FLAT: Feedforward Latent Triangle Splatting for Geometrically Accurate Scene Generation

Orest Kupyn[1,2]* **Goutam Bhat**[1] **Philipp Henzler**[1]
Fabian Manhardt[1] **Christian Rupprecht**[1,2] **Federico Tombari**[1,3]

[1]Google Research
[2]University of Oxford, Visual Geometry Group
[3]TU Munich

Abstract

Generating explorable 3D scenes from a single image requires strong generative priors and accurate geometric representations suitable for downstream use. Current video diffusion models offer high-quality generation and implicitly encode multi-view geometric structure in latent space. However, existing feedforward latent scene decoders typically output volumetric 3D Gaussians that lack a well-defined surface, limiting their use in simulation or standard graphics pipelines. This motivates decoding *surface-aligned* primitives that are not only renderable but also closer to explicit geometric assets. We ask whether compressed video diffusion latents can be mapped directly to explicit surface primitives in a single pass. To this end, we introduce FLAT and, for the first time, show that triangle splats can be decoded directly from video diffusion latents. Compared with decoding 3D Gaussians, predicting flat primitives is notoriously more challenging due to high sensitivity to primitive orientations, oftentimes leading to poor gradient flow. FLAT solves with two key ingredients: a ray-centered rotation parameterization for triangle regression and a novel *product window function* that improves gradient flow during differentiable triangle rendering. On standard benchmarks, FLAT achieves significantly better geometric accuracy while maintaining competitive visual quality compared to state-of-the-art feedforward baselines. We further show that a lightweight test-time refinement step converts the predicted triangle soup into a fully opaque, game-engine-ready representation that supports real-time rendering. By evaluating 3DGS, 2DGS, and triangle splatting variants under an identical training setup, we provide the first systematic analysis of representation tradeoffs in feedforward scene generation. The project page is available at `https://flat-splat.github.io`.

1 Introduction

Creating explorable 3D environments is a challenging problem, with applications in mixed reality [68], robotics simulation [18, 34, 65], game asset creation [62], and autonomous driving [64, 43]. These applications require not only visually plausible content but also geometrically accurate, physically grounded scene representations that capture 3D layout, surface structure, and scale to support novel view synthesis, physical simulation, and interaction. The challenge is compounded when only a single image or text caption is available as input. The scene is under-determined: depth is ambiguous, and occluded surfaces are uncovered as the camera moves. Generating a complete explorable 3D scene from a single image, therefore, requires strong geometric and generative priors [38, 3].

*Work done during an internship at Google.

Figure 1: FLAT regress soft triangles directly from video diffusion latent, enabling geometrically accurate and high fidelity scene generation.

Recent advances in video diffusion models [55, 46, 59, 35, 47] offer a viable path towards this goal. These models learn rich priors and implicit 3D world understanding from internet-scale data. Nevertheless, video diffusion models alone cannot support interactive scene exploration due to high render time. Furthermore, they cannot ensure multi-view consistency. A number of approaches thus follow a generate-then-optimize paradigm [67, 73, 17] wherein a 3D Gaussian Splatting [33] or NeRF [45] representation is optimized to fit frames generated by the video model. This enables real-time rendering, but introduces substantial computational overhead due to the per-scene optimization.

Wonderland [38], Lyra [3], and Generative Gaussian Splatting [49] demonstrate that scene parameters can be regressed directly from video latents using lightweight decoders on top of frozen video diffusion models. By minimizing photometric error between original and rendered images, these approaches can generate high-quality, explorable 3D scenes. However, all these methods are restricted to generating only 3D Gaussians as output; these are volumetric, semi-transparent blobs that are well-suited to training scene decoders via differentiable rendering. Yet these same properties make them unsuitable for most graphics engines, which rely on opaque surface representations such as triangles or meshes [24]. While there are approaches to extract meshes from a Gaussian representation [60, 22], these often require complex post-processing and cannot produce satisfactory results.

Directly optimizing opaque triangle or mesh-based representations for the scene, however, is difficult due to the non-differentiability of the rendering process [42, 10]. Soft triangle representations [25, 24] can overcome this issue to enable per-scene optimization. Unfortunately, training feedforward triangle splatting decoder presents itself with further challenges. Exemplary, directly regressing vertices can easily result in degenerate solutions. Unlike volumetric Gaussians, incorrectly oriented flat triangles contribute negligibly to rendered images, yielding poor gradient supervision, especially early in training. Together, these issues make stable feedforward prediction of non-volumetric primitives an open problem, requiring careful choices in both parameterization and differentiable rendering.

We introduce FLAT, a feedforward model that directly predicts semi-opaque triangle-splatting primitives [25, 24] from the latent space of a frozen video diffusion model in a single forward pass. Given an input image, FLAT produces a geometrically accurate, physically grounded scene representation supervised by depth and normals. We enable efficient feedforward triangle decoding with two technical ingredients. First, we formulate a stable parameterization for flat primitives: each decoder token predicts a ray-centered triangle defined by a constrained Cholesky-style shape transform and residual rotations around a ray-aligned frame, avoiding degenerate triangles and unstable world-space orientations. Second, we introduce a modified window function that improves gradient flow across primitive boundaries during differentiable triangle rendering. Feedforward triangle model produces significantly more accurate geometry and on-par visual quality with volumetric variants Figure 1. For compatibility with standard graphics pipelines and game engines, we also provide an optional lightweight refinement step that converts the semi-transparent feedforward prediction into fully opaque triangles. We also train 3DGS [33] and 2DGS variants [28] under identical conditions, enabling direct comparison of the representations. Our contributions are as follows:

- We show for the first time that explicit, *non-volumetric* surface primitives can be decoded directly from compressed video diffusion latents in a single forward pass, and formulate the previously underexplored problem of how to parameterize and train feedforward flat-primitive decoding.
- We introduce the key ingredients that make this practical: a ray-centered local triangle parameterization with constrained Cholesky-style shape, residual orientation prediction around a ray-aligned frame, and a novel product window function that improves gradient flow and stabilizes training.
- We introduce FLAT, a feedforward pipeline from a single image to a game-engine-compatible format, and provide the first systematic comparison of 3DGS, 2DGS, and triangle splatting under identical latent decoding conditions, characterizing tradeoffs among rendering quality, geometric accuracy, and downstream mesh compatibility.

2 Related Work

Novel View Synthesis and Scene Generation 3D scene generation methods can be grouped into several categories. Early works train multi-view generation models [41, 51, 58, 17] to expand the viewset and then reconstruct an explicit 3D representation [45, 33]. Recently, ViewCrafter [67] has extended the generate-then-optimize paradigm to video diffusion, generating large, dense views from a point-cloud-conditioned video model. WorldStereo [73] adds explicit memory and 3D consistency optimization to video diffusion, enabling large-scale viewpoint generation. Yet all these methods require a complex two-stage pipeline with an expensive scene-optimization step, which limits scalability and computational efficiency. Recent feedforward novel view synthesis models [8, 11, 52, 66, 61, 39, 72, 70, 76] predict explicit 3D scene parameters, typically 3DGS, from RGB images. This streamlines the 3D scene generation pipeline by replacing complex scene optimization with a lightweight feedforward model. Yet such a pipeline discards all the intermediate features generated by a multi-billion-parameter video model only to re-estimate scene geometry from pixels.

Alternatively, geometry-free novel view synthesis methods completely omit the explicit prediction of 3D scene parameters. LVSM [31] directly maps input images to novel-view outputs, completely eliminating intermediate scene representations. LagerNVS incorporates features from geometry foundation models [57], showing the effectiveness of 3D-aware latent features for geometry-free generation. Recently, Genie [5] released a model that generates novel views in near-real time with high 3D consistency. Yet it requires the entire model to run for every new view rendered, demanding substantial computational resources. Thus, for many tasks, explicit 3D representations remain crucial.

Wonderland [38] extends the video diffusion model to 3D by training a 3DGS decoder directly from the latent space, enabling efficient single-stage 3D scene generation. Yet the task is highly complex: the decoder must infer scene geometry, appearance, and depth solely from rendering losses in a frozen, compressed latent space, while being guided by often imperfect cameras. Generative Gaussian Splatting [49] demonstrates that the highest quality is achieved with a second-stage scene optimization that starts from feedforward latent model predictions, which, yet again, increases the pipeline complexity. Lyra [3] improves the quality of the feedforward latent 3DGS model by incorporating multi-view video latents generated from a small set of preset trajectories. This allows bypassing supervision for noisy and complex trajectories but limits the scale and diversity of the generated scenes. All of the latent feedforward scene generation models are based on 3DGS. In contrast, FLAT explores a non-volumetric triangle representation for accurate scene geometry and direct compatibility with rendering engines. We also enable generation of diverse camera trajectories without post-scene optimization by improving the decoder architecture and camera pose guidance.

3D Scene Representations NeRF-style [45] volumetric representations established a powerful framework for view synthesis, but their rendering cost motivated a shift towards more efficient explicit scene parameterizations. 3D Gaussian Splatting [33] showed that collections of anisotropic 3D Gaussians enable high-quality real-time rendering. However, while Gaussian-based representations are flexible and efficient, they do not always provide the surface regularity, geometric precision, or structural control that some downstream tasks require. Thus, various modifications and alternatives of 3D Gaussians have been explored [23, 54, 30, 9], including 2D Gaussian splatting [28] for improved geometric accuracy. Recently, completely different representations, such as smooth 3D convexes [26] or radiance foams [19], have been proposed. Triangle Splatting [25] introduces differentiable rendering of soft triangles – the most classical primitive in computer graphics. MeshSplatting [24] further extends this line of work by enabling connectivity, allowing for differentiable mesh

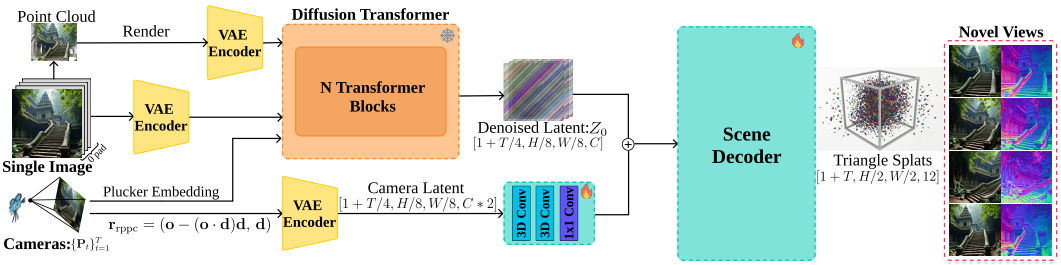


Figure 2: **Pipeline:** Starting from a single image, we construct a point-cloud-based control video by rendering along the target camera trajectory. The control video and camera embeddings condition a frozen video diffusion model [7]. The scene decoder then fuses denoised video latent with the camera latent and decodes triangle-splat scene representation for novel-view synthesis.

optimization. However, extending feedforward scene generation beyond Gaussians is challenging: non-volumetric representations have compact gradients and require precise orientation. FLAT addresses these issues, showing that triangles can be predicted directly from video latent, which in turn enables efficient and flexible scene generation with strong geometric accuracy and direct compatibility with modern rendering engines.

3 Method

Given a single RGB image $\mathbf{I}_0 \in \mathbb{R}^{H \times W \times 3}$ and a camera trajectory $\{\mathbf{P}_t\}_{t=1}^{T}$, where each $\mathbf{P}_t = (\mathbf{K}_t, \mathbf{R}_t, \mathbf{t}_t)$ denotes camera intrinsics and extrinsics, our goal is to produce an explicit 3D scene representation that can be rendered from arbitrary viewpoints in real time. The scene is represented as a set of surface primitives in world space, decoded in a single forward pass.

3.1 Pipeline Overview

Our method augments a frozen camera-conditioned latent video diffusion model [7] with a feedforward scene decoder, as illustrated in Figure 2. At test time, the pipeline takes as input a single RGB image $\mathbf{I}_0$ and a target camera trajectory $\{\mathbf{P}_t\}_{t=1}^{T}$. Conditioned on the input view and camera information, the video generator outputs a denoised latent $\mathbf{z} \in \mathbb{R}^{F' \times C' \times H' \times W'}$. We train a scene decoder that maps the latent directly to explicit scene parameters. The decoder predicts a set of surface primitives, which are then converted into world coordinates to form the final scene representation. In this way, FLAT reuses the strong generative prior of the video model, enabling plausible generation of scene content beyond the input view in a single forward pass without expensive per-scene optimization.

3.2 Feedforward Triangle Splatting

We represent the scene as a set of triangle splats, following differentiable triangle rendering [25]. Each triangle $\mathbf{T_m}$ is defined by three vertices $\mathbf{v}_i \in \mathbb{R}^3$, a color $\mathbf{c_m}$, a smoothness parameter σ_m and an opacity $o_m \in [0, 1]$. To render a triangle, we project each vertex to the image plane with a standard pinhole camera model. Given camera intrinsics $\mathbf{K}$ and pose $(\mathbf{R}, \mathbf{t})$, the projected vertices are

$$\mathbf{q}_{m,i} = \mathbf{K}(\mathbf{R_i}\mathbf{v}_{m,i} + \mathbf{t_i}), \tag{1}$$

where the three points $\mathbf{q}_{m,i} \in \mathbb{R}^2$ form the projected triangle T_m^{2D} in the image plane. To enable differentiable rasterization, we assign each pixel p a soft coverage value $I_m(p) \in [0, 1]$ via a window function described below. The rendered image is then obtained by accumulating the contributions of all overlapping triangles in front-to-back depth order, following the standard alpha-compositing equation used in differentiable splatting methods [28, 33, 26].

Decoding Triangles: A triangle splat can be parametrized by the model in several ways – directly predicting 3D vertices, edge vectors, or a canonical template with learned scale and rotation. Because triangles are flat primitives, orientation errors can yield negligible contributions to the rendered image, whereas degenerate solutions, such as three vertices forming a line, degrade training stability and

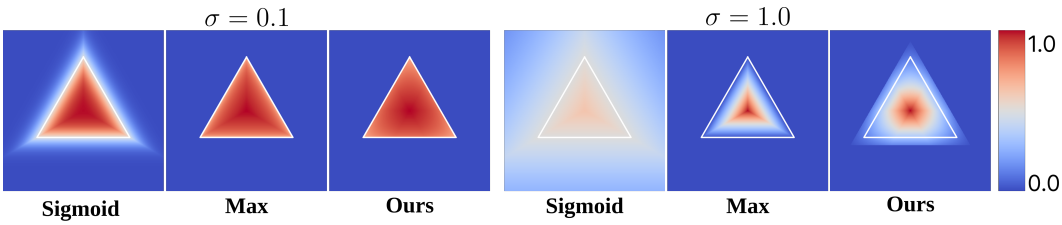


Figure 3: **Window Function:** Comparison of sigmoid-based window function [26, 14], max edge distance is used in [25] and ours. FLAT function extends the influence outside the triangle boundary and improves gradient flow by routing to all three vertices.

require additional constraints. Thus, feedforward training is particularly sensitive to the choice of parameterization. We predict each triangle relative to a ray-centered local frame and convert it to world space during post-processing. Concretely, each decoder output token predicts the parameters of a single triangle splat for a local 2×2 image region. For an anchor ray with origin $\mathbf{r}_o$ and direction $\mathbf{r}_d$, the network predicts a depth value D, three shape parameters, rotation parameters, color coefficients, opacity, and the sharpness parameter σ. The triangle center is placed at $\mathbf{r}_o + D \cdot \mathbf{r}_d$, while its in-plane geometry is first defined in a 2D coordinate system tangent to the ray.

Regressing three unconstrained vertices can result in degenerate triangles. Instead, we start from a canonical centered equilateral triangle in 2D and transform it with a lower-triangular matrix

$$\mathbf{L} = \begin{bmatrix} L_{00} & 0 \\ L_{10} & L_{11} \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{2}$$

whose coefficients are directly regressed by the model. The diagonal terms L_{00} and L_{11} are forced to be positive to maintain a valid triangle, while the off-diagonal term L_{10} controls shear. Applying $\mathbf{L}$ to the canonical triangle yields a flexible family of anisotropic 2D triangles while guaranteeing strictly positive area and avoiding degenerate configurations during training. We then translate the transformed vertices so that their centroid coincides with the anchor point along the ray.

Finally, the local 2D triangle is lifted to 3D using a ray-tangent frame. Orientation is parametrized by two residual tilt angles and an in-plane spin angle. We found this decomposition to be more numerically stable than predicting a full 3D rotation, such as a quaternion, for each triangle. Early in training, direct prediction of world-space rotation often leads to unstable orientation, vanishing render support, and model divergence. By predicting residual rotations around a ray-aligned frame, triangles inherit position from the predicted ray depth, shape from the Cholesky-style 2D transform, and orientation from a locally constrained rotation. In our experiments, the rotation parameterization is a key ingredient for stable feedforward latent decoding of non-volumetric primitives.

Window Function: A window function replaces the hard triangle with a smooth approximation, enabling effective gradient flow. Thus, a choice of approximation is crucial for a stable feedforward decoder. For a projected triangle T_m^{2D}, let $L_{m,i}(p) = \mathbf{n}_{m,i}^\top p + d_{m,i}$ denote the signed distance to the i-th supporting edge line, where the outward normals $\mathbf{n}_{m,i}$ are chosen such that $L_{m,i}(p) < 0$ inside the triangle. Let s_m be the triangle incenter and let $\rho_m = -\max_i L_{m,i}(s_m)$ denote its inradius in screen space. We define the normalized edge response as

$$u_{m,i}(p) = -\frac{L_{m,i}(p)}{\rho_m} \tag{3}$$

so that $u_{m,i}(p) > 0$ inside the triangle and $u_{m,i}(p) = 1$ at the incenter. We then apply a shifted clipping

$$r_{m,i}(p) = \text{clamp}\left(u_{m,i}(p) + \epsilon,\, 0,\, 1\right), \tag{4}$$

where $\epsilon > 0$ extends support beyond the exact boundary. The final window value is

$$I_m(p) = \left(\prod_{i=1}^{3} r_{m,i}(p)\right)^{\sigma_m}, \tag{5}$$

where σ_m controls the sharpness of the splat. Each pixel receives a signal from the full triangle rather than only the most active edge. The shift by ϵ also yields non-zero derivatives beyond the boundary, which improves stability early in training.

Compared with the original triangle-splatting formulation [25] our formulation avoids the max reduction inside the window, as shown in Figure 3. In practice, this improves gradient flow, which is particularly important in our feedforward latent model.

3.3 Feedforward Scene Decoder

In this section, we describe our feedforward scene decoder that regresses the 3D scene parameters.

Architecture: In contrast to other feedforward scene generation methods [3, 38] that train small transformer decoders [15] or mamba-based architectures [20] from scratch, we modify the decoder of a pretrained video VAE instead. Concretely, we reuse the RGB decoder backbone of Wan-2.1 [55]. We introduce camera conditioning via zero-convolutional blocks and attach lightweight output heads that map intermediate decoder features to triangle parameters rather than RGB pixels. In addition, we remove the last upsampling stage of the decoder to reduce the number of predicted primitives, predicting triangle parameters for a 2×2 pixel area. This transfer-learning setup simplifies optimization of the challenging problem: the pretrained decoder implicitly captures local appearance and spatial patterns, allowing the model to focus on high-quality rendering and accurate geometry.

Camera Conditioning: We encode camera information as dense per-pixel ray embeddings aligned with the video latent. Starting from the pixel-aligned Plücker ray embedding

$$\mathbf{r}_{\mathrm{pl}} = (\mathbf{o} \times \mathbf{d},\, \mathbf{d}), \tag{6}$$

where $\mathbf{o} \in \mathbb{R}^3$ and $\mathbf{d} \in S^2$ are the ray origin and direction, we follow DiffusionGS [6] and replace the moment vector with the point on the ray closest to the world origin:

$$\mathbf{r}_{\mathrm{rppc}} = (\mathbf{o} - (\mathbf{o} \cdot \mathbf{d})\mathbf{d},\, \mathbf{d}). \tag{7}$$

This RPPC parameterization better exposes the ray position and relative depth to the decoder.

Let $\mathbf{r}^{\mathrm{rppc}} \in \mathbb{R}^{T \times 6 \times H \times W}$ denote the dense RPPC maps for a video. Following [3], we split them into reference-point and direction components, $\mathbf{r}^{\mathrm{ref}}, \mathbf{r}^{\mathrm{dir}} \in \mathbb{R}^{T \times 3 \times H \times W}$, and encode them separately with the pretrained VAE encoder $\mathcal{E}$:

$$\mathbf{E}^{\mathrm{ref}} = \mathcal{E}(\mathbf{r}^{\mathrm{ref}}), \qquad \mathbf{E}^{\mathrm{dir}} = \mathcal{E}(\mathbf{r}^{\mathrm{dir}}), \tag{8}$$

where $\mathbf{E}^{\mathrm{ref}}, \mathbf{E}^{\mathrm{dir}} \in \mathbb{R}^{T' \times C \times H' \times W'}$ match the video autoencoder downsampling. We concatenate them along channels and project back to the decoder width:

$$\mathbf{E}^{\mathrm{cam}} = \phi\left(\left[\mathbf{E}^{\mathrm{ref}}; \mathbf{E}^{\mathrm{dir}}\right]\right), \qquad \mathbf{E}^{\mathrm{cam}} \in \mathbb{R}^{T' \times C \times H' \times W'}, \tag{9}$$

where ϕ is a lightweight learned fusion layer. We inject $\mathbf{E}^{\mathrm{cam}}$ through zero-initialized blocks, so that camera features remain aligned with visual latents as the model gradually learns to use them. Because the decoder is time-causal, we train on shorter sequences and still decode larger scenes during inference.

3.4 Model Training

We use a pre-trained video diffusion model and only train the scene decoder. Our training relies on a dataset of videos with known camera trajectories as well as depth and normal maps. For each video, we precompute its latents using the frozen VAE. The scene decoder is then trained to regress the 3D scene representation from the video latents and the camera trajectory.

Implementation Details. We use Uni3C [7] as the video model, which is built on top of Wan-2.1 [55] and generates 49 to 81 frames using a resolution of 432×768. The VAE encoder temporarily downsamples the video by a factor of $r_t = 4$ and spatially by $r_s = 8$. We train the scene decoder in four progressive stages from 320 to $768p$, due to the high computational cost. Depending on the stage, a total of $V = N$ supervision views are used, equally split between seen and unseen views. More details are presented in Section D. We use the AdamW optimizer with a learning rate of $1e-4$ and train our model on 8 H100 GPUs for 200 000 iterations.

Losses: FLAT is supervised with a combination of photometric and geometry losses, together with several regularization terms. In line with other feedforward 3D models [76, 3, 38], we use a pixel-wise L_2 loss, along with a perceptual LPIPS loss [71], between the rendered and target frames. We also supervise rendered depth with a scale-invariant disparity loss, as in MiDaS [48]. Finally, we directly supervise our rendered normals against the pseudo-ground-truth normals $\mathcal{L}_{\mathrm{N}} = \frac{\sum_i M_i(1-\hat{\mathbf{n}}_i \cdot \mathbf{N}_i)}{\sum_i M_i}$, where $M_i = 1$ if $\alpha_i > 0.5$, $\hat{\mathbf{n}}$ is rendered normal and N is a ground truth. Finally, during the high-resolution training stage, we apply an opacity regularization term, as commonly used in feedforward 3D Gaussian methods [76, 3], and remove triangles with opacity below 40%. The full objective is a weighted sum of these terms:

$$\mathcal{L} = \lambda_{\mathrm{rgb}}\mathcal{L}_2 + \lambda_{\mathrm{perc}}\mathcal{L}_{\mathrm{LPIPS}} + \lambda_{\mathrm{D}}\mathcal{L}_{\mathrm{D}} + \lambda_{\mathrm{N}}\mathcal{L}_{\mathrm{N}} + \lambda_{\mathrm{O}}\mathcal{L}_{\mathrm{O}}, \tag{10}$$

where $\lambda_{\mathrm{rgb}} = 1.0$, $\lambda_{\mathrm{perc}} = 0.5$, $\lambda_{\mathrm{D}} = 0.01$, $\lambda_{\mathrm{N}} = 0.01$ and $\lambda_{\mathrm{O}} = 0.001$.

3.5 Opaque Mesh Conversion:

For a game-engine-compatible format, we use the global triangle sharpness σ and the connected-triangles support, following [24]. We set the initial $\sigma = 0.5$ and convert the predicted semi-opaque and sharp triangles into a mesh using a fast post-processing optimization over generated video frames. This procedure converts the semi-transparent renderable soup into a more coherent, opaque triangle, allowing direct export to various rendering engines. Starting from the feedforward output, we first refine depth, geometry, color, and opacity under the same photometric rendering objective used during training, then perform 50 iterations of an aggressive opacity-selection stage that pushes per-triangle opacity toward binary values and removes triangles with low support. The surviving triangles are snapped to near-opaque opacity, locally densified near boundaries, and stitched by merging mutually nearest boundary vertices and pruning floaters. Last, we run a brief repair stage that adjusts vertex positions and colors to recover image fidelity after the topology change.

4 Evaluation

We evaluate FLAT on the task of feedforward 3D scene generation from a single input image. Since most methods we compare with are closed-source, we follow the evaluation protocol described in Lyra [3], Bolt3D [53], and Wonderland [38].

Dataset: We train on a mixture of real and synthetic videos. Real videos from RealEstate10K [75] and DL3DV [40] provide realistic scene statistics, appearance variation, and naturally occurring camera motion. However, the camera trajectories from SfM are often noisy and scale-ambiguous. For these datasets, we use the camera annotations from RealCam-Vid [74]. Synthetic data complements real videos in two ways. First, we sample $25{,}000$ images from the object-centric S3OD [37] dataset, synthesize videos with the Uni3C model [7] with basic camera motions such as pans and zooms, and store the corresponding trajectory metadata. This significantly expands the data distribution and covers a wider variety of scenes in contrast to a limited set of real videos. Second, we regenerate videos from the first frame and the target trajectory for RealEstate10K and DL3DV. These regenerated sequences ensure that the model learns from the actual distribution of the video diffusion model and adapts to its noise and biases, reducing the train–test gap when decoding from its latent outputs. In practice, we first pretrain on the larger synthetic data and then perform a final finetuning stage on the real videos. To map all scenes to the same scale, we predict metric camera poses and depths with MapAnything [32]. We observed that Uni3C does not perfectly match the input camera conditions for challenging trajectories, so we recomputed the camera trajectory using MapAnything predictions for the generated videos. For real videos, we keep RealCam-Vid poses rescaled to metric scale. Pseudo-ground-truth surface normals are predicted by NormalCrafter [4].

4.1 Scene Generation Evaluation

We evaluate our approach for the image-to-3D scene generation task on RealEstate10K and DL3DV datasets in Table 1. To fairly analyze the impact of representation and isolate it from pipeline differences, we train comparable 3D Gaussian Splatting (3DGS) and 2D Gaussian Splatting (2DGS) variants with the same training hyperparameters and evaluate them under the same protocol. We also include state-of-the-art methods based on 3DGS representation for comparison.

Figure 4: **Geometric Quality:** The latent triangle model generates finer, more accurate geometry compared to Gaussian representations that are optimized for visual quality, while still maintaining high rendering fidelity.

Table 1: **Novel View Synthesis and Geometry Quality.** Feedforward triangle splatting generates significantly more accurate geometry compared to other representations while maintaining high visual quality compared to state-of-the-art methods. Our 3DGS variant achieves the highest visual fidelity, confirming the effectiveness of the training pipeline and design choices. Geometric quality denotes accuracy of generated normals. We report mean geometric quality over both RealEstate10K and DL3DV.

Method	Representation	RealEstate10K			DL3DV			Geometry Quality	
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	L_1 ↓	Cosine ↑
ZeroNVS	3DGS	13.01	0.378	0.448	13.35	0.339	0.465	–	–
ViewCrafter	3DGS	16.84	0.514	0.341	15.53	0.525	0.352	–	–
Wonderland	3DGS	17.15	0.550	0.292	16.64	0.574	0.325	–	–
Bolt3D	3DGS	21.54	0.747	0.234	-	-	-	–	–
Lyra	3DGS	21.79	0.752	0.219	20.09	0.583	0.313	–	–
FLAT	3DGS	**22.39**	**0.762**	**0.203**	**20.71**	**0.663**	**0.275**	0.686	0.116
FLAT	2DGS	22.03	0.734	0.219	20.44	0.647	0.284	0.388	0.587
FLAT	Triangles	21.45	0.710	0.245	20.04	0.627	0.314	**0.211**	**0.853**

In addition to the standard image quality metrics, we also evaluate the geometric accuracy of the generated scene by directly comparing rendered normal maps with normals extracted from ground-truth frames. Since FLAT is directly supervised with NormalCrafter [4], we employ Metric3D-v2 [27] to lower the impact of model bias. For the 3DGS variant, we compute normal with finite differences from nearby depth points [28]. Since 3DGS are volumetric blobs, they do not define clear geometry, generating near-random normals. Importantly, 2DGS explicitly models surfaces, yet it cannot be effectively supervised with high-quality normals. In our experiments, the direct supervision leads to numerical instability and model divergence; thus, we train with the original objective of normal self-consistency [28]. This improves geometric quality over 3DGS, yet the predicted surfaces remain soft and less structured than those produced by triangle splatting. The triangle model achieves superior geometric quality Figure 4, with a cosine similarity of 0.853 to Metric3D labels, compared to 2DGS's 0.587 (averaged over both RealEstate10K and DL3DV). At the same time, visual metrics are comparable to those of other state-of-the-art methods. Table 1 also demonstrates the overall effectiveness of our training pipeline. All three representations generate high-quality visuals comparable to or superior to current state-of-the-art methods. 3DGS remains the strongest rendering-oriented baseline overall due to its volumetric nature, improving the quality of novel view synthesis over previous state-of-the-art methods, thus serving as an approximate upper bound for the triangle splats. Essentially, its blob-like parameterization is easier to predict, can naturally handle various 3D structures, such as semi-opaque fog and thin edges via "needle"-like Gaussians, and directly optimizes pixel-wise metrics such as PSNR. Triangles, instead, recover

sharper, more geometrically faithful surfaces, provide an explicit, non-volumetric representation, and are substantially better aligned with downstream mesh extraction and real-time graphics pipelines. Overall, these results support explicit triangle-based feedforward scene decoding as a valid alternative when geometric accuracy and downstream compatibility matter.

4.2 Mesh Conversion Evaluation

Our approach FLAT provides the key benefit that predicted triangles can be converted into an opaque mesh with a lightweight post-processing step. We compare the quality of this mesh with the meshes obtained from 2DGS and 3DGS representations in Table 2. Importantly, existing methods for surface extraction from 3DGS/2DGS rely on dense view coverage and are highly sensitive to hyperparameter choices. We observe that, given our sparse-view coverage and smaller scene scale, each scene requires careful hyperparameter tuning, and no single set of parameters works well across indoor and outdoor scenes. Thus, traditional marching cubes or TSDF surface-extraction methods simply fail in most scenes. In contrast, our predictions only require simple postprocessing, forcing opaque sharp triangles and connecting nearby edges, which significantly reduces the hyperparameter sensitivity. Consequently, the opaque meshes obtained via triangles achieve a PSNR improvement of over 7 dB compared to 3DGS meshes on RealEstate10K.

Table 2: **Opaque mesh conversion evaluation.** We compare opaque mesh extraction strategies across scene representations on RealEstate10K and DL3DV.

Representation	Conversion	Vertices	RealEstate10K			DL3DV		
			PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
2DGS	TSDF	5M	15.89	0.633	0.468	12.00	0.433	0.563
3DGS	GS2Mesh [60]	4M	14.18	0.619	0.452	12.31	0.465	0.541
Triangles	Ours	0.5M	**21.23**	**0.749**	**0.388**	**19.71**	**0.609**	**0.466**

4.3 Ablation Study

Table 3: **Ablation studies.** We analyze the effects of architecture, window function, and representation design on RealEstate10K and DL3DV.

Architecture	Window Function	Representation	Rotation	RealEstate10K			DL3DV		
				PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Ours	Ours	Ours	Global	< 10	< 0.4	> 0.4	< 10	< 0.4	> 0.4
Ours	Ours	3 Offsets	Residual	20.09	0.674	0.289	19.18	0.588	0.372
Ours	Triangle Splatting	Ours	Residual	20.65	0.693	0.282	19.75	0.610	0.341
LongLRM [76]	Ours	Ours	Residual	21.24	0.701	0.275	19.74	0.608	0.355
Ours	Ours	Ours	Residual	**21.45**	**0.710**	**0.245**	**20.04**	**0.627**	**0.314**

We ablate the main design choices of FLAT across parameterization, rendering, conditioning, and post-processing. In particular, we study the effect of the ray-centered triangle parameterization, the modified triangle window function, the rotation parametrization, and model architecture. These ablations show that stable feedforward decoding of triangle primitives depends on the combination of all components. Predicting rotation directly in world space leads to model divergence to complete noise or empty renders. Employing the LongLRM Mamba-based decoder used in Lyra also underperforms, suggesting that its limited capacity is insufficient for decoding complex non-volumetric primitives. Changing the predicted parameterization reduces training stability, and reverting to the original window formulation weakens gradient flow.

5 Conclusion

We presented FLAT, a feedforward approach for generating 3D scenes from a single input image. Our method combines the strong generative prior of a frozen camera-conditioned latent video model with a lightweight scene decoder that predicts triangle splats directly in a single forward pass. This

design avoids expensive per-scene optimization, while still enabling plausible generation beyond the input view and real-time rendering of the resulting scene representation. At the representation level, we showed that non-volumetric surface primitives can be decoded from video latents when the parameterization and rendering formulation are chosen carefully. In particular, our ray-centered triangle parameterization and modified differentiable triangle-splatting formulation improve optimization stability and enable practical feedforward prediction of explicit surface structure at high resolution. We further showed that a short post-processing stage can convert the predicted triangle soup into a substantially more coherent opaque mesh while preserving visual quality. Our results suggest that latent video generation models can serve not only as image synthesizers, but also as powerful priors for direct 3D scene prediction. We expect this to encourage further work on explicit geometrically accurate feedforward scene generation and tighter integration between controllable generative models and real-time 3D rendering.

Acknowledgments

C.R. is funded by the European Union (ERC, Volute, 101222037). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. We thank Oleg Gordiichuk for the help with the mobile rendering demo.

References

[1] Zhaochong An, Menglin Jia, Haonan Qiu, Zijian Zhou, Xiaoke Huang, Zhiheng Liu, Weiming Ren, Kumara Kahatapitiya, Ding Liu, Sen He, et al. Onestory: Coherent multi-shot video generation with adaptive memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16173–16184, 2026.

[2] Zhaochong An, Orest Kupyn, Théo Uscidda, Andrea Colaco, Karan Ahuja, Serge Belongie, Mar Gonzalez-Franco, and Marta Tintore Gazulla. Vggrpo: Towards world-consistent video generation with 4d latent reward. *arXiv preprint arXiv:2603.26599*, 2026.

[3] Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David B. Lindell, Zan Gojcic, Sanja Fidler, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra: Generative 3d scene reconstruction via video diffusion model self-distillation. In *International Conference on Learning Representations (ICLR)*, 2026.

[4] Yanrui Bin, Wenbo Hu, Haoyuan Wang, Xinya Chen, and Bing Wang. Normalcrafter: Learning temporally consistent normals from video diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8330–8339, 2025.

[5] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.

[6] Yuanhao Cai, He Zhang, Kai Zhang, Yixun Liang, Mengwei Ren, Fujun Luan, Qing Liu, Soo Ye Kim, Jianming Zhang, Zhifei Zhang, et al. Baking gaussian splatting into diffusion denoiser for fast and scalable single-stage image-to-3d generation and reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 25062–25072, 2025.

[7] Chenjie Cao, Jingkai Zhou, Shikai Li, Jingyun Liang, Chaohui Yu, Fan Wang, Xiangyang Xue, and Yanwei Fu. Uni3c: Unifying precisely 3d-enhanced camera and human motion controls for video generation. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

[8] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

[9] Haodong Chen, Runnan Chen, Qiang Qu, Zhaoqing Wang, Tongliang Liu, Xiaoming Chen, and Yuk Ying Chung. Beyond gaussians: Fast and high-fidelity 3d splatting with linear kernels. *arXiv preprint arXiv:2411.12440*, 2024.

[10] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in neural information processing systems*, 32, 2019.

[11] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pages 370–386. Springer, 2024.

[12] Ruihang Chu, Yefei He, Zhekai Chen, Shiwei Zhang, Xiaogang Xu, Dingdong WANG, Hongwei Yi, Xihui Liu, Hengshuang Zhao, Yu Liu, et al. Wan-move: Motion-controllable video generation via latent trajectory guidance. *Advances in Neural Information Processing Systems*, 38:404–432, 2026.

[13] Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation. *arXiv preprint arXiv:2510.02283*, 2025.

[14] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 31–44, 2020.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] Hongyang Du, Junjie Ye, Xiaoyan Cong, Runhao Li, Jingcheng Ni, Aman Agarwal, Zeqi Zhou, Zekun Li, Randall Balestriero, and Yue Wang. Videogpa: Distilling geometry priors for 3d-consistent video generation. *arXiv preprint arXiv:2601.23286*, 2026.

[17] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024.

[18] Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv preprint arXiv:2602.06949*, 2026.

[19] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4135–4145, 2025.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[21] Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation. *arXiv preprint arXiv:2605.13724*, 2026.

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

[23] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19812–19822, 2024.

[24] 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. *arXiv preprint arXiv:2512.06818*, 2025.

[25] Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Daniel Rebain, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, et al. Triangle splatting for real-time radiance field rendering. In *Thirteenth International Conference on 3D Vision*, 2025.

[26] Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21360–21369, 2025.

[27] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[28] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.

[29] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

[30] Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable radial kernel splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21513–21523, 2025.

[31] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

[32] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

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

[34] Seung Wook Kim, Bradley Brown, Kangxue Yin, Karsten Kreis, Katja Schwarz, Daiqing Li, Robin Rombach, Antonio Torralba, and Sanja Fidler. Neuralfield-ldm: Scene generation with hierarchical latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8496–8506, 2023.

[35] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[36] Orest Kupyn, Fabian Manhardt, Federico Tombari, and Christian Rupprecht. Epipolar geometry improves video generation models. *arXiv preprint arXiv:2510.21615*, 2025.

[37] Orest Kupyn, Hirokatsu Kataoka, and Christian Rupprecht. S3od: Towards generalizable salient object detection with synthetic data. In *International Conference on Learning Representations (ICLR)*, 2026.

[38] Hanwen Liang, Junli Cao, Vidit Goel, Guocheng Qian, Sergei Korolev, Demetri Terzopoulos, Konstantinos N Plataniotis, Sergey Tulyakov, and Jian Ren. Wonderland: Navigating 3d scenes from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 798–810, 2025.

[39] Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

[40] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024.

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

[42] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7708–7717, 2019.

[43] Jiageng Mao, Boyi Li, Boris Ivanovic, Yuxiao Chen, Yan Wang, Yurong You, Chaowei Xiao, Danfei Xu, Marco Pavone, and Yue Wang. Dreamdrive: Generative 4d scene modeling from street view images. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 367–374. IEEE, 2025.

[44] Xiaofeng Mao, Zhen Li, Chuanhao Li, Xiaojie Xu, Kaining Ying, and Kaipeng Zhang. Yume1. 5: A text-controlled interactive world generation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7752–7761, 2026.

[45] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[46] OpenAI. Video generation models as world simulators, 2024. URL `https://openai.com/index/video-generation-models-as-world-simulators/`. Accessed: 2024.

[47] Adam Polyak et al. Movie gen: A cast of media foundation models, 2025. URL `https://arxiv.org/abs/2410.13720`.

[48] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[49] Katja Schwarz, Norman Mueller, and Peter Kontschieder. Generative gaussian splatting: Generating 3d scenes with video diffusion priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27510–27520, 2025.

[50] Tianchang Shen, Sherwin Bahmani, Kai He, Sangeetha Grama Srinivasan, Tianshi Cao, Jiawei Ren, Ruilong Li, Zian Wang, Nicholas Sharp, Zan Gojcic, Sanja Fidler, Jiahui Huang, Huan Ling, Jun Gao, and Xuanchi Ren. Lyra 2.0: Explorable generative 3d worlds. *arXiv preprint arXiv:2604.13036*, 2026.

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

[52] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024.

[53] Stanislaw Szymanowicz, Jason Y Zhang, Pratul Srinivasan, Ruiqi Gao, Arthur Brussee, Aleksander Holynski, Ricardo Martin-Brualla, Jonathan T Barron, and Philipp Henzler. Bolt3d: Generating 3d scenes in seconds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 24846–24857, 2025.

[54] Maria Taktasheva, Lily Goli, Alessandro Fiorini, Zhen Li, Daniel Rebain, and Andrea Tagliasacchi. 3d gaussian flats: Hybrid 2d/3d photometric scene reconstruction. *arXiv preprint arXiv:2509.16423*, 2025.

[55] Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[56] Angtian Wang, Haibin Huang, Jacob Zhiyuan Fang, Yiding Yang, and Chongyang Ma. Ati: Any trajectory instruction for controllable video generation. *arXiv preprint arXiv:2505.22944*, 2025.

[57] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[58] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023.

[59] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners, 2025. URL `https://arxiv.org/abs/2509.20328`.

[60] Yaniv Wolf, Amit Bracha, and Ron Kimmel. Gs2mesh: Surface reconstruction from gaussian splatting via novel stereo views. In *European Conference on Computer Vision*, pages 207–224. Springer, 2024.

[61] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 16453–16463, 2025.

[62] Yongzhi Xu, Yonhon Ng, Yifu Wang, Inkyu Sa, Yunfei Duan, Zhenhong Sun, Yang Li, Pan Ji, and Hongdong Li. Sketch2scene: Automatic generation of interactive 3d game scenes from user's casual sketches. *arXiv preprint arXiv:2408.04567*, 2024.

[63] Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025.

[64] Yu Yang, Alan Liang, Jianbiao Mei, Yukai Ma, Yong Liu, and Gim Hee Lee. X-scene: Large-scale driving scene generation with high fidelity and flexible controllability. *arXiv preprint arXiv:2506.13558*, 2025.

[65] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16227–16237, 2024.

[66] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. Yonosplat: You only need one model for feedforward 3d gaussian splatting. *arXiv preprint arXiv:2511.07321*, 2025.

[67] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024.

[68] Jinyan Yuan, Bangbang Yang, Keke Wang, Panwang Pan, Lin Ma, Xuehai Zhang, Xiao Liu, Zhaopeng Cui, and Yuewen Ma. Immersegen: Agent-guided immersive world generation with alpha-textured proxies. *IEEE Transactions on Visualization and Computer Graphics*, 2026.

[69] Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. *arXiv preprint arXiv:2603.04379*, 2026.

[70] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024.

[71] 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 conference on computer vision and pattern recognition*, pages 586–595, 2018.

[72] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. Flare: Feed-forward geometry, appearance and camera estimation from uncalibrated sparse views. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21936–21947, 2025.

[73] Yisu Zhang, Chenjie Cao, Tengfei Wang, Xuhui Zuo, Junta Wu, Jianke Zhu, and Chunchao Guo. Worldstereo: Bridging camera-guided video generation and scene reconstruction via 3d geometric memories. *arXiv preprint arXiv:2603.02049*, 2026.

[74] Guangcong Zheng, Teng Li, Xianpan Zhou, and Xi Li. Realcam-vid: High-resolution video dataset with dynamic scenes and metric-scale camera movements. *arXiv preprint arXiv:2504.08212*, 2025.

[75] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[76] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4349–4359, 2025.

A Pipeline Flexibility

A useful property of FLAT is that it generates scene parameters from the denoised latent of base Wan-2.1 without modifying the latent space or finetuning the diffusion transformer. At inference time, one can simply add FLAT decoder or replace the standard VAE RGB decoder, while leaving the upstream video generator unchanged. As a result, any Wan-2.1 variant finetuned from the base model can produce explicit triangle-based scene geometry instead of RGB frames. This includes image-to-video, text-to-video, video-to-video, control or editing pipelines [12, 56], as well as more real-time [69, 21], interactive [63, 44], long / autoregressive [1, 29, 13] or world-consistent variants [16, 2, 36, 73].

This decoder-swap design makes FLAT practical for many applications. The method does not require a separate 3D decoder for each pipeline mode; instead, it reuses the shared latent representation. Consequently, improvements in the upstream generator, such as better motion quality, stronger conditioning, or new control interfaces, can be directly transferred to scene generation without retraining a separate scene decoder for each variant. In this sense, FLAT is best viewed as a geometry-aware explicit 3D generator for a broader family of video-generation pipelines. Figure 5 illustrates this flexibility. We additionally verify this for text-to-video setting. Figure 6 shows two scenes produced from text prompts alone. In both examples, FLAT decodes the final text-to-video latents into explicit triangle-based scenes whose rendered views remain consistent with the predicted normals. These results suggest that the scene decoder is not limited to image-conditioned generation.

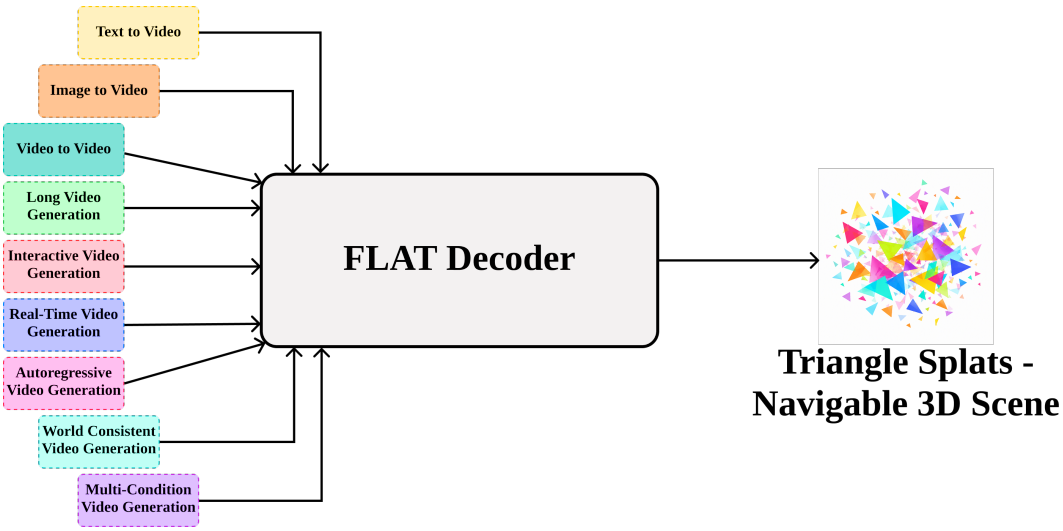


Figure 5: **Pipeline Flexibility:** FLAT replaces the standard RGB decoder with a latent scene decoder. Because multiple Wan variants share the same latent space, our scene decoder can be attached to any of these, including image-to-video, text-to-video, video-to-video, interactive, and world-consistent pipelines.

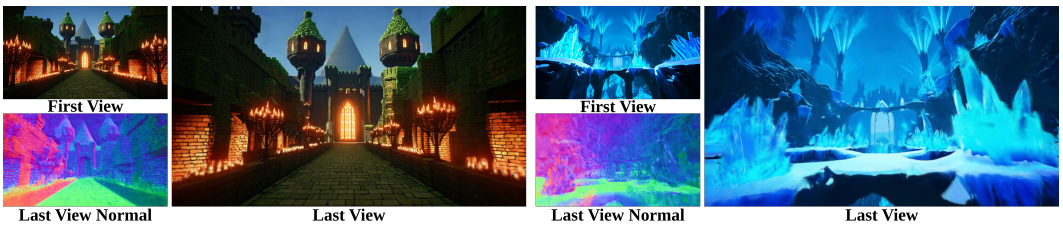


Figure 6: **Text-to-3D Scene:** Examples obtained by attaching FLAT to a Wan-2.1 text-to-video pipeline. For each scene, we show rendered views together with the corresponding predicted normal map. The examples demonstrate that the same latent scene decoder can convert text-to-video model latents into explicit geometry.

B Post Optimization

Though FLAT is fully feedforward, a short test-time optimization can further improve both visual and geometric quality. The feedforward prediction already provides a strong initialization, so optimization mainly corrects common failure cases of the latent feedforward model, including surface misalignment, semi-transparent structures, floating low-importance triangles, thin objects and overly diffuse normal predictions. In practice, we find that even a very short refinement of as few as 250 steps is often sufficient to improve both visual and geometric quality.

The optional post-optimization further aligns trinalge splat renders with RGB frames. We apply aggressive pruning to remove weak or unsupported triangles. This final cleanup is especially important for geometric quality: while diffuse low-opacity triangles can partially hide local prediction errors in RGB space, they tend to blur surface orientation and soften normal boundaries. Removing them produces sharper normal maps and cleaner local geometry. Qualitative examples are shown in Figure 7, where optimization improves both rendered appearance and predicted normals.

Figure 7: **Post Optimization:** Predictions and camera-space normals before and after optimization. A short refinement pass fixes common failures of the feedforward model and, together with aggressive pruning, produces cleaner geometry and sharper normals.

Table 4: **Effect of Post Optimization on RealEstate10K.** A simple optimization pass consistently improves the feedforward prediction.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
FLAT	21.45	0.710	0.245
FLAT + Optimization	**23.01**	**0.790**	**0.230**

C Limitations and Broader Impact

Despite strong geometric and visual quality FLAT still faces several limitations from the triangle representation and the feedforward generation. First, triangle splats are explicit, non-volumetric primitives, better aligned with surfaces but not optimized for standard novel-view synthesis performance. In particular, optimizing for pixel-level metrics such as PSNR remains more difficult than for 3DGS Figure 8. Also, content such as very thin, elongated structures, reflections, semi-transparent regions, and fine, high-frequency details is challenging to model with triangles and remains a primary source of failure Figure 10. Second, although our opaque-mesh conversion substantially improves usability, the resulting geometry is still sparser than a clean, watertight mesh: local connectivity can

be incomplete, surfaces can remain oversharpened or fragmented, and producing dense, fully coherent geometry still requires additional post-processing. This is a limitation of all scene mesh recovery methods, so a clean, densely connected, watertight mesh remains an open problem. The current model is also limited in scale. We train on a relatively small amount of data compared to modern video generation systems due to computational constraints and lack of high-quality ground truth data, and we expect both visual fidelity and geometric consistency to improve with dataset scaling. More fundamentally, FLAT predicts a scene from a single input image and one generated trajectory, so it must resolve severe 3D ambiguity from sparse view coverage. This can lead to incorrect geometry of occluded regions and failures on out-of-distribution scenes. In addition, our method currently targets a single generated scene or short camera path rather than a truly large explorable world [73, 50]. Extending it to persistent large-scale environments requires integration with long world-consistent video generation.

In terms of broader impact, our approach can make 3D scene generation more practical for applications such as simulation, robotics, gaming, and AR/VR, where geometric structure matters alongside image quality. At the same time, the same technology could be misused to create realistic synthetic environments or deceptive media. As with other generative models, improving realism lowers the cost of producing misleading content. Finally, training and deploying such systems require substantial computing resources, which carry environmental costs.

Figure 8: **Metric Limitations:** Gaussians are optimized for PSNR directly due to their inherent smoothness. The triangle model often generates sharper details while achieving lower PSNR.

Figure 9: **Qualitative Results:** More qualitative results covering indoor, outdoor, and object-centric scenes, focusing on surface and visual quality. Each sample consists of input image, novel view and novel view normal map.

Figure 10: **Failure Cases:** Thin, elongated surfaces, tiny details and reflections remain challenging to model with triangles.

Table 5: **Training Schedule:** The trainer progressively scales both number of views and resolution to reduce task complexity and improve computational efficiency

Stage	Iterations	Resolution	Input Views	Target Views
1	20,000	320p	17	17
2	40,000	320p	49	49
3	75,000	640p	49	49
4	75,000	768p	49	24

D Training Details

The multi-stage training pipeline follows a progressive resolution and view-scaling schedule across four stages Table 5. Stage 1 runs for $20,000$ iterations at 320p resolution, training on 17-view RealEstate10K sequences, using 17 input-conditioning views and 17 target views to quickly adapt the decoder to a new task. Stage 2 trains on 49-view trajectories from a mix of real and synthetic videos over $40,000$ iterations at 320p with 49 target views sampled for supervision. Stage 3 increases the image resolution to 640p for $75,000$ iterations. Finally, Stage 4 performs high-resolution fine-tuning at 768p for $75,000$ iterations using memory-efficient 8-bit Adam optimization, pruning, and gradient checkpointing to reduce GPU memory consumption.

Figure 11: **Converted Mesh:** Top Row: semi opaque triangles predicted by the model. Bottow Row: opaque game engine compatible mesh generated by lightweight conversion step. The scene render remains high as strong semi-opaque geometrically accurate initial predictions simplify conversion process.

E Scene Decoder Architecture

Scene decoder matches Wan-2.1 VAE architecture. It utilizes 3D causal convolution (CausalConv3D) that pads the temporal dimension exclusively on the past frames to maintain temporal causality. The input video latents $x_v \in \mathbb{R}^{B\times 16\times T'\times H'\times W'}$ and Plücker embeddings $x_p \in \mathbb{R}^{B\times 32\times T'\times H'\times W'}$ are fused via a lightweight adapter, where T', H', and W' denote the latent dimensions. To ensure the

Table 6: Computational Complexity: Scene decoding consumes only a marginal fraction of the total compute relative to the video generation.

Resolution	Views	Parameters (M)	Compute (FLOPs)	Throughput (FPS)
768p	49	73.34	9.8 TFLOPs	2.70

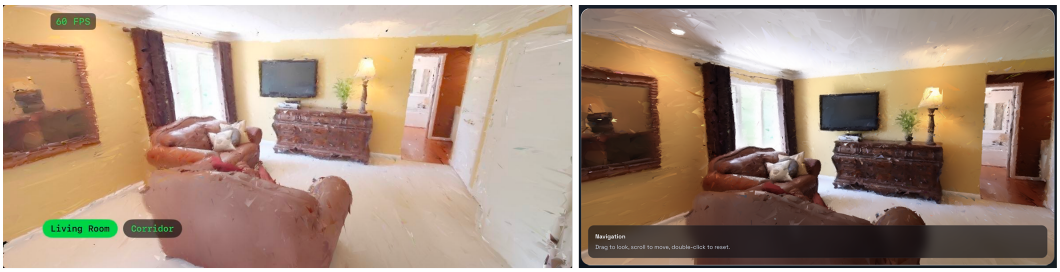
geometry conditioning does not destabilize the pre-trained latents at initialization, the final projection of the adapter is zero-initialized:

The fused latent x_{in} is then processed by a 3D UNet-style decoder, outlined in Table 7. Decoder employs RMSNorm, SiLU activations, and Scaled Dot-Product Attention (SDPA) throughout. Temporal upsampling occurs strictly in the first two upsampling stages, yielding a fixed $4\times$ temporal expansion. The final upsampling stage defaults to an identity pass, yielding an output tensor that is $2\times$ strided relative to the original image dimensions. Table 6 demonstrates detailed performance decoder metrics. We decode high resolution scene with 49 views in less than 300ms on H100 GPU.

Table 7: Detailed architecture of the WanSceneDecoder. Output shapes are denoted as (C, T', H', W') with the batch size B omitted for brevity. Residual blocks (ResBlock) consist of RMSNorm, SiLU, and CausalConv3D layers.

Stage	Layer / Operation	Output Shape	Details
Conditioning	Plücker Adapter	$(16, T', H', W')$	$3 \times 3 \times 3$ CausalConv, SiLU, ZeroConv
	Fusion (x_{in})	$(16, T', H', W')$	Addition with video latents
Input	Conv_{in}	$(384, T', H', W')$	$3 \times 3 \times 3$ CausalConv, padding=1
Mid Block	ResBlock_1	$(384, T', H', W')$	Dropout=0.0
	Attention	$(384, T', H', W')$	SDPA, 1×1 Conv Projections
	ResBlock_2	$(384, T', H', W')$	Dropout=0.0
Up Block 1	$3 \times$ ResBlock	$(384, T', H', W')$	
	Resample (3D)	$(384, 2T', 2H', 2W')$	Nearest-exact, Temporal + Spatial up
Up Block 2	$3 \times$ ResBlock	$(192, 2T', 2H', 2W')$	
	Resample (3D)	$(192, 4T', 4H', 4W')$	Nearest-exact, Temporal + Spatial up
Up Block 3	$3 \times$ ResBlock	$(96, 4T', 4H', 4W')$	
	Resample (Identity)	$(96, 4T', 4H', 4W')$	Identity spatial pass, no upsampling
Output Head	RMSNorm + SiLU	$(96, 4T', 4H', 4W')$	Independent or Monolithic head
	Conv_{out}	$(C_{out}, 4T', 4H', 4W')$	$3 \times 3 \times 3$ CausalConv
	Reshape	$(4T' \cdot 4H' \cdot 4W', C_{out})$	Flatten to Splat parameters

F Mesh Conversion Analysis



iPhone 15 **Web Browser**

Figure 12: **Cross-Platform Rendering:** Rendering raw output without any postprocessing or mesh cleanup. The converted solid triangles can be rasterized by any rendering engine across various platforms, supporting high-resolution and high-fps efficient rendering across devices.

To evaluate the quality of the conversion methods, we analyzed the geometry and topology of the output meshes. The direct conversion from soft triangles yields highly well-formed local geometry, with nearly zero degenerate faces (0.02%) and no fully isolated, disconnected triangles (0.00%). On average each triangle is connected to 3.1 other triangles. This aligns with the expectation of 3 for regular manifold surfaces, proving ability to extract compact global structure. To quantify the remaining topological complexity, we measure the rate of non-manifold edges (edges with more than two connected faces), which accounts for 10.60% of the mesh. Since the extracted mesh is not fully watertight, and rather represent a collection of locally connected surfaces, these non-manifold regions naturally emerge in locally dense zones where the network utilizes intersecting surface sheets to represent semi-transparent boundaries and fine details. The visual comparison is presented in Figure 11. The resulting mesh can be effectively rendered on any platform with high efficiency. We have verified the compatibility by rendering our results in browser, on iPhone 15 and Google Pixel devices without incorporating any custom rendering engines Figure 12.