Designing Animatable Stylized Characters with Multi-Layer 3D Gaussians

WENQING WANG, DisneyResearch|Studios, Switzerland and ETH Zürich, Switzerland
CLARA FERNANDEZ, DisneyResearch|Studios, Switzerland
YANG ZHANG, DisneyResearch|Studios, Switzerland
ABDELAZIZ DJELOUAH, DisneyResearch|Studios, Switzerland

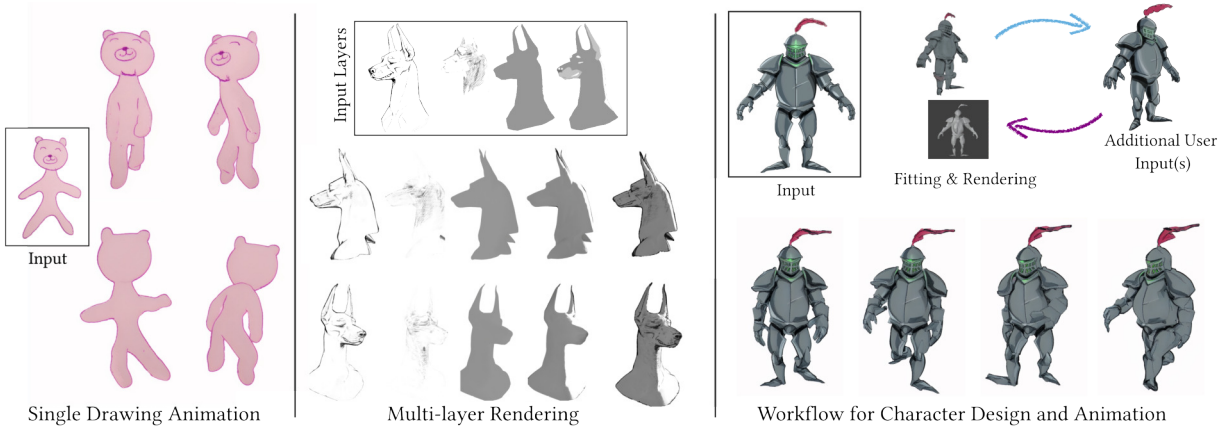


Fig. 1. We present a unified framework for generating animatable stylized characters from one or multiple input images. Our multi-layer mesh-anchored Gaussian representation supports layer-decomposed stylization and view-dependent effects such as line art and lighting. The system provides artists with fine-grained control to adjust and edit the different stylization layers from different viewpoints as needed.

Animatable stylized characters are central to animation pipelines, yet transforming a few hand-drawn references into a rigged, editable, and renderable asset remains labor-intensive. Existing generative methods tend to deviate from the original artist-provided designs, while single image based methods offer unclear paths for integration into artist workflows. We introduce a unified framework that produces fully rigged, animatable characters from a single image, while supporting additional references when available. The core representation couples a deformable mesh with a multi-layer Gaussian field, enabling layered rendering, stable articulation, and rich view-dependent appearance. Fitting is accelerated by generative priors in 3D and image space. Beyond single-image animation, the system supports an artist-in-the-loop workflow for assembling or extending a character sheet through iterative image edits that are straightforward to model and control in a Gaussian formulation. Our design allows for decoupled modules that learn complex appearance phenomena, such as line work and lighting. We report results that match or exceed prior approaches while better matching goals of alignment and controllability, for stylized 4D character synthesis.

Authors' Contact Information: Wenqing Wang, DisneyResearch|Studios, Zürich, Switzerland and ETH Zürich, Zürich, Switzerland, wangwen@student.ethz.ch; Clara Fernandez, DisneyResearch|Studios, Zürich, Switzerland, clara.fernandez@disneyresearch.com; Yang Zhang, DisneyResearch|Studios, Zürich, Switzerland, yang.zhang@disneyresearch.com; Abdelaziz Djelouah, DisneyResearch|Studios, Zürich, Switzerland, aziz.djelouah@disneyresearch.com.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
SA Conference Papers '26, Kuala Lumpur, Malaysia
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2842-6/2026/12
https://doi.org/10.1145/3829340.3842267

CCS Concepts: • **Computing methodologies** → **Point-based models**; **Non-photorealistic rendering**.

ACM Reference Format:
Wenqing Wang, Clara Fernandez, Yang Zhang, and Abdelaziz Djelouah. 2026. Designing Animatable Stylized Characters with Multi-Layer 3D Gaussians. In *SIGGRAPH Asia 2026 Conference Papers (SA Conference Papers '26), December 01–04, 2026, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3829340.3842267

1 Introduction

Creating animatable assets and characters is a key aspect of animation production (both 2D and 3D). Traditionally, this workflow begins with a character sheet containing one or multiple reference images. Based on these 2D designs, modelers create an initial 3D representation. This is followed by an iterative refinement process where artists and modelers constantly adjust the geometry and appearance until the model meets production standards. Once the modeling is complete, the asset is rigged with a digital skeleton and skinned to enable movement. Only after these labor-intensive steps is the character ready for animation. Such an effort is however less justified with stylized content, where the animation would not leverage the advantages of a physically precise 3D character (both in terms of simulation and rendering). Additionally, cheaply providing animatable characters would permit more artistic explorations. Unfortunately, bridging the gap between a 2D design and a dynamic, 4D (animated 3D) representation remains a significant challenge.

Several works have explored building 3D assets from drawings and images [Buchanan et al. 2013; Dvoroznak et al. 2020]. More recent methods use generative priors either in image space [Long et al. 2024; Shi et al. 2023] or 3D [Hunyuan3D et al. 2025; Xiang et al. 2025] and show impressive results. These however don't align exactly with the input images, which is key for an artist-driven workflow. Concurrent to our work, Yoon *et al.* [2026] designed a workflow to build a 3D character asset for 2D animation compensating misalignment in the input images and enforcing predefined color palettes, but these constraints would not be applicable to a large set of content and styles. In a different direction, several works can be used to animate a single hand-drawn image either through specifically designed workflows [Smith et al. 2023; Zhou et al. 2025, 2024] or by using conditioned video generation models [Hu 2024; Wang et al. 2025; Zhang et al. 2025]. It is however unclear how artists can make edits in such approaches.

We propose a unified framework that supports character animation from a single image with competitive results, while extending to complex artist-required features. The core part of our proposal is a multi-layer mesh-anchored Gaussian representation, which can be animated and supports layered rendering and view-dependent effects such as texture overlays, line art and lighting as shown in Figure 1. We use 3D and image-space priors to accelerate Gaussian and mesh fitting. Beyond single-image animation, our approach guides character sheet creation via interactive user edits (Fig. 1).

Our contributions are as follows:

- *A Unified 4D Character Synthesis Framework.* We present a system to generate fully rigged and animatable 3D characters from a single image, but also capable of integrating more 2D references.
- *Complex rendering and controls via the multi-layer mesh-anchored Gaussian representation.* We introduce decoupled modules that learn complex effects, such as texture overlays, line art and shading. Some of these view-dependent effects are transferable across diverse character assets and support real-time rendering.
- *Artist-in-the-loop workflow.* Our system provides artists with fine-grained control to adjust and edit the different stylization layers from different viewpoints as needed.

2 Related Work

Our work intersects diverse fields within computer graphics, and it is not possible to provide an exhaustive review for each. Instead, we focus our discussion on the three primary categories that are most relevant to our technical approach, with an emphasis on the works most relevant in terms of comparisons.

3D Reconstruction from Sketches and Drawings. There is a large body of work in the space of 3D reconstruction from sketches and drawings. Starting from hand-drawn inputs, several systems were developed focusing on creating 3D shapes [Igarashi et al. 1999; Nealen et al. 2007; Schmidt et al. 2007]. As deep learning matured, subsequent research utilizes neural networks to predict depth relations, flow fields, and surface normals to mitigate the inherent ambiguity of single-view sketches [Li et al. 2018]. Other approaches [Delanoy et al. 2018; Han et al. 2020; Lun et al. 2017] have considered the incorporation of additional sketches (typically front and side views). Early works [Buchanan et al. 2013; Dvoroznak et al. 2020] used several heuristics to generate a texture mesh that can be rigged and/or animated, starting from a single color input image. More recently, research has shifted toward leveraging priors from generative models. For example, image-conditioned multi-view models such as Zero-1-to-3 [Liu et al. 2023], MVDream [Shi et al. 2023] or Wonder3D [Long et al. 2024] generate multi-view images via 2D diffusion and reconstruct the 3D from them. It is possible to use generative models to directly predict the mesh from a set of input color images [Hunyuan3D et al. 2025; Xiang et al. 2025; Zhao et al. 2025a]. The results are impressive; however, the resulting meshes don't correspond exactly to the input image(s). Unless guided by specifically designed tools [Zhou et al. 2023], multi-view drawings or character sheets are not geometrically consistent. Weber *et al.* [2024] investigated using 3D Gaussian representations with specifically designed matching approach. In this space, the concurrent work of Yoon *et al.* [2026] tries to address similar challenges; however, the objectives are different. The proposed method compensates for the inconsistencies from the input drawings while still optimizing a single mesh representation. The rendering supports different layers but the design choices are limited to a specific style.

Deformable and Animatable 3DGS. Since the first works [Kerbl et al. 2023; Zwicker et al. 2001a,b] around 3D Gaussian Splatting (3DGS), the field has seen several evolutions in particular moving towards more controllable representation, such as tighter integration between Gaussian primitives and explicit mesh proxies [Guédon and Lepetit 2024a; Waczyńska et al. 2024]. SuGaR [Guédon and Lepetit 2024b] introduces a regularization term that encourages Gaussians to align with a coarse mesh. MILo [Guédon et al. 2025] aims at detailed surface reconstruction. This is also beneficial for dynamic tasks [Gao et al. 2024, 2025]. To bridge the gap between high-fidelity splatting and skeletal animation, several works have integrated 3DGS with traditional rigging techniques [Kwon et al. 2024; Liu et al. 2024]. However, these methods are primarily limited to human characters. RigGS [Yao et al. 2025] proposed a skeleton-aware, node-controlled Gaussian deformation. However, it still requires extensive multi-view video data for training. Our method builds upon the paradigm of anchoring Gaussians to a mesh proxy, specifically tailored for articulated character synthesis and artist-driven control. Our objective is not a precise 3D mesh reconstruction, rather, a controllable and animatable representation, to support a drawing driven character and asset creation. We in particular model effects such as view-dependent line art and stylized shading, while having relatively low data requirements. We can note that as we aim at stylized content, we adopt losses somewhat similar to recent 3D GS stylization works [Mei et al. 2024; Zhao et al. 2025b].

2D & 3D Character Animation. The animation of hand-drawn characters has progressed from image-space deformations to 3D-guided volumetric approaches. Classical techniques utilize As-Rigid-As-Possible (ARAP) deformation to animate 2D shapes [Hornung et al. 2007; Su et al. 2018; Takeo 2005]. The *Animated Drawings* framework [Smith et al. 2023] recently popularized the approach for diverse artistic styles. Guided by text-to-video prior, AniClipart [Wu et al. 2025b] uses a differentiable ARAP algorithm to optimize and

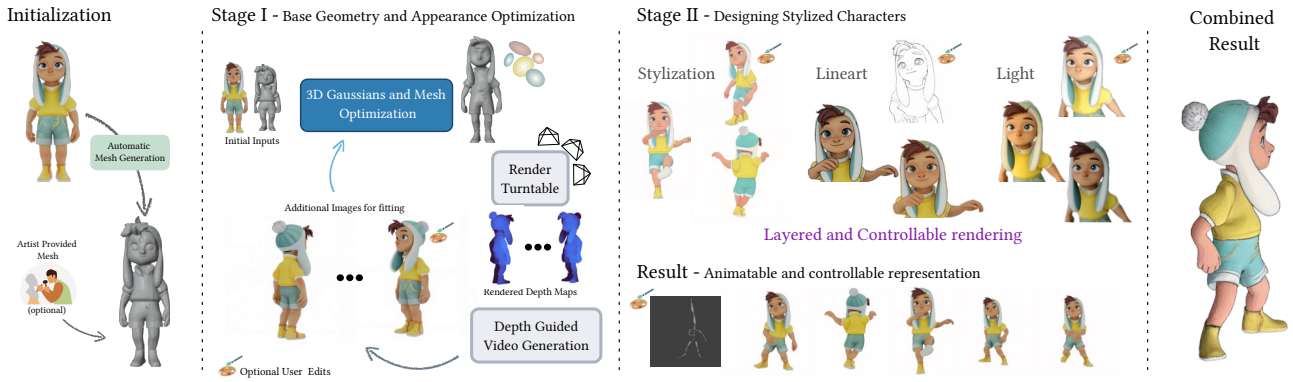


Fig. 2. Overview. Starting from an input image and an initialization mesh (generated automatically or provided by the artist), the pipeline consists of two stages: the first focuses on generating and refining the geometry and base appearance; The second can be used to further define stylization or a variety of static and view-dependent effects. Artists are in the loop throughout the process as they can adjust the mesh or the images at any moment. The result is an animatable and controllable representation.

warp a clipart to a new pose. To bridge the gap between 2D artistic expression and 3D geometric stability, *DrawingSpinUp* [Zhou et al. 2024] established a workflow by utilizing mesh reconstruction [Long et al. 2024] and specialized silhouette line removal to produce animations from single character drawings. In a different vein, generative models can be used for directly posing and animating characters [Hu 2024; Wang et al. 2025; Zhang et al. 2025]. While demonstrating impressive results, they have several limitations that render them challenging to use in artist workflows: large training dataset, no explicit editable representation , etc. Combining the advantage of both worlds, Zhou *et al.* [2025], construct an explicit 3D model for an input hand-drawn character and the generation is seen as an inpainting task guided by the rendered 3D. Despite the quality of the results from a single image, it is unclear how complex effects or additional images from different viewpoints can be integrated.

3 Overview

We introduce a framework for layered animatable character design, starting from one or several reference images. We use automatic mesh generation, and anchor a set of 3D Gaussians to the mesh. Then we fit the mesh and Gaussians jointly to align with the reference image(s). Following this initial fitting, we render a turntable of depth maps to serve as a condition for a video generation model, together with the original color images. By leveraging the video model's strong priors, we meaningfully extend color information to alternative viewpoints while ensuring geometric coherence through the rendered depth maps. The resulting frames are then integrated as additional input for the 3DGS fitting. Crucially, this framework allows for artistic intervention at multiple stages. For instance, an artist can provide a custom initial mesh (even if it does not match the image perfectly) which is particularly useful for transferring existing rigs or animation sequences to new characters. Furthermore, users can manually correct or edit the frames produced by the video model, allowing our framework to act as a flexible guide and support system for character design. The resulting mesh can be rigged using Mixamo [Adobe Inc. 2022] for humanoid characters or more generic methods such as RigAnything [Liu et al. 2025]. This rig serves as a geometric driver: as the mesh deforms with the skeletal pose, the anchored Gaussians follow the surface movement via fixed barycentric coordinates.

Our framework (Figure 2) consists of 2 stages. We model the base layer of the character in the first stage, and stylization or other controllable view-dependent effects in the second stage. By combining these features, users can produce complex, animatable characters without the need for traditional, labor-intensive 3D modeling.

4 Stage I: Base Geometry and Appearance Optimization

We use a mesh-anchored Gaussian representation to combine the flexibility of advanced appearance modeling with the practical advantages of an explicit mesh, such as standard rigging and animation. Notably, our framework bypasses the need for perfect geometry required by traditional workflows, accommodating approximate meshes without significant loss in quality.

4.1 Data Preparation and Generative Priors

To ensure best results, it is important to remove from the reference image(s) any view-dependent effects or lighting conditions that should not be baked into the representation, especially if these effects need to be controlled. If not available, other strategies can be adopted either through specialized modules such as contour removal or more generic image edit models [Wu et al. 2025a].

3D Mesh Preparation. We adopt TRELLIS [2025] to generate the base geometry. Alternatively, this can be provided by the artist or other mesh generation tools. We use standard mesh preprocessing operations available in any 3D modeling software to ensure proper vertex density and enforce manifold triangulation.

Generate Turntable via Video Synthesis. To minimize the manual effort of providing multiple reference images, we leverage generative video diffusion models to synthesize the missing viewpoints. Specifically, we use LTX-Video [HaCohen et al. 2026] to generate a 360-degree turntable video. We condition the generation on the reference image(s) and a depth video rendered from the initial fitted

mesh. This guidance ensures that the generated video remains spatially consistent with our geometric proxy. The resulting turntable sequence is treated as pseudo-ground truth to supervise the joint optimization of the mesh and Gaussian attributes.

4.2 Mesh-Anchored Gaussian Representation

Consistent with recent mesh-bound frameworks [Shao et al. 2024; Waczyńska et al. 2024], we anchor our 3D Gaussians directly to the surface of the triangle mesh. We begin by uniformly sampling N points on the surface. Each Gaussian g_i is defined by fixed barycentric coordinates (w_1, w_2, w_3) relative to a specific face $f = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$. This is well suited for animation, as the position of a Gaussian $\mathbf{p}_i$ at frame t is uniquely defined by the displaced vertices $\mathbf{v}(t)$ as:

$$\mathbf{p}_i(t) = w_1\mathbf{v}_1(t) + w_2\mathbf{v}_2(t) + w_3\mathbf{v}_3(t) \tag{1}$$

where $w_1, w_2, w_3 \in [0, 1]$ and $w_1 + w_2 + w_3 = 1$. This representation ties character appearance directly to underlying geometry, bypassing the need for complex, learned deformation fields.

Local Coordinate Frames and Rotations. To ensure Gaussians rotate naturally during animation, their orientation is strictly driven by the mesh's local tangent space. For each Gaussian, we construct a time-varying orthonormal frame $\{\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t)\}$ based on the triangle's current state. Let $\mathbf{n}_j(t)$ be the normal at vertex $\mathbf{v}_j(t)$. We interpolate a smooth normal at the Gaussian's barycentric position $\tilde{\mathbf{n}}(t) = w_1\mathbf{n}_1(t) + w_2\mathbf{n}_2(t) + w_3\mathbf{n}_3(t)$. Given a triangle edge $\mathbf{e}_1(t) = \mathbf{v}_2(t) - \mathbf{v}_1(t)$, the local axes are computed as:

$$\mathbf{z}(t) = \frac{\tilde{\mathbf{n}}(t)}{||\tilde{\mathbf{n}}(t)||},\ \mathbf{y}(t) = \frac{\mathbf{z}(t) \times \mathbf{e}_1(t)}{||\mathbf{z}(t) \times \mathbf{e}_1(t)||},\ \mathbf{x}(t) = \mathbf{y}(t) \times \mathbf{z}(t) \tag{2}$$

The resulting rotation matrix $\mathbf{R}_i(t) = [\mathbf{x}(t)|\mathbf{y}(t)|\mathbf{z}(t)]$ is converted into a quaternion $\mathbf{q}_i(t)$ for the splatting pipeline.

Dynamic Scale Compensation. To avoid "skin-cracking" or density loss, we use a scale compensation factor $\mathbf{s}_{\text{comp}}(t)$. For a given mesh triangle with vertices $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$, we define its local dimensions using its base edge length $L = \|\mathbf{v}_2 - \mathbf{v}_1\|$ and its perpendicular altitude H, measured from $\mathbf{v}_3$ to the base edge. At any given frame t, we adjust the Gaussian scales by comparing the current triangle dimensions (L, H) to their rest-pose reference $(L_{\text{ref}}, H_{\text{ref}})$:

$$s_x(t) = \frac{L(t)}{L_{\text{ref}}}, \quad s_y(t) = \frac{H(t)}{H_{\text{ref}}}. \tag{3}$$

The scale along the normal direction $(s_z(t))$ is inversely adjusted based on the surface expansion:

$$s_z(t) = \text{clamp}\left(\frac{1}{s_x(t) \cdot s_y(t)}, 0.5, 1.5\right) \tag{4}$$

$\mathbf{s}_{\text{comp}}(t)$ gives the final scale of the Gaussian. This is in line with related work in mesh-anchored Gaussian representations.

4.3 Multi-Layer Gaussians

We introduce a multi-layer Gaussian strategy to provide a richer volumetric "shell" around the character, which enables the support of complex stylization and view-dependent effects. At initialization, we distribute Gaussians uniformly across the mesh surface and create additional layers by displacing them along the vertex normals, both inward and outward (See Figure 3.a). All layers remain strictly anchored to the underlying mesh topology via barycentric coordinates. We keep the same approach of adjusting Gaussian's orientation and scale based on the mesh deformation. During stage I, the inner and surface layers are optimized, whereas the outer layers are kept fixed and completely transparent.

4.4 Joint 3DGS and Mesh Optimization

We employ a joint optimization strategy and backpropagate the rendering loss simultaneously to the Gaussian attributes and the mesh vertices. While the Gaussians learn surface colors and low-degree Spherical Harmonic (SH) coefficients, the mesh updates its vertex positions to better represent the character's geometry. We deliberately constrain the opacity of all Gaussians to 1.0 and fix their intrinsic scales. This forces the model to minimize the photometric error primarily by deforming the mesh vertices.

Reconstruction Loss. The primary supervision is derived from the provided reference(s) and the synthesized turntable frames. The reconstruction loss incorporates both color $\mathcal{L}_{\text{img}}$ and alpha-mask $\mathcal{L}_{\text{mask}}$ components:

$$\mathcal{L}_{rec} = \lambda_i \mathcal{L}_{img}(C_v, \hat{C}_v) + \lambda_m \mathcal{L}_{mask}(\alpha_v, \hat{\alpha}_v). \tag{5}$$

where C_v and α_v represent the rendered color and opacity, while $\hat{C}_v$ and $\hat{\alpha}_v$ are the corresponding (pseudo) ground truth values. $\mathcal{L}_{\text{img}}$ is a weighted combination of L_1 distance and a structural similarity term (D-SSIM). The mask loss $\mathcal{L}_{\text{mask}}$ is implemented as a mean squared error between the rendered alpha and the ground truth mask if available. We employ a multi-background training strategy, rendering the asset against random, white, and black backgrounds.

Style Consistency Loss. This loss ensures the appearance remains consistent across viewpoints where no (pseudo) ground truth is available. It consists of a style loss $\mathcal{L}_{style}$ [Mei et al. 2024], aligning output and target distributions in feature space, and a Semantic Color Consistency Loss, $\mathcal{L}_{cluster}$. This term acts as a structural prior, ensuring that Gaussians belonging to the same semantic region undergo synchronized stylistic updates.

The consistency loss $\mathcal{L}_{\text{cluster}}$ is structured in two phases: Initial Semantic Grouping and Dynamic Consistency Optimization. Prior to the stylization process, we employ the K-Means algorithm to partition the total N Gaussian primitives into K disjoint clusters $\{S_1, S_2, \ldots, S_K\}$ based on their initial base colors $\bar{\mathbf{c}}_i$ (distilled from the source mesh). These assignments remain fixed during optimization to serve as a stable spatial-semantic reference.

During the stylization phase, we compute the centroid $\boldsymbol{\mu}_k$ for each cluster in the current latent color space: $\boldsymbol{\mu}_k = \frac{1}{|S_k|}\sum_{j \in S_k} \mathbf{c}_j$, where $\mathbf{c}_j$ denotes the current color attribute of the j-th Gaussian. The consistency loss $\mathcal{L}_{cluster}$ is then defined as the mean squared error (MSE) between each Gaussian's color and its respective cluster centroid:

$$\mathcal{L}_{cluster} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i \in S_k} \left|\mathbf{c}_i - \boldsymbol{\mu}_k\right|^2 \tag{6}$$

Our overall style consistency loss is a weighted combination of the VGG-based style loss and the proposed cluster regularization:

$$\mathcal{L}_{cons} = \lambda_{style}\mathcal{L}_{style} + \lambda_{cluster}\mathcal{L}_{cluster} \quad (7)$$

Geometric Regularization. To maintain an accurate and smooth mesh during vertex deformation, we apply geometric constraints on the mesh's vertex positions $\mathcal{V}$: (a) *Normal Consistency* [Ravi et al. 2020]: minimizes the angular difference between neighboring face normals to encourage surface smoothness; (b) *ASAP Loss [Liu et al. 2008]:* to regularize the deformation process, while less strictly penalizing change in edge length (e.g., ARAP loss [Igarashi et al. 2005]); and (c) *Edge Length Regularization [Mescheder et al. 2019]:* to prevent extreme stretching of individual triangles and maintain a well-conditioned mesh topology throughout the deformation.

The final objective function is a weighted sum of all components:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{cons} + \lambda_{norm}\mathcal{L}_{norm} + \lambda_{asap}\mathcal{L}_{asap} + \lambda_{edge}\mathcal{L}_{edge} \quad (8)$$

5 Stage II: Designing Stylized Characters

In a second stage, we model the outer layers of the Gaussians for both static stylization and more complex and controllable view-dependent effects, while keeping the mesh fixed.

5.1 Layer-Decomposed Stylization

Artists paint stylization effects in separate layers to isolate and refine different aesthetic components while keeping the underlying design intact. Similarly, our multi-layer Gaussians initialization enables fine-grained stylization by assigning distinct roles to different Gaussian layers. Specifically, the inner and surface layers are optimized to capture the character's base appearance and perform reference-based stylization. The outer layers instead are dedicated to static stylistic edits, like texture overlays, and some view-dependent effects, such as line art. Other view-dependent effects such as lighting, can be selectively or collectively applied across these layers.

To model view-dependent effects, we use a lightweight Multi-Layer Perceptron (MLP). Conditioned on the desired controls (e.g. camera parameters, Gaussian attributes and local geometric features), the MLP outputs per-primitive information used to modulate Gaussian attributes, such as color, opacity, and scale. We supervise this process using image pairs representing the presence and absence of the targeted effect. Once trained, the MLPs can be applied to other characters, enabling artists to apply consistent effects across multiple assets without retraining.

Multiple stylization layers can be trained independently and applied jointly, with or without the base layer, allowing for a highly flexible and modular stylization workflow. By leveraging this multi-layer structure, we achieve diverse stylistic expressions while maintaining high geometric consistency across various poses and dynamic sequences. The complete pipeline is illustrated in Figure 3.

For all stylization effects, we assume that the artist provides one or more references, which our framework extends to the entire character geometry. Similar to the first optimization stage, we synthesize the missing stylized viewpoints with LTX-Video [HaCohen et al. 2026] to serve as pseudo-ground truth. Next we present how the proposed representation models different static and view-dependent effects, and introduce additional losses that facilitate the stylization process.

5.1.1 *Static effects.*

Reference-based Stylization. For reference-based stylization, we assume that the artist provides a reference(s) with a new final look for the character, rather than different stylization layers. Therefore, to learn the new style, we align the Gaussian primitives from the first stage (inner and surface layers) with the new target artistic distribution by optimizing $\mathcal{L}_{rec}$ and $\mathcal{L}_{cons}$. This process enables the model to inherit the specific style, color palettes, and material properties while preserving the character's geometric identity. The outer layers are kept transparent.

Texture Overlays. Different from reference-based stylization, we use the outer layers Gaussians to learn a texture overlay, and keep the inner and surface layers frozen. For the same outer layer Gaussians, we can learn different texture overlays, i.e. different Gaussian attributes, that get composited together in the desired order to produce the final stylized character. For each texture overlay, we optimize the Gaussian primitives (color, opacity, scale) using $\mathcal{L}_{rec}$ together with a dual-term regularization loss $\mathcal{L}_{act}$ for the opacity values.

$$L_{act} = \lambda_s L_{sparse} + \lambda_b L_{binary} \quad (9)$$

where L_{sparse} is a sparsity loss, defined as an L_1 penalty to encourage a parsimonious selection of Gaussians for each texture overlay, and L_{binary} is the binary entropy loss to ensure a sharp transition between texture overlays. This pushes the opacity values towards the discrete limits of $\{0, 1\}$, creating a clear boundary between stylistic layers.

5.1.2 *View-dependent effects.*

View-dependent effects are modeled through a lightweight Multi-Layer Perceptron (MLP).

Line Art. View direction and surface orientation are critical for line art effects. Therefore, we condition the line art MLP on the angle between the Gaussian normal and the camera view direction θ, the local surface normal $\mathbf{n}$, the layer depth d (originating from our multi-layer initialization) and the view direction $\mathbf{v}$. The resulting input feature vector $f_{lineart}$ for each Gaussian is:

$$f_{lineart} = [\theta, \mathbf{n}, d, \mathbf{v}] \quad (10)$$

The MLP outputs a per-primitive activation via a sigmoid function, to determine which outer layers Gaussians are influenced by the effect. By training on image pairs with and without line art, our model learns a mapping that generates stylized contours across all poses (see Figure 2). The user can adjust the color and scale (line thickness) of the Gaussians without retraining the core mapping. For line art, we limit the MLP to only influence the outer layers Gaussians, and optimize the MLP weights using $\mathcal{L}_{rec}$ together with the same dual-term regularization loss $\mathcal{L}_{act}$ for the activation values.

Lighting. A similar approach can be adopted for adjusting character shading based on light information. The lighting MLP receives a feature vector f_{light} for each Gaussian, defined as:

$$f_{light} = [\mathbf{p}, \mathbf{n}, \mathbf{l}, \mathbf{v}, s] \quad (11)$$

where $\mathbf{p}$ and $\mathbf{n}$ are the position and normal of the Gaussian, $\mathbf{l}$ and $\mathbf{v}$ are the normalized directions to the light source and camera respectively. For simplicity, we focus on a directional plane light

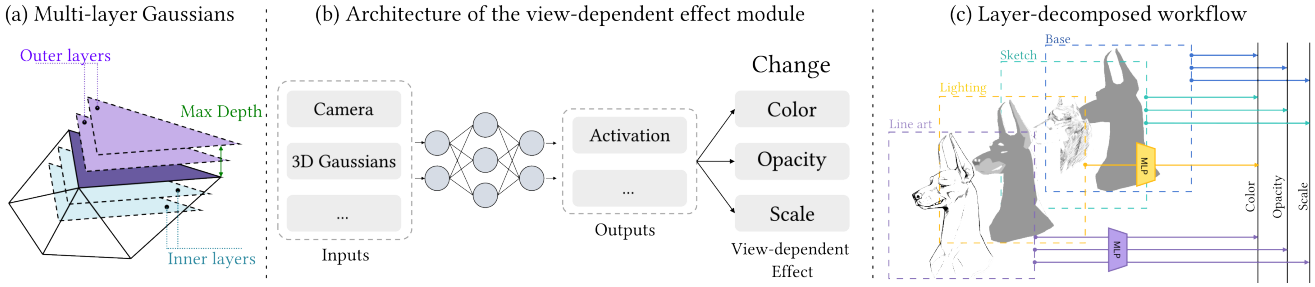


Fig. 3. Multi-layer Stylization. We introduce a multi-layer Gaussian representation as a rich volumetric "shell" around the character. This allows artists to control complex texture stylized overlays and view-dependent effects in different layers. View-dependent effects such as line art or lighting are modeled through an MLP and are transferable across characters. Multiple stylization layers can be trained independently and applied jointly.

source (analogous to an area light in Blender) and compute a shadow feature s via a differentiable orthographic depth-testing module, consistent with the parallel-ray nature of a plane light. We first render a global depth map from the perspective of the light source. For each Gaussian, we project its mean position into the light's clip space and compare its depth ($dist_z$) against the sampled depth from the light-space depth map (d_{light}). The shadow feature is determined via a sigmoid-based threshold:

$$s = \sigma\left(k^3 \cdot (dist_z - d_{light} - \epsilon)\right) \tag{12}$$

where k controls the penumbra transition and ϵ prevents self-shadowing artifacts. This results in a value where $s \approx 1$ indicates the Gaussian is in shadow.

The MLP predicts diffuse (k_d) and specular (k_s) coefficients to modulate the light source $\mathbf{L}$, (defined by its color $\mathbf{C}_L$ and intensity P_L). The final color $\mathbf{c}_{out}$ is calculated as:

$$\mathbf{c}_{out} = (\mathbf{c}_{base} \cdot k_d \cdot \mathbf{L}) + (k_s \cdot \mathbf{L} \cdot I_{max}) \tag{13}$$

where $\mathbf{c}_{base}$ represents the pre-trained base color of the Gaussian and $I_{max} = 4.0$ is a scalar multiplier that controls the maximum dynamic range of the specular highlights. Using this parametrization, the model learns to represent complex light-material interactions. This decoupled structure allows for high-performance interactivity. Parameters such as light color $\mathbf{C}_L$, intensity P_L, light direction $\mathbf{l}$, and shadow sharpness k can be modified in real-time without any retraining. Only the reconstruction loss $\mathcal{L}_{rec}$ is used to optimize the Lighting MLP weights.

5.2 Image Post-processing

To mitigate residual artifacts common in 3D Gaussian Splatting, we incorporate the state-of-the-art single-step diffusion refinement model Difix3D [Wu et al. 2025c]. We utilize the original 2D character sheet as a consistent style reference, allowing the diffusion prior to enhance high-frequency details and sharpen stylized boundaries. This post-processing step increases the visual fidelity of the final output while preserving the character's original design intent.

6 Experiments

6.1 Implementation Details

Framework and Hardware. Our framework is implemented in PyTorch. All experiments, including training and final inference, were conducted on a single NVIDIA GeForce RTX 4090 GPU.

Model Representation and Architecture. For multi-layer Gaussian initialization, we sample Gaussians from seven layers: one central layer on the mesh surface, three layers displaced internally along the surface normals, and three layers displaced externally. The density of Gaussians in each layer is scaled proportionally to the local surface area of the mesh. To maintain computational efficiency and real-time rendering performance, we cap the total primitive count at 1.5 million Gaussians per character. For the view-dependent effects, we employ MLPs that operate independently on the feature vectors of each Gaussian. The hidden space is set to 64 channels, while the input dimensions depend on the specific use case.

Training Details and hyperparameters. We use the Adam optimizer, with learning rates of 0.0005 for mesh vertices, 0.0025 for colors, 0.005 for scales, and 0.05 for opacities. The number of color clusters is set to 32. Mesh refinement takes between 2K to 20K steps depending on the desired quality, while each static/view-dependent stylization is trained for 10K steps. Loss weights during training are: $\lambda_{img} = 10$, $\lambda_{mask} = 10$, $\lambda_{style} = 0.2$, $\lambda_{cluster} = 10$, $\lambda_{norm} = 1$, $\lambda_{asap} = 100$, and $\lambda_{edge} = 2$. Each view-dependent effect is trained on 400 paired images rendered from an existing artist-built asset. Once trained, the MLP architecture enables real-time inference on entirely unseen characters. All evaluated results across different characters are generated from this single, unified set of MLPs without any per-case optimization. The sole exception is the triceratops character, where we fine-tuned our pre-trained MLP on just two artist-provided images to capture the specific lighting style.

Training and Inference Time. The process spans two distinct stages: the initial optimization of geometry and appearance, and the subsequent stylized rendering including view-dependent effects. Depending on the desired quality of the geometry and the number of input reference images, the mesh refinement takes from 10 minutes to an hour, and the appearance reconstruction takes 15 minutes. In the second stage, each layer-decomposed stylization effect

Table 1. Mesh refinement ablation study. Results are averaged across the entire *Synthetic 3D Models* set. Bold indicates the best performance.

	CD (10^{-3}) ↓	F-Score (%) ↑	Precision (%) ↑	Recall (%)↑	NC ↑
TRELLIS	0.4385	60.3853	60.9950	59.9267	0.8443
w/o norm	**0.1426**	86.7330	90.4010	83.7023	0.8139
w/o ASAP	0.1553	88.3258	94.4003	83.3317	0.9034
ours	0.1517	**88.6621**	**94.6967**	**83.7310**	**0.9076**

Table 2. View-dependent effect training ablation study. Results are averaged across 400 random camera angles. Bold indicates the best performance.

Number of training images	PSNR(↑)	LPIPS(↓)	tLPIPS(↓)	Chamfer distance px(↓)
4	24.6449	0.1099	0.0425	2.8920
40	25.0791	0.0811	0.0159	1.4976
400	**25.2089**	**0.0792**	**0.0134**	**1.4648**

Table 3. Quality degradation (LPIPS↓) compared to ground truth. The angle indicates the deviation from the training view.

Method	0°	45°	90°	135°	180°	225°	270°	315°
TRELLIS	0.1331	0.1730	0.2461	0.2311	0.1951	0.2333	0.2362	0.1842
ModelSheetTo3D	0.2166	0.2477	0.2529	0.2503	0.2385	0.2668	0.2556	0.2420
LTX (w. TRELLIS depth)	**0.0083**	0.1683	0.2455	0.1816	0.1636	0.2352	0.2514	0.1473
Ours	0.0184	**0.1593**	**0.2219**	**0.1786**	**0.1191**	**0.1975**	**0.2229**	**0.1314**

requires 10 minutes of training time. Inference for rendering the view-dependent effects is real-time.

6.2 Datasets

We perform experiments on a variety of datasets. First, *Hand-drawn Illustrations* from the Amateur Drawing Dataset [Smith et al. 2023]. This set is used to evaluate and compare against methods animating a character from a single image. Second, we use *Synthetic 3D Models* which is needed to quantitatively evaluate geometric reconstruction accuracy and test our method on more complex characters. We build this set using both in-house models and models from Sketchfab [Sketchfab Inc. 2026]. Finally, we use artist-made *multi-layer drawings*, to experiment with layered rendering and our artist-in-the-loop concept for character design.

6.3 Ablation Study

Initial Fitting and losses. Our initial fitting process is analogous to several works in the field of mesh-anchored Gaussians [Guédon and Lepetit 2024b; Tobiasz et al. 2025; Waczyńska et al. 2024]. We also note that obtaining a more precise mesh is beneficial, however we are more interested in being able to render images. The main benefit of the mesh is for artist control and animation. Mesh refinement results are quantitatively assessed on the *Synthetic 3D Models* using Chamfer Distance (CD), F-Score, Precision, Recall, and Normal Consistency (NC). Following standard 3D reconstruction evaluation protocols, all meshes are normalized to a unit sphere. As shown in Table 1, the initial mesh generated by TRELLIS serves as our baseline. We then compare this against two ablation configurations: "w/o norm" (removing $\mathcal{L}_{norm}$) and "w/o ASAP" (removing $\mathcal{L}_{asap}$).

Advantage of Using Video Generative Models. Using the generative video model to synthesize additional views for 3DGS fitting largely improves the quality of the rendering. Figure 4 shows for 1 and 4 input views, how the resulting animation behaves. Using a single image, there are obvious rendering errors, and using a post-processing step [Wu et al. 2025c] is not sufficient to fix the errors. Using the additional views generated by the video model clearly helps. The additional image processing is most useful in this case, as it helps fix smaller issues.

MLP Training. We train the line art MLP using varying numbers of views from a single character. In Table 2, we evaluate its generalization on 400 randomly sampled views of another character for which ground truth is available. Even with only 4 training views, the MLP captures the underlying view-dependent behavior and successfully transfers it to the unseen character.

6.4 Comparisons

Character from Single Image. For animation from a single image, we compare against *DrawingSpinUp* [2024], *AnimateAnyone* [2024] and LTX-2.3 [HaCohen et al. 2026]. These methods cover the range of skeletal and diffusion based animation. Unfortunately we could not compare against a recent method [Zhou et al. 2025] as the model was not publicly available We used OpenPose as guiding signal for *AnimateAnyone* [Hu 2024]. This conditioning was unsuccessful for LTX-2.3, so instead we used depth maps produced by our mesh as condition. Results are illustrated in Figure 6. Results are of comparable quality to *DrawingSpinUp* in the simpler drawing style, but our method seems to perform better on the characters from the *Synthetic 3D Models* set. *AnimateAnyone* is not adapted for this task without any fine-tuning. Despite using the higher quality depth map from our mesh, the video diffusion model would still deviate from the model as the motion becomes more and more complex. More importantly, our method is able to address a larger set of applications as demonstrated next. In supplemental material we provide video results and compare against *DrawingSpinUp*.

Quality degradation. Using only the front view as input, we measure the prediction degradation (LPIPS↓) of different methods versus the ground truth (See Table 3.) On the input viewpoint (0°), LTX is reproducing the image and quality loss is due to the VAE. Our method produces significantly better results on all the viewpoints, at the cost of a slight difference with the input view.

Character from Multiple Images. Our method can be extended to using multiple input images. For this we compare against TRELLIS [Xiang et al. 2025] and ModelSheetTo3D [Yoon et al. 2026]. We use an example from their dataset, a model from our *Synthetic 3D Models* set and an artist made drawing. In the case of the artist made drawing, we have access to the different layers and we support modeling and rendering them. In all cases our method is able to produce the best results. The exact views and additional viewpoints are provided in supplementary material.

To quantitatively evaluate our method against the baselines, we use four 3D characters to render ground truth turntable images. We provide 4 input views for each character. LTX is additionally guided by TRELLIS depth maps. As shown in Table 4, LTX is a strong generator, but our mesh-anchored optimization yields closer GT alignment, suggesting our method corrects inconsistent video-generated supervision rather than amplifying it.

Table 4. Quantitative comparison on the *Synthetic 3D Models* set.

Method	PSNR(↑)	SSIM(↑)	LPIPS(↓)	CLIP(↑)	tLPIPS(↓)
TRELLIS	18.9448	0.8843	0.1140	0.9190	0.0051
ModelSheetTo3D	14.8611	0.8532	0.2322	0.9115	0.0094
LTX	22.5696	0.9190	0.0405	0.9585	0.0062
Ours	**23.8219**	**0.9346**	**0.0337**	**0.9687**	**0.0035**

Table 5. User preference for animation and stylization tasks (ours vs others). The number of answers is in parentheses.

Task	Temp.Consistency	Faithfulness to Ref	Overall Quality
All tasks	81.37% (559/687)	82.97% (570/687)	80.49% (553/687)
Animation	82.37% (285/346)	88.15% (305/346)	84.39% (292/346)
Stylization	80.35% (274/341)	77.71% (265/341)	76.54% (261/341)

Table 6. User preference for our method against each baseline, averaged for the two tasks. The number of answers is in parentheses.

Baseline	Temp.Consistency	Faithfulness to Ref	Overall Quality
AnimateAnyone	100.00% (113/113)	100.00% (113/113)	100.00% (113/113)
DrawingSpinUp	72.88% (86/118)	75.42% (89/118)	76.27% (90/118)
LTX	74.78% (86/115)	89.57% (103/115)	77.39% (89/115)

Table 7. Run time efficiency analysis.

Method	Preparation time (min)	Inference time per frame (sec)
AnimateAnyone	5	3.3
LTX	3 - 60	0.7
ModelSheetTo3D	5	5.3
DrawingSpinUp	25 - 30	0.3
Ours	25 - 60	0.05 w/o Difix, otherwise 2.5

User Study. We conducted a two-alternative forced choice user study with 26 participants (including 4 artists) comparing our animation and stylization results against existing methods. The user study consisted of two tasks: (1) Animation: with 5 characters and different animations. (2) Stylization: with 3 characters and 5 styles. We evaluated user preference (ours vs others) on temporal consistency, faithfulness to reference, and overall quality. The results are summarized in Table 5 and Table 6. Most users preferred our method over the baselines, indicating that our approach is more effective in maintaining temporal consistency, faithfulness to reference, and overall quality in both animation and stylization tasks.

Runtime efficiency. As the workflows largely vary across methods, we summarize this into 2 parts: preparation time and inference. The preparation time includes mesh generation, fitting, any appearance fine-tuning or control signal extraction. The inference time represents the time to render an image from a control signal. On the inference part we note that our splatting-only rendering is real-time, which is needed for animators. The final rendering with post-processing can be slower. This division maps to traditional animation pipelines, which utilize rapid viewport rendering (e.g., OpenGL) during the authoring phase, followed by slow high-fidelity ray tracing for the final output.

6.5 Applications

The proposed approach has several advantages. Figure 5 shows how a line art MLP trained using paired images with and without line art, can be applied to other characters and even be modified in terms of color and thickness in real-time. Similarly, Figure 8 illustrates how a lighting MLP can be trained on images with and without lighting to shade the Gaussians, allowing artists to adjust the light direction or color in real-time. For character stylization, we update the base layers of the Multi-layer Gaussians. Figure 9 shows results for 2 different characters from the *Synthetic 3D Models* set.

The most interesting application is however to leverage the proposed workflow to design stylized characters, where 3D modeling is not a requirement. This can be beneficial in the early ideation phase or for strongly stylized characters. Figure 11 shows an example of this workflow that we tested with 2 artists.

Workflow Protocol. Artists draw a character reference with different layers, taking 10 to 60 minutes depending on complexity. Our system then generates a 3D stylized asset within 60 minutes (depending on the number of stylization layers). Artists inspect novel angles and animations, select an unsatisfactory viewpoint and/or pose, render its layers, and correct them. If only appearance fixes are needed, our system can improve the character within seconds. Otherwise, if the artist would like to correct the geometry, it would take longer depending on the complexity of the geometry fix. While quality drops when navigating to viewpoints significantly distant from the initial input, artists can easily integrate new drawings for those angles to progressively refine the 3D model.

Advantages and Artist Feedback. Both artists believe that focusing on drawings and skipping 3D modeling is highly beneficial for creating stylized characters that utilize traditional 2D visual language. While traditional workflows require alignment lines to maintain proportions, our tool ensures geometric consistency by anchoring corrections to the predicted 3D structure. Current workflows for achieving 2D stylized characters require 3D modeling and complex shaders, which typically take a few days according to our interviews. Our workflow bypasses this complexity and is significantly faster, making it ideal for rapid look development and prototyping. Finally, other generative-assisted baselines show worse performance and/or lack of control. Currently, the primary obstacle remains output quality. For future development, both artists recommend focusing on improving quality and control, addressing issues like noisiness, and ensuring the tool is highly responsive to edits.

7 Discussion

We present a novel framework for generating animatable stylized 4D characters from sparse 2D references, which achieves superior performance in both quantitative metrics and qualitative assessment. Our layer-decomposed stylization provides artists with flexible control for character stylization and design. Artists' feedback is encouraging and future work will focus on addressing current limitations. This includes (1) inaccuracies in the mesh proxy (due to occlusion and complex structures), which doesn't capture the correct topology (Figure 10), (2) Strongly inconsistent references or (3) Highly unique or heavily stylized inputs (Figure 11 triceratops) that deviate from the video model's training distribution.

Acknowledgments

Special thanks to the artists Violaine Fayolle and Doriano van Essen for creating high-quality 3D assets and 2D paintings for our work. They also contributed to the artist-in-the-loop evaluation by providing appearance fixes and insightful feedback.

References

Adobe Inc. 2022. Mixamo. https://www.mixamo.com.

Philip Buchanan, Ramakrishnan Mukundan, and Michael Doggett. 2013. Automatic single-view character model reconstruction. In *Proceedings of the international symposium on sketch-based interfaces and modeling*. 5–14.

Johanna Delanoy, Mathieu Aubry, Phillip Isola, et al. 2018. 3d sketching using multi-view deep volumetric prediction. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–22.

Marek Dvoroznak, Daniel Sykora, Cassidy Curtis, et al. 2020. Monster mash: a single-view approach to casual 3D modeling and animation. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–12.

Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. 2024. Mesh-based gaussian splatting for real-time large-scale deformation. *arXiv preprint arXiv:2402.04796* (2024).

Xiangjun Gao, Xiaoyu Li, Yiyu Zhuang, Qi Zhang, Wenbo Hu, Chaopeng Zhang, Yao Yao, Ying Shan, and Long Quan. 2025. Mani-gs: Gaussian splatting manipulation with triangular mesh. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 21392–21402.

Antoine Guédon, Diego Gomez, Nissim Maruani, Bingchen Gong, George Drettakis, and Maks Ovsjanikov. 2025. Milo: Mesh-in-the-loop gaussian splatting for detailed and efficient surface reconstruction. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–15.

Antoine Guédon and Vincent Lepetit. 2024a. Gaussian frosting: Editable complex radiance fields with real-time rendering. In *European conference on computer vision*. Springer, 413–430.

Antoine Guédon and Vincent Lepetit. 2024b. 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*. 5354–5363.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, et al. 2026. LTX-2: Efficient Joint Audio-Visual Foundation Model. *arXiv preprint arXiv:2601.03233* (2026).

Zhizhong Han, Baorui Ma, Yu-Shen Liu, and Matthias Zwicker. 2020. Reconstructing 3D Shapes From Multiple Sketches Using Direct Shape Optimization. *IEEE Transactions on Image Processing* 29, 8721–8734. doi:10.1109/TIP.2020.3018865

Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. 2007. Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics (ToG)* 26, 1 (2007), 1–es.

Li Hu. 2024. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8153–8163.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. 2025. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442* (2025).

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 99)*. 409–416.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Youngjoong Kwon, Baole Fang, Yixing Lu, Haoye Dong, Cheng Zhang, Francisco Vicente Carrasco, Albert Mosella-Montoro, Jianjin Xu, Shingo Takagi, Daeil Kim, et al. 2024. Generalizable human Gaussians for sparse view synthesis. In *European Conference on Computer Vision*. Springer, 451–468.

Changjian Li, Hao Pan, Yang Liu, et al. 2018. Robust flow-guided neural prediction for sketch-based freeform surface modeling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–12.

Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. 2025. Riganything: Template-free autoregressive rigging for diverse 3d assets. *ACM Transactions on Graphics (TOG)* 44, 4 (2025), 1–12.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. 2008. A local/global approach to mesh parameterization. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 1495–1504.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. 2023. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9298–9309.

Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. 2024. Humangaussian: Text-driven 3d human generation with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6646–6657.

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. 2024. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9970–9980.

Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, et al. 2017. 3d shape reconstruction from sketches via multi-view convolutional networks. In *2017 International Conference on 3D Vision (3DV)*. IEEE, 67–77.

Yiqun Mei, Jiacong Xu, and Vishal M Patel. 2024. ReGS: Reference-based controllable scene stylization with gaussian splatting. *Advances in Neural Information Processing Systems* 37 (2024), 4035–4061.

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

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. Fibermesh: designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 papers*. 41–es.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501* (2020).

Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A Jorge. 2007. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses*. 43–es.

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. 2024. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1606–1616.

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. 2023. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512* (2023).

Sketchfab Inc. 2026. Sketchfab. https://sketchfab.com/.

Harrison Jesse Smith, Qingyuan Zheng, Yifei Li, Somya Jain, and Jessica K Hodgins. 2023. A method for animating children's drawings of the human figure. *ACM Transactions on Graphics* 42, 3 (2023), 1–15.

Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. 2018. Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 chi conference on human factors in computing systems*. 1–12.

Igarashi Takeo. 2005. As-rigid-as-possible shape manipulation. *ACM Transactions on Computer Graphics* (2005).

Rafał Tobiasz, Grzegorz Wilczyński, Marcin Mazur, Sławomir Tadeja, and Przemysław Spurek. 2025. MeshSplats: mesh-based rendering with gaussian splatting initialization. *arXiv preprint arXiv:2502.07754* (2025).

Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. 2024. Games: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459* (2024).

Xiang Wang, Shiwei Zhang, Changxin Gao, Jiayu Wang, Xiaoqiang Zhou, Yingya Zhang, Luxin Yan, and Nong Sang. 2025. Unianimate: Taming unified video diffusion models for consistent human image animation. *Science China Information Sciences* 68, 10 (2025), 200103.

Ethan Weber, Riley Peterlinz, Rohan Mathur, Frederik Warburg, Alexei A Efros, and Angjoo Kanazawa. 2024. Toon3D: Seeing Cartoons from New Perspectives. *arXiv preprint arXiv:2405.10320* (2024).

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. 2025a. Qwen-image technical report. *arXiv preprint arXiv:2508.02324* (2025).

Jay Zhangjie Wu, Yuxuan Zhang, Haithem Turki, Xuanchi Ren, Jun Gao, Mike Zheng Shou, Sanja Fidler, Zan Gojcic, and Huan Ling. 2025c. Difix3d+: Improving 3d reconstructions with single-step diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 26024–26035.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. 2025b. Aniclipart: Clipart animation with text-to-video priors. *International Journal of Computer Vision* 133, 6 (2025), 3149–3165.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. 2025. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 21469–21480.

Yuxin Yao, Zhi Deng, and Junhui Hou. 2025. Riggs: Rigging of 3d Gaussians for modeling articulated objects in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5592–5601.

Hyejeong Yoon, Wonjong Jang, Yoonha Hwang, and Seungyong Lee. 2026. 3D Character Reconstruction from Hand-drawn Model Sheets. In *Computer Graphics Forum*. Wiley Online Library, e70323.

Jiaxu Zhang, Xianfang Zeng, Xin Chen, Wei Zuo, Gang Yu, and Zhigang Tu. 2025. Mikudance: Animating character art with mixed motion dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19689–19699.

Yian Zhao, Rushi Ye, Ruochong Zheng, Zesen Cheng, Chaoran Feng, Jiashu Yang, Pengchong Qiao, Chang Liu, and Jie Chen. 2025b. Tune-Your-Style: Intensity-tunable 3D Style Transfer with Gaussian Splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19032–19042.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. 2025a. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202* (2025).

Jie Zhou, Zhongjin Luo, Qian Yu, Xiaoguang Han, and Hongbo Fu. 2023. GA-Sketching: Shape Modeling from Multi-View Sketching with Geometry-Aligned Deep Implicit Functions. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, e14948.

Jie Zhou, Linzi Qu, Miu-Ling Lam, and Hongbo Fu. 2025. From Rigging to Waving: 3D-Guided Diffusion for Natural Animation of Hand-Drawn Characters. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–11.

Jie Zhou, Chufeng Xiao, Miu-Ling Lam, and Hongbo Fu. 2024. Drawingspinup: 3d animation from single character drawings. In *SIGGRAPH Asia 2024 Conference Papers*. 1–10.

M. Zwicker, H. Pfister, J. van Baar, and M. Gross. 2001a. EWA volume splatting. In *Proceedings Visualization, 2001. VIS '01.* 29–538. doi:10.1109/VISUAL.2001.964490

Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. 2001b. Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 371–378. doi:10.1145/383259.383300

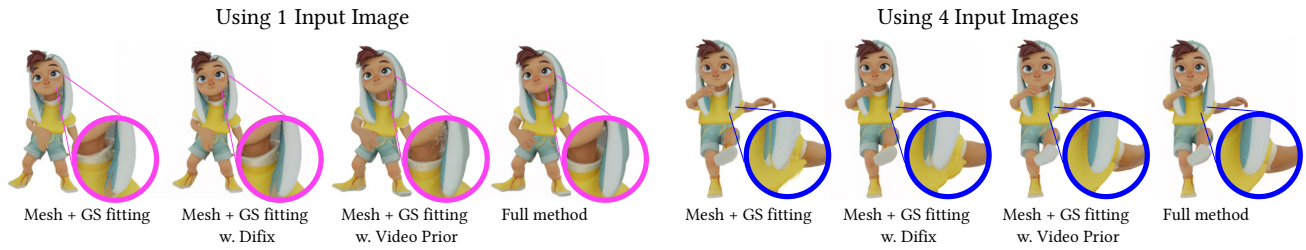


Fig. 4. Ablation Views and Generative Priors. Using the generative video model to synthesize additional views for 3DGS fitting largely improves the quality of the rendering. Using images only for fitting, there are obvious rendering errors. Using a post-processing step Difix [Wu et al. 2025c] is not sufficient.

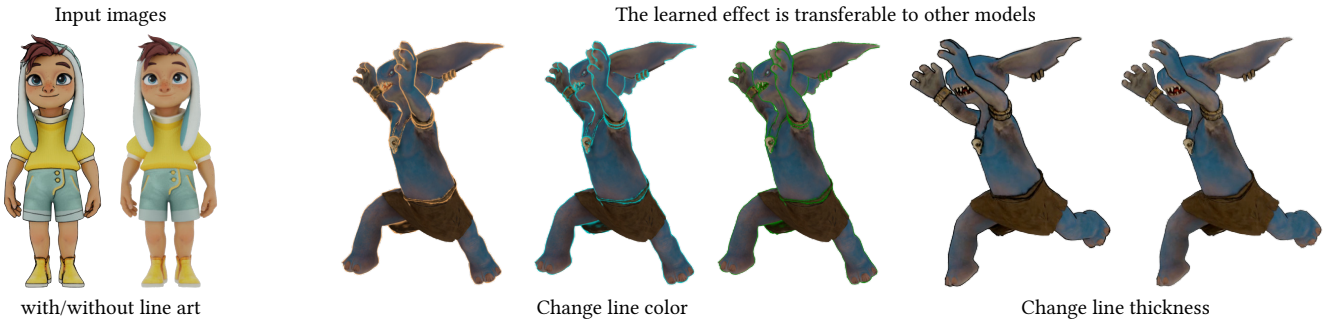


Fig. 5. Line art. The input images are provided with and without line art. They are used to train an MLP to be applied on the Multi-layer Gaussians. Once trained, it can be applied to other characters. Both color and thickness can be controlled.



Fig. 6. Single drawing animation: When starting from a single image, our approach achieves comparable or better results than existing methods.



Fig. 7. Modeling using a character sheet. TRELLIS [Xiang et al. 2025] achieves impressive results for a fully 3D mesh reconstruction method. However, the resulting mesh doesn't exactly reconstruct the input images. We note also issues as the input images are outside of distribution (e.g. the dog). The ModelSheetTo3D [2026] is well suited for characters with simple colors, but it struggles more with the other examples. Our method is able to represent both examples, while it can properly render lighting and line work. For each set, the first image corresponds to an input view while all the others are newly generated. The examples have respectively 3, 4 and 1 input image(s).

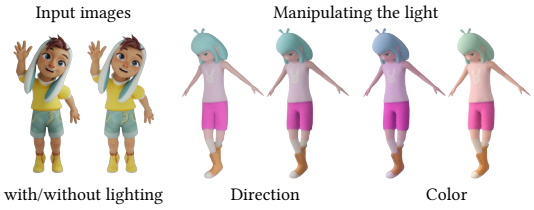


Fig. 8. Lighting. We use input images with and without lighting. The MLP is then used to shade the Gaussians. Both light direction and color can be adjusted.

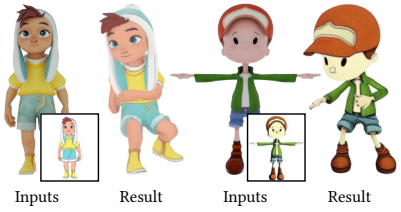


Fig. 9. Examples with character stylization. Given a single stylized reference, our model extends the style to all views and poses.

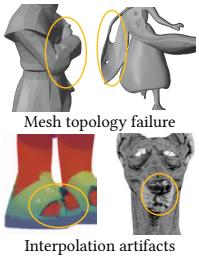


Fig. 10. Limitations of our approach.

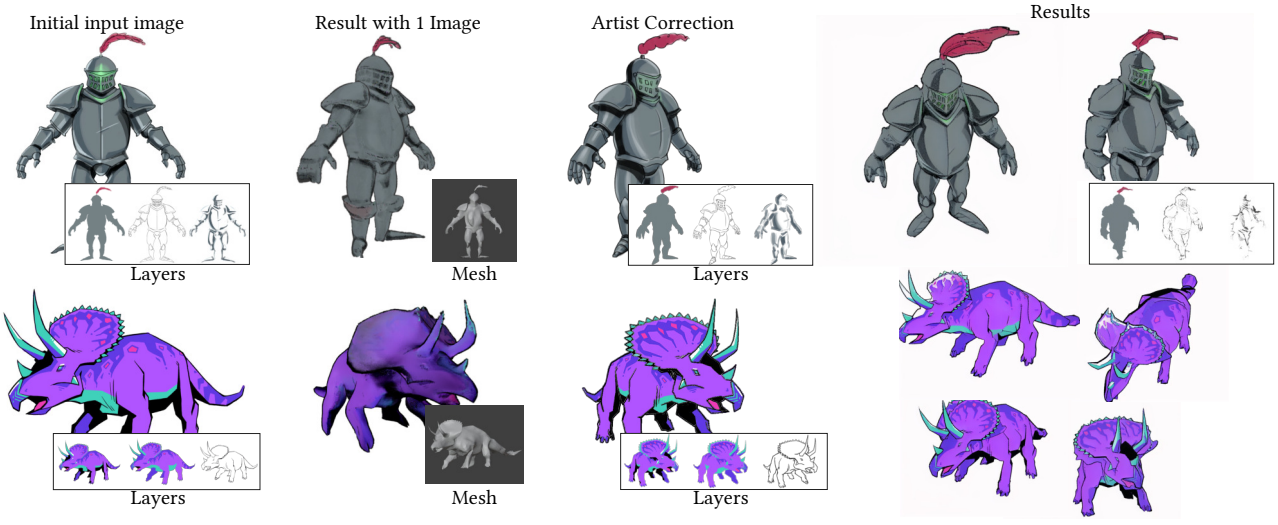


Fig. 11. Artist in the loop. Artists often paint stylization effects in separate layers to isolate and refine different aesthetic components while keeping the underlying design intact. Given a single layered stylized reference, our model creates a multi-layer animatable character. Artists can freely adjust different layers from different viewpoints and quickly see the updated results. This allows for a highly flexible and modular stylization workflow.