathbb{R}^{N_V \times N_J}$ used in Eq. 1.

Network Outputs. The dense outputs of Eq. 3 have the following channel layout, for 15 channels in total together with the 3-channel vertex residual:

- $\mathbf{O}_f \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 1}$: along-normal offset o_p (Eq. 2),
- $\mathbf{Q}_f \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 4}$: unnormalized quaternion, normalized to $\bar{\mathbf{q}}_p$ in the triangle frame of Sec. 3.1,
- $\mathbf{S}_f \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 3}$: log per-axis scale, with $\mathbf{s}_p = \exp(\mathbf{S}_f(p))$,
- $\mathbf{A}_f \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 1}$: opacity logit, with $\alpha_p = \mathrm{sigmoid}(\mathbf{A}_f(p))$,
- $\mathbf{C}_f \in \mathbb{R}^{N_{\mathrm{uv}} \times N_{\mathrm{uv}} \times 3}$: view-independent color $\mathbf{c}_p$.

The final convolution is initialized near zero, with biases chosen so that, at the start of training, the Gaussians lie pinned to the template surface with sensible default scale and opacity.

Training Loop. Each training step samples one (frame, camera) pair (f, c), runs the full forward pass of Eqs. 1–5, renders $\mathcal{G}^{\mathsf{P}}(\boldsymbol{\theta}_f)$ into camera c, evaluates the loss of Eq. 6, and backpropagates through the differentiable rasterizer into the network $\mathcal{F}_\psi$.

B Implementation Details

In Sec. 3.3 of the main paper, we refer to the appendix for the loss weights. We provide them here, together with further implementation details.

Architecture and Training. The SMPL-X template starts with 6,327 vertices; each face is subdivided twice to increase surface resolution, and the UV atlas is 1024×1024 pixels; the StyleUNet follows Li et al. (2024c) with 15 output channels. We train on a single NVIDIA H100 GPU, with λ_{rgb}=1, λ_{ssim}=0.2, λ_{mask}=0.5, and λ_{lap}=5.0.

Pose-Space Blend. The bandwidth σ is the median nearest-neighbor distance in the training poses, and $K = 16$. App. E gives the bandwidth and neighbor selection procedure.

Timing. At inference, the network takes 38 ms and the blend 2.1 ms; excluding the network, the asset renders at 410 FPS with OpenGL, and the avatar renders at 95 FPS in 3DGS mode. Export details are in App. F.

C Additional Qualitative Results

In Sec. 4.2 of the main paper, we compare mesh assets on held-out poses and report splat-based renderings as a reference. Here we provide additional visual comparisons between the splat-based renderings and our exported mesh asset.

Comparison with Splat-Based Rendering. Tab. 1 of the main paper reports the splat-based renderings of Animatable Gaussians (Li et al., 2024c) and of our own trained Gaussians as a reference for the gap between volumetric and mesh-based rendering, while Fig. 4 compares only mesh assets. Fig. 6 shows this gap visually on held-out poses. The splat renderings retain slightly sharper high-frequency detail, most visibly in pose-dependent wrinkles, which the pose-space blend partly averages out; the overall appearance, silhouette, and texture of our exported asset remain close to both splat renderings, while ours is the only one of the three that is a textured mesh.



Figure 6: **Splat-based rendering versus our exported mesh asset on held-out poses.** Each row shows one held-out pose of a different subject. From left to right: the splat rendering of Animatable Gaussians, the splat rendering of our trained Gaussians before asset extraction, our exported textured-mesh asset, and the captured reference. The two left columns are not engine assets and serve as a reference for the gap between volumetric and mesh-based rendering.

D Additional Ablations

In Sec. 4.3 of the main paper, we ablate each of the three components of our pipeline with one variant and refer to the appendix for further variants. We provide them here in Tab. 4.

Mesh Geometry. Predicting raw per-vertex offsets instead of Laplacian coordinates, or removing $\mathcal{L}_{\text{lap}}$ from Eq. 6, produces high-frequency canonical-mesh artifacts that translate into temporally inconsistent asset maps.

Asset Maps. Dropping the displacement map, and then the normal map, measures how much surface detail each map carries into the mesh asset.

Pose-Space Blend. The joint-space distance of Eq. 8 outperforms naive axis-angle L2, especially around poses with large rotations. Increasing the number of neighbors K from 1 to 16 improves quality.

E Pose-Space Blending Details

In Sec. 3.5 of the main paper, we introduce the pose-space blend. Here we provide further details on its bandwidth and neighbor selection, which are referenced in App. B.

Bandwidth Selection. The kernel bandwidth σ in Eq. 9 controls how far a query pose can reach into the training database. We set it from the training set itself: for every training pose $\boldsymbol{\theta}_f$ we compute the distance $d(\boldsymbol{\theta}_f, \boldsymbol{\theta}_{f'})$ of Eq. 8 to its nearest other training pose, and take σ as the median of these F values. A bandwidth much smaller than this median collapses the blend onto the single nearest pose and reintroduces temporal popping between poses, as in the $K = 1$ setting of Tab. 4; a bandwidth much larger averages over unrelated poses and washes out pose-dependent wrinkles.

Table 4: **Additional ablations on held-out poses (AvatarReX, 2000-frame split).** Variants are grouped by the component they modify; each row changes one design choice of the full pipeline. The best result is highlighted in red.

Variant	PSNR↑	SSIM↑	LPIPS↓
Mesh geometry (Sec. 3.2)			
Raw vertex offsets (no Laplacian)	26.71	0.965	0.057
w/o Laplacian regularization	26.93	0.967	0.055
Asset maps (Sec. 3.4)			
Texture only	27.21	0.969	0.053
Texture + normal	27.23	0.969	0.053
Pose-space blend (Sec. 3.5)			
Axis-angle pose distance	26.38	0.967	0.055
$K = 1$	26.65	0.968	0.055
$K = 4$	27.17	0.969	0.054
Full (Ours)	27.24	0.969	0.053

Neighbor Truncation. The blend of Eqs. 9–10 uses only the $K = 16$ nearest training poses under $d(\cdot, \cdot)$, with weights normalized over this set. The full sum over all F training poses is unnecessary because the kernel decays quickly: with $K = 16$ the discarded tail carries less than 2% of the total weight on our training sequences. Tab. 4 shows that quality improves from $K = 1$ to $K = 16$. The nearest-neighbor query itself runs on the $N_J \times 3$ joint locations $\mathbf{J}(\boldsymbol{\theta})$ of Eq. 8 rather than on the asset maps, so it costs a fraction of a millisecond per query pose.

F Exported Asset Format

In Sec. 3.5 of the main paper, we describe the asset exported for a query pose, and in Sec. 4.2 we refer to this appendix for the export details of the mesh-asset baselines. Here we document the exact contents and file layout of our asset, so that the export stage can be reimplemented, and we describe how the baseline meshes were produced.

Contents. An exported asset for a query pose $\boldsymbol{\theta}^\star$ consists of four items: the posed mesh $\mathbf{v}^{\mathsf{P},\star}$ with the fixed template topology and the fixed UV layout; the blended texture map $\mathbf{T}^\star$; the blended tangent-space normal map $\mathbf{N}^\star$, renormalized after blending; and the blended scalar displacement map $\mathbf{D}^\star$ along the surface normal. Topology, UV coordinates, and skinning weights are shared across all poses of a subject and are written once, so a per-pose asset stores only vertex positions and the three asset maps.

Files. We write the mesh as OBJ with the shared UV layout, the texture as 8-bit sRGB PNG, and the normal and displacement maps as 16-bit PNG or 32-bit EXR depending on the target renderer, at the atlas resolution of 1024×1024. An animation shares the topology file across frames and stores only the per-pose vertex positions and maps.

Rendering. Because the asset is an ordinary textured mesh, no custom rasterizer is involved. We render with OpenGL for the quantitative evaluation, since it matches the training-time camera model exactly, and with Blender (Cycles renderer and Bullet physics) for the scene renderings in Fig. 1 and Sec. 4.4. The displacement map is consumed either as true displacement on a subdivided mesh or as a bump term, depending on the renderer; both paths are supported by standard engines and neither requires modification of the asset.

Baseline Mesh Export. For the *AG-Mesh* baseline of Sec. 4.2, given the Animatable Gaussians avatar at a query pose and a set of virtual cameras, we render the median depth and the foreground mask into every camera, fuse the masked depth into a TSDF volume with fixed voxel size, and extract a triangle mesh with Marching Cubes; the mesh is then decimated to approximately one hundred thousand triangles. The virtual cameras are 48 views on a sphere around the subject; the extracted mesh is UV-unwrapped with xatlas and textured by projecting the same virtual-camera renders into the atlas, weighted by view-to-normal alignment. Because the procedure is repeated for every query pose, AG-Mesh yields a different topology and UV layout per pose. For *DeMapGS + LBS* we use the public implementation with our SMPL-X template and UV unwrap, trained on a single frame per subject from the same training cameras as our method (14 on AvatarReX, 48

on ActorsHQ) and without its external normal supervision, and repose the result by linear blend skinning of the template with its texture, normal, and displacement maps held fixed. All baseline assets are rendered with the same OpenGL setup as ours.