LEARNING TEMPORALLY AWARE COMPACT GAUSSIANS FOR FEED-FORWARD 4D RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic scene reconstruction from monocular video remains a fundamental challenge in computer vision. Existing feed-forward methods predict per-pixel 3D Gaussians for each frame, leading to duplicated Gaussians and a view-dependent bias that hinder learning scene motion. We present Temporally Aware Compact 4D Gaussian Splatting (**T4GS**), a feed-forward 4D reconstruction framework built upon a compact set of timestamp-conditioned learnable query tokens. Each token aggregates features across the full temporal context and decodes a 3D Gaussian whose position is modulated by the target timestamp, enabling globally coherent motion modeling. The resulting temporally coherent Gaussian representation serves as a motion-aware anchor for camera-controllable video generation and supports 4D feature lifting for point tracking and dynamic scene understanding. Across multiple dynamic benchmarks, T4GS achieves strong novel-view synthesis without input camera poses while using significantly fewer Gaussians and remaining robust to large temporal gaps.

1 INTRODUCTION

Dynamic 4D reconstruction from monocular video is a fundamental problem in computer vision, with applications in AR/VR simulation (Pan et al., 2023), robotics (Nair et al., 2022), and content creation (Yang et al., 2026). Recent feed-forward methods tackle this problem by predicting per-pixel 3D Gaussians (Kerbl et al., 2023) from the input video, either producing a bullet-time reconstruction at a queried timestamp (Liang et al., 2024; Shen et al., 2025) or attaching attributes such as velocity or flow to represent the full 4D scene (Xu et al., 2025; Yang et al., 2026). Despite their effectiveness, we find that these methods exhibit two critical failure modes. First, under *large temporal motion*, they produce ghosting artifacts at interpolated timestamps (Fig. 1-(a)). Second, they show a strong *view-dependent bias*, relying predominantly on the temporally closest view and leaving regions visible only in temporally distant views poorly reconstructed (Fig. 1-(b)).

We argue that both issues stem from a design choice shared by all existing feed-forward 4D methods: *per-pixel Gaussian prediction*. In 4D scenes, since consecutive frames change only slightly, a per-pixel model can already achieve a low photometric loss by relying on the temporally closest views, without learning the underlying motion. This shortcut explains both failure modes. Ghosting arises because the Gaussians predicted from each neighboring view fail to model motion and remain near their positions at that view's timestamp, so under large motion, a moving object is duplicated at different locations (Fig. 1-(a)). The view-dependent bias arises because the model learns to rely on Gaussians only from temporally close frames, pushing opacity of the Gaussians from temporally distant views towards zero, resulting in failures of reconstructing such regions (Fig. 1-(b)).

To address these issues, we propose Temporally Aware Compact 4D Gaussian Splatting (T4GS), a *query-based, pose-free, feed-forward* framework for dynamic scene reconstruction. Specifically, we extract multi-frame features with a geometric model (Wang et al., 2025a). We then introduce a compact set of learnable queries that aggregate temporally and geometrically consistent information across the input frames via a transformer decoder, and decode them into 3D Gaussians that represent the scene at a certain target timestamp. By decoding the same learnable token across timestamps, we can directly obtain its Gaussian trajectory through time. Unlike bullet-time approaches that condition the full pipeline on the target time, we condition *only* the decoder, so encoder features are extracted only once and a new target time requires only a lightweight decoder pass.

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

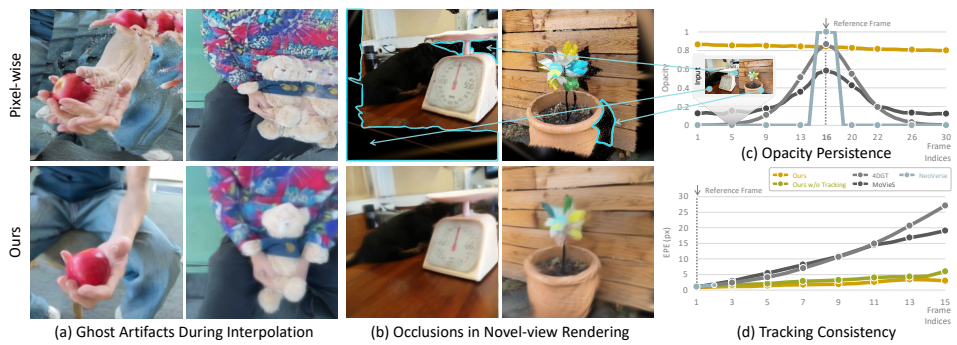


Figure 1: **Failures of pixel-wise feed-forward 4DGS.** (a) Duplicated Gaussians from nearby input views cause ghost artifacts. (b,c) Opacity falls off sharply away from the observed timestamp, so regions visible only in temporally distant views remain black in (b). Blue arrows link each such region to a context view in which it is visible (inset in (c)). (d) Drift grows with frame distance, confirming pixel-wise Gaussians cannot model global motion.

Unlike static scenes, we find that training a query-based Gaussian model on dynamic scenes is not trivial. As we decode the same set of query tokens across different timestamps, query tokens that model static regions should ideally be decoded into the same set of Gaussians, while for query tokens that represent dynamic regions, it should learn to model the dynamic motions. However, under the photometric loss alone on monocular videos, we find that the model fails to learn such behavior, showing jittering effects also for static regions, degrading the overall reconstruction quality.

To mitigate this, we introduce a curriculum training strategy and additional losses. To keep static regions fixed, we pre-train on static data before introducing dynamic scenes, and we then incorporate both monocular and multi-view video data. We observe that multi-view video helps the framework learn geometry, since observing one timestamp from several angles pins each Gaussian to the single location consistent with all views, which is a supervision signal that smoothly transitioning monocular video cannot provide. To use multi-view videos effectively, we propose a multi-view coverage sampling for context views to minimize occlusions in the target view. We further propose tracking and 3D flow losses that encourage each learnable token to decode the same Gaussian across time.

Once trained, T4GS predicts a Gaussian field with globally coherent 4D motion, enabling downstream applications beyond reconstruction. ***(i) Motion-aware anchor for video diffusion models (VDMs):*** Existing works condition VDMs on dense geometry for consistent generation, but this geometry often contains ghosting that leads to inconsistent motion. In line with findings that accurate conditions matter more than dense ones (Jin et al., 2026), our field serves as a coarse but temporally faithful anchor that preserves scene motion. ***(ii) Feature lifting:*** Lifting arbitrary image features to Gaussians has been a promising application of per-scene optimized 4DGS, but has remained difficult for feed-forward methods due to their limited ability to model global motion. T4GS enables efficient feature lifting by reusing its learned attention patterns, and the resulting 4D feature fields achieve more robust results in point tracking and dynamic scene understanding.

To validate our framework, we conduct extensive novel-view synthesis experiments on multiple dynamic benchmarks (Gao et al., 2022; Pan et al., 2023; Sturm et al., 2012; Yoon et al., 2020). Although our compact representation uses **0.008**$\times$ or fewer Gaussians than per-pixel methods, it achieves higher or competitive performance on all datasets by effectively modeling scene motion even in highly dynamic settings. We further show that our VDM generates high-fidelity video with more faithful motion, and we evaluate the 4D feature field on point tracking and dynamic scene understanding, confirming that the model produces meaningful 4D representations.

2 Related Works

Per-scene Optimization-based 4D Reconstruction. Given a monocular video, traditional methods reconstruct non-rigid contents through RGBD sensors (Bozic et al., 2020), hand-crafted priors (Kumar et al., 2017; Ranftl et al., 2016), or with low-rank assumptions (Bregler et al., 2000; Dai et al., 2014). Recent works extend powerful representations such as Neural Radiance Field (NeRF) (Mildenhall et al., 2021) and 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) to dynamic

scenarios. NeRF-based methods augment the network architecture with additional components such as flow fields (Du et al., 2021; Li et al., 2021; 2023), deformable fields (Park et al., 2021a;b), or voxel-based representations (Fridovich-Keil et al., 2023). 3DGS-based methods incorporate dynamic designs such as deformable fields (Luiten et al., 2024; Yang et al., 2024b), 4D Gaussian primitives (Wu et al., 2024), velocity (Wang et al., 2025e), or motion bases (Lei et al., 2025; Wang et al., 2025b). While these methods achieve compelling 4D reconstruction results, they require per-scene optimization, making it difficult to generalize across multiple scenes.

Feed-forward 4D Reconstruction. To address these limitations, recent approaches have explored generalizable feed-forward networks by learning priors from large-scale datasets or leveraging foundational priors (Leroy et al., 2024; Wang et al., 2025a; Han et al., 2025a). These methods either generate Gaussians at a specific timestamp (Shen et al., 2025; Liang et al., 2024) or additionally predict per-Gaussian velocities to capture motion (Xu et al., 2025; Ma et al., 2025; Lin et al., 2025; Hur et al., 2026). Despite their success, predicting per-pixel Gaussians yields millions of primitives, introducing redundancy, visual artifacts, and high computational cost. Moreover, per-pixel modeling also induces a view-dependent bias (An et al., 2026; Wang et al., 2025d). Rendering at a given timestamp draws almost entirely on Gaussians from the temporally nearest views, which suffices for smooth motion and leaves little incentive to reason about long-range motion. We instead reconstruct scenes through a compact set of learnable query tokens, an architectural bottleneck that compels the model to aggregate temporal information globally.

3 Methodology

We present T4GS, a query-based feed-forward framework that produces a compact 4D representation from a dynamic monocular video. We define the problem (§3.1), analyze the failure modes that motivate our design (§3.2), and describe the architecture (§3.3), training strategy (§3.4), analysis of the trained model (§3.5), and two applications (§3.6): a camera-controllable VDM that uses T4GS as a motion-aware anchor, and a feature-lifting decoder that reuses the learned attention patterns.

3.1 Problem Setup

Given a monocular video $\mathcal{V} = \{I_t \in \mathbb{R}^{H\times W\times 3}\}_{t=1}^{T}$ of T frames with width W and height H, and corresponding timestamps $\mathcal{T} = \{t \in \mathbb{R}\}_{t=1}^{T}$, our goal is a feed-forward model that renders high-quality novel views at an arbitrary timestamp $t_b \in [1, T]$ *without* calibrated camera poses at inference. Given context frames $\mathcal{V}_c \subseteq \mathcal{V}$ and their timestamps $\mathcal{T}_c \subseteq \mathcal{T}$, the model outputs a compact set of N Gaussians $\{\mathbf{G}_{t_b}^i\}_{i=1}^{N}$ representing the scene at a target timestamp t_b. Each Gaussian is defined as $\mathbf{G}^i = \{\mu^i, \sigma^i, \Sigma^i, c^i\}$, with mean $\mu^i \in \mathbb{R}^3$, opacity $\sigma^i \in [0, 1)$, covariance $\Sigma^i \in \mathbb{R}^{3\times 3}$, and view-dependent color $c^i \in \mathbb{R}^{3(l+1)^2}$ encoded with spherical harmonics (Fridovich-Keil et al., 2022) of degree l. Iterating over $t_b \in \mathcal{T}$ yields the full dynamic reconstruction and any intermediate t_b can be queried.

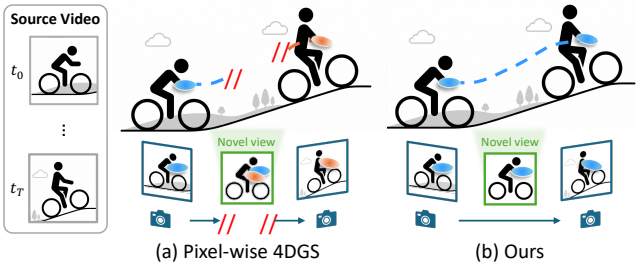


Figure 2: **Pixel-wise 4DGS vs. our approach.** **(a)** Pixel-wise methods produce duplicated, view-dependent Gaussians that cause ghosting artifacts. **(b)** Our approach aggregates global temporal context, yielding a compact, unified Gaussian set with temporally coherent motion.

3.2 Motivation

As illustrated in Fig. 2-(a), prior works (Yang et al., 2026; Xu et al., 2025; Lin et al., 2026) that predict per-pixel Gaussians produce redundant Gaussians tied to individual input views, each reliable only near the timestamp at which it was observed. This leads to the two failure modes discussed in §1. To better understand how each failure case emerges, we introduce two diagnostics, one for each failure mode: opacity persistence for the view-dependent bias and tracking consistency for ghosting.

Opacity Persistence. The view-dependent bias arises when each Gaussian models the scene only around the timestamp at which it was observed, rather than persisting across the sequence. Ideally, Gaussians representing a region visible in a reference frame should keep their opacity at other timestamps, even though such regions can be occluded. In Fig. 1-(c), we plot the opacity of the Gaussians predicted from a reference frame against the target timestamp. For per-pixel methods, this opacity decays as the target timestamp moves away from that of the reference frame, confirming that each Gaussian is bound to the view that observed it. This results in reconstruction failures. For example, although the region outlined in blue in Fig. 1-(b) is visible in a context view, per-pixel methods leave it black because the Gaussians from that view have opacity close to zero at the target timestamp.

Tracking Consistency. A Gaussian that models motion correctly should behave like a tracked point, following the same scene location from frame to frame. We therefore examine how faithfully each Gaussian follows the motion it should. In Fig. 1-(d), we take the Gaussians decoded from the first frame, trace their rendered 2D positions over time, and compare them against pseudo ground-truth tracks from CoWTracker (Lai et al., 2026). Per-pixel methods drift increasingly far from these tracks as time advances, indicating that their Gaussians do not accurately trace where the scene point actually is, even though NeoVerse and MoVieS are trained with motion supervision. Gaussians from per-pixel approaches model the scene adequately around their own timestamp but never acquire the global motion, consistent with the ghosting artifacts we observe. Further analysis is in § C.

3.3 Temporally aware Compact 4D Gaussian Splatting

To address the limitations of per-pixel prediction, we instead propose modeling 4D scenes with a compact set of Gaussians through a query-based transformer decoder, in which learnable queries aggregate information across frames via self-attention and are decoded into Gaussians (Fig. 2-(b)). The overall architecture is illustrated in Fig. 3.

Model Architecture. Given the context frames $\mathcal{V}_c$, we first extract a geometry feature map $\mathbf{F}_t \in \mathbb{R}^{h\times w\times d}$ for each frame I_t using an encoder $\mathcal{E}(\cdot)$, where h, w, and d denote the height, width, and channel dimension of the feature map, respectively. We initialize the encoder with VGGT (Wang et al., 2025a), which provides strong geometric priors learned from large-scale data.

On top of the extracted multi-frame features $\mathbf{F} = \{\mathbf{F}_t\}_{t\in\mathcal{T}_c}$, we adopt a query-based transformer decoder $\mathcal{D}_\mathcal{G}$, in which a small set of learnable queries gathers information from $\mathbf{F}$ and is subsequently decoded into Gaussians. Specifically, we introduce N learnable query tokens $\mathbf{Q} \in \mathbb{R}^{N\times d}$ and concatenate them with the flattened visual features as $[\mathbf{Q};\mathbf{F}]$. The concatenated tokens are passed through L transformer layers with full self-attention, gathering information from $\mathbf{F}$:

$$[\bar{\mathbf{Q}};\bar{\mathbf{F}}] = \mathcal{D}_\mathcal{G}([\mathbf{Q};\mathbf{F}]), \tag{1}$$

where $\bar{\mathbf{Q}}$ and $\bar{\mathbf{F}}$ denote the refined queries and features. Each refined query $\bar{\mathbf{Q}}^i \in \mathbb{R}^d$ is mapped to a 3D Gaussian $\mathbf{G}^i_{t_b}$ by a single-layer MLP. Since N is small, the model cannot assign a separate set of Gaussians to each input view as per-pixel methods do. Instead, the same tokens must represent both static and dynamic regions consistently across frames. This constraint discourages the shortcut of relying on the temporally closest view (§3.2) and encourages the model to learn the global motion.

Temporal Modelling. Dynamic reconstruction requires conditioning on both the capture time of each frame and the target time to decode. Rather than imposing motion priors such as per-Gaussian velocity (Xu et al., 2025), we directly decode the Gaussians at a queried target time t_b. Existing bullet-time methods (Liang et al., 2024) inject t_b at the input, which requires a full forward pass per timestamp and produces Gaussians with no correspondence across timestamps, causing flickering. We instead inject the time embedding only at the decoder, so the query tokens and encoder features are shared across timestamps and a new t_b requires only a lightweight decoder pass. Each token thus decodes into a corresponding Gaussian at every timestamp, yielding its 3D trajectory.

A straightforward design for temporal modeling is to embed each frame's absolute timestamp t into its features and the target timestamp t_b into the queries. Instead, we express all timestamps relative to the target, fixing the query timestamp to $\hat{t}_b = 0$ and shifting the frame timestamps by t_b:

$$\hat{\mathbf{F}}_{t|t_b} = \mathbf{F}_t + \mathcal{H}\big(\psi(t - t_b)\big), \quad \hat{\mathbf{Q}} = \mathbf{Q} + \mathcal{H}\big(\psi(0)\big), \tag{2}$$

where ψ denotes the sinusoidal positional embedding and $\mathcal{H}$ a two-layer MLP. Such design enables the queries to receive an identical embedding for every t_b, and t_b affects the decoder only with the offsets $t - t_b$ on the context features. Finally, $\{\hat{\mathbf{F}}_{t|t_b}\}_{t\in\mathcal{T}_c}$ and $\hat{\mathbf{Q}}$ are fed into Gaussian decoder $\mathcal{D}_\mathcal{G}$.

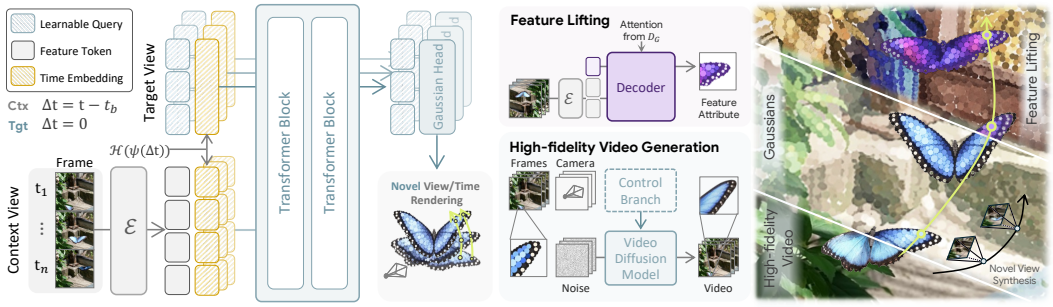


Figure 3: **Main architecture of T4GS.** (a) Learnable queries aggregate encoder features and are decoded into 3D Gaussians at the target time t_b. (b) The globally consistent motion of T4GS enables 4D feature lifting and camera-controllable video generation.

Although both designs carry the same temporal information, the relative formulation performs better in our ablation (Tab. 7). We conjecture this is because the input distribution of the queries remains fixed across target times, so each query consistently describes the same scene region and the token-to-Gaussian correspondence is preserved. In contrast, the image features already carry positional encodings, which likely make them more robust to a varying time embedding, so the temporal offset is better absorbed on the feature side.

3.4 Training

Unlike the static case, every Gaussian in the dynamic case is free to move as the target time varies. Ideally, only Gaussians on dynamic objects should move. However, we observe that photometric loss on monocular video alone rarely induces this and causes Gaussians to jitter in static regions. Moreover, the architectural bias in §3.3 encourages but does not guarantee that each token decodes a consistent Gaussian. We address both issues through a training curriculum and additional losses.

Dataset Curation. To prevent the decoded Gaussians from jittering in static regions, we first pre-train on static data before introducing dynamic data. Since everything is static, its rendering must be identical at any target timestamp, so randomly sampling the target timestamp penalizes drift induced by the temporal conditioning.

We then train on monocular and multi-view dynamic videos with static data. Multi-view data provides a constraint that monocular video cannot: supervising the same timestamp from several cameras places Gaussians at multi-view consistent positions. However, a context drawn from a single camera overlaps little with other target cameras, leaving holes in their renderings with little training signal. Rather than heuristically masking the loss over these regions, we propose a multi-view coverage sampling strategy, which augments the context $\mathcal{V}_c$ of video k' with one frame from each remaining video k at a random context timestamp. As consecutive frames largely share the same content, a single frame from each view covers most of its visible regions, making this strategy simple yet effective:

$$\mathcal{V}_c' = \mathcal{V}_c \cup \{I_{k,t^*}\}_{k \neq k'}, \quad t^* \sim \mathcal{U}(\mathcal{T}_c). \tag{3}$$

Loss Functions. Our primary objective is a photometric loss $\mathcal{L}_{\text{photo}}$. We additionally use depth and normal losses, $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{normal}}$, supervised by Wang et al. (2025c), to help the model better understand geometry in dynamic settings.

To further encourage each token to decode the same region across target times, we propose a tracking loss $\mathcal{L}_{\text{track}}$. If a token's Gaussian represents the same region over time, its projected trajectory should coincide with that region's point track. We thus project each Gaussian mean μ_t^i onto frame t, obtaining $\hat{\mathbf{p}}_t^i$. Using the projections $\hat{\mathbf{p}}_{t'}^i$ at a randomly selected frame $t' \in \mathcal{T}_c$ as query points, we obtain pseudo ground-truth tracks and binary visibilities $\{\mathbf{p}_t^i, m_t^i\}_{t=1}^T = \mathcal{P}(\mathcal{V}, \hat{\mathbf{p}}_{t'}^i)$ from the point tracker $\mathcal{P}$ (Lai et al., 2026). The loss penalizes deviations from these tracks:

$$\mathcal{L}_{\text{track}} = \sum_i \sum_t m_t^i \cdot \text{Huber}\big(\|\hat{\mathbf{p}}_t^i - \mathbf{p}_t^i\|\big). \tag{4}$$

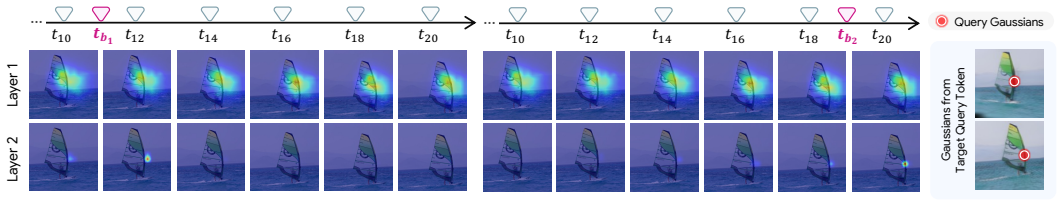


Figure 4: **Analysis of attention patterns.** Visualization of attention maps between the learnable query tokens and multi-frame features. For the query token decoding a specific Gaussian (red dot), the first layer attends to geometrically corresponding regions across all frames, while the second layer concentrates on frames temporally close to the target timestamp.

While the $\mathcal{L}_{\text{track}}$ constrains motion in 2D, we further propose a 3D flow loss $\mathcal{L}_{\text{flow}}^{\text{3D}}$ that keeps static regions fixed in 3D and lets only dynamic regions move. We first compute each Gaussian's displacement between consecutive timestamps in the camera coordinates of frame t, $\hat{\mathbf{f}}_t^i = \pi_t \mu_{t+1}^i - \pi_t \mu_t^i$, where π_t denotes the extrinsic of frame t. Using a dynamic mask M_{dyn} sampled at each Gaussian's projected location, we split the Gaussians into a dynamic set $\mathcal{S}_{\text{dyn}}$ and a static set. Dynamic displacements are supervised by the pseudo ground-truth 3D flow $\mathbf{f}_t^i$, bilinearly sampled from the flow map $\mathbf{f}_t^{\text{3D}}$ at the projection of μ_t^i, while static displacements are pushed to zero. Both terms are weighted by the visibility v_t^i of $\mathbf{G}_t^i$, so that only non-occluded Gaussians are supervised:

$$\mathcal{L}_{\text{flow}}^{\text{3D}} = \sum_t \Big(\sum_{i \in \mathcal{S}_{\text{dyn}}} v_t^i \, |\hat{\mathbf{f}}_t^i - \mathbf{f}_t^i| + \lambda_{\text{static}} \sum_{i \notin \mathcal{S}_{\text{dyn}}} v_t^i \, |\hat{\mathbf{f}}_t^i| \Big). \tag{5}$$

Our final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}} + \lambda_{\text{normal}} \mathcal{L}_{\text{normal}} + \lambda_{\text{track}} \mathcal{L}_{\text{track}} + \lambda_{\text{flow}}^{\text{3D}} \mathcal{L}_{\text{flow}}^{\text{3D}}, \tag{6}$$

where λ_{depth}, λ_{normal}, λ_{track}, λ_{static}, and $\lambda_{\text{flow}}^{\text{3D}}$ denote the weights of the corresponding losses. Detailed descriptions of each loss term are provided in § A.2.

3.5 Analysis

Analysis of Emergent Properties. After training, we observe that the compact set of query tokens enables our model to capture global motion faithfully. To understand how this capability emerges, we examine how the learnable tokens aggregate multi-frame features. Specifically, we visualize the self-attention maps of $\mathcal{D}_{\mathcal{G}}$, taking the query from a single time-conditioned token $\hat{\mathbf{Q}}^i$ and the keys from the visual features $\{\hat{\mathbf{F}}_{t|t_b}\}_{t \in \mathcal{T}_c}$. We then vary only t_b to examine whether the same token keeps attending to, and decoding, the same region as the target time changes.

As shown in Fig. 4, each token attends to consistent feature regions as t_b varies, with two decoder layers playing complementary roles. The first layer attends broadly to geometrically corresponding regions across all frames, keeping the token's attention stable over time without attention supervision. The second layer concentrates on frames temporally close to t_b, letting each token retrieve only the features needed to decode its Gaussian at that time. We attribute this to implicit pressure from the objective: reconstructing a dynamic scene at arbitrary timestamps with *compact* Gaussians requires placing them at geometrically coherent locations while reasoning about motion across frames.

Revisiting the Failure Modes. These properties explain how T4GS alleviates the failure modes of per-pixel methods identified in §3.2. With few Gaussians, the model must allocate them to meaningful regions, and since each token attends to the same region across time, its Gaussian remains consistent as the target time varies. **(i)** Aggregating all views into a few Gaussians places them only where the scene requires, avoiding redundant, overlapping Gaussians and thus ghosting (Fig. 1-(a)). **(ii)** A Gaussian retains high opacity even when temporarily occluded, since its token aggregates over all frames and keeps modeling its part of the scene regardless of the target time (Fig. 1-(b,c)). **(iii)** Since opacity stays high at every timestamp and each token decodes a corresponding Gaussian, that Gaussian must follow a globally consistent trajectory (Fig. 1-(d)). Notably, a variant trained without motion losses ($\mathcal{L}_{\text{track}}$ and $\mathcal{L}_{\text{flow}}^{\text{3D}}$) still drifts substantially less than per-pixel methods, indicating that this property stems from the query-based design rather than motion supervision.

3.6 APPLICATIONS

Camera-Controllable VDM. Recent works (Lee et al., 2025b; Seo et al., 2026; Yang et al., 2026) use VDMs (Wan et al., 2025) for scene-consistent novel view video generation, utilizing estimated point maps or 4D Gaussian fields as an anchor for consistent generation. However, the ghosting and view-dependent bias problems prevent this anchor from expressing the scene motion accurately, often resulting in the generated novel views with inconsistent motion from the original video.

In contrast, T4GS is capable of rendering novel views of both static and dynamic content with faithful motion. Although coarse, this anchor lets the VDM focus on high-fidelity synthesis rather than repairing the anchor. We build on Wan-VACE (Jiang et al., 2025), which conditions video generation on reference frames through a ControlNet (Zhang et al., 2023)-style branch. Given a context video $\mathcal{V}_c$ and videos $\tilde{\mathcal{V}}$ rendered by T4GS along a novel trajectory, we mark the context frames as inactive ($M_c = 0$) and the rendered frames as reactive ($M_t = 1$), and feed the concatenated latents and masks into the control blocks. Aligned with recent findings that accurate conditions matter more than dense ones (Jin et al., 2026), renderings from T4GS provide coarse but accurate scene motion information, enabling the model to follow arbitrary camera trajectories while keeping the dynamic content consistent. Further details are in §A.4.

Any-Feature 4D Lifting. 4D feature lifting underpins downstream tasks such as dynamic scene understanding and tracking. However, the view-dependent bias of feed-forward 4DGS methods prevents them from modeling global motion, so each Gaussian inherits features from its source view alone and rarely accumulates evidence from other frames, so feature lifting has remained limited to per-scene optimization (Zhou et al., 2025) at substantial cost. We instead leverage the emergent property of $\mathcal{D}_\mathcal{G}$, which consistently aggregates multi-frame features into Gaussians carrying global motion, and present the first feed-forward architecture that directly generates 4D feature fields.

Since $\mathcal{D}_\mathcal{G}$ aggregates image features across frames into each query, feeding it features from a vision foundation model (VFM), $\mathbf{F}'_t = \mathcal{E}'(I_t)$, would let each query hold VFM features aggregated across frames (An et al., 2026). However, $\mathcal{D}_\mathcal{G}$ cannot serve both purposes, as its input is committed to decoding Gaussians. We therefore introduce a feature decoder $\mathcal{D}_\mathcal{F}$ with its own queries $\hat{\mathbf{Q}}'$, which reuses the attention maps $\mathbf{A}_\mathcal{G}$ of $\mathcal{D}_\mathcal{G}$ and replaces only the values with time-embedded VFM features $\hat{\mathbf{F}}'$: $[\bar{\mathbf{Q}}'; \bar{\mathbf{F}}'] = \mathcal{D}_\mathcal{F}\big([\hat{\mathbf{Q}}'; \hat{\mathbf{F}}']; \mathbf{A}_\mathcal{G}\big)$, where a single MLP head maps each $\bar{\mathbf{Q}}'^i$ to the feature attribute of $\mathbf{G}^i$. Since the i-th feature query shares the attention map of the i-th Gaussian query, attaching these features to the Gaussians yields a 4D feature field. Further details are in §A.3.

4 EXPERIMENTS

4.1 EXPERIMENTAL DETAILS

We use $N = 2048$ learnable queries and $L = 2$ transformer layers, with Wang et al. (2025a) as the visual encoder. We adopt the standard 3DGS representation (Kerbl et al., 2023) but fix the spherical harmonic degree to 0, representing appearance as a single RGB color per Gaussian. We train at 224×224 resolution on Spring (Mehl et al., 2023), Kubric (Greff et al., 2022), EgoExo4D (Grauman et al., 2024), and RealEstate10K (Zhou et al., 2018), optimizing with AdamW (Loshchilov & Hutter, 2017) at learning rates of 10^{-5} for the decoder and 10^{-7} for the backbone, cosine-annealed to $0.1\times$ their initial values. We train for 30K steps with a per-GPU batch size of 1 on 4 NVIDIA H100 GPUs, and static pre-training follows C3G (An et al., 2026). We evaluate novel-view synthesis on four dynamic datasets. **DyCheck** (Gao et al., 2022) and **NVIDIA** (Yoon et al., 2020) evaluate rendering from *unseen viewpoints* at both context and intermediate timestamps, whereas **ADT** (Pan et al., 2023) and **TUM-Dynamics** (Sturm et al., 2012) evaluate rendering at *unobserved* timestamps, testing temporal interpolation and global motion understanding. Further details are in §A.5 and §B.

4.2 QUANTITATIVE RESULTS

Novel View Synthesis. We compare against feed-forward 4DGS networks, both pose-required (Xu et al., 2025; Lin et al., 2026) and pose-free (Yang et al., 2026). In Tab. 1 and Fig. 5, T4GS outperforms or is competitive with all baselines **without input camera poses**, using **0.008**$\times$ or fewer Gaussians. Notably, pose-required methods receive ground-truth poses providing cross-view cor-

Table 1: **Quantitative results of novel view synthesis on dynamic datasets.** We evaluate T4GS on multiple dynamic datasets against both pose-required and pose-free per-pixel feed-forward 4DGS methods. **Bold** denotes the best and underline denotes the second-best performance.

Methods	Pose	#G ↓	DyCheck			ADT			TUM-Dynamics			NVIDIA		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
4DGT (Xu et al., 2025)	✓	272K	12.19	0.221	0.485	21.26	0.800	**0.177**	17.27	0.579	0.329	16.31	0.334	0.565
MoVieS (Lin et al., 2026)	✓	802K	12.45	0.243	**0.413**	25.55	0.747	0.202	19.48	0.663	**0.209**	17.47	0.445	**0.376**
NeoVerse (Yang et al., 2026)	✗	802K	12.43	0.223	0.426	21.99	0.632	0.273	18.97	0.609	0.280	15.76	0.318	0.468
Ours	✗	**2K**	**15.64**	**0.397**	0.444	**27.70**	**0.816**	0.202	**20.84**	**0.668**	0.277	**20.62**	**0.497**	0.384

Table 2: **Quantitative results of novel view synthesis with various intervals Δt on TUM-Dynamics.** We evaluate T4GS against per-pixel feed-forward 4DGS methods across varying Δt of input frames. **Bold** denotes the best and underline denotes the second-best performance.

Methods	Pose	$\Delta t = 2$			$\Delta t = 4$			$\Delta t = 6$			$\Delta t = 8$		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
4DGT (Xu et al., 2025)	✓	19.26	0.674	0.250	18.08	0.623	0.297	17.27	0.579	0.329	16.27	0.547	0.351
MoVieS (Lin et al., 2026)	✓	20.74	**0.731**	**0.150**	19.57	0.669	**0.192**	19.48	0.663	**0.209**	18.59	0.622	**0.232**
NeoVerse (Yang et al., 2026)	✗	19.86	0.655	0.217	19.67	0.643	0.241	18.97	0.609	0.280	18.98	0.620	0.271
Ours	✗	**22.20**	0.728	0.231	**21.42**	**0.692**	0.258	**20.84**	**0.668**	0.277	**20.30**	**0.653**	0.289



Figure 5: **Qualitative results of novel view synthesis on dynamic datasets.**

respondence, whereas a pose-free model must also disentangle camera and object motion. Due to per-pixel prediction, pixel-wise 4DGS shows view-dependent bias, failing to capture global motion and producing ghosting, whereas our queries aggregate over the entire sequence and decode Gaussians following the global motion of their regions. The advantage is most pronounced on **DyCheck**, where context and target views are far apart and a Gaussian observed once must persist until rendering. Consistent with §3.2, per-pixel methods drive opacity toward zero away from the observing timestamp, leaving holes, whereas ours remain opaque. The gap also widens with larger temporal intervals: when varying the input stride $\Delta t \in \{2, 4, 6, 8\}$ on TUM-Dynamics (Tab. 2), T4GS degrades far less, since per-pixel Gaussians model motion only near their observed timestamps.

Gaussian Motion Tracking. To validate the motion modeling of T4GS over per-pixel methods, we evaluate point tracking against feed-forward 4DGS on ADT (Pan et al., 2023). Following Wang et al. (2025b), we convert the Gaussians into a tracking field by propagating their centers across timestamps from query points on the ADT split of TAPVid-3D (Koppula et al., 2024). We vary the input stride $\Delta t \in \{2, 4, 8\}$ and report $< \delta_{16}$, the fraction of predicted tracks within 16 pixels of the ground truth, computed only on moving points to avoid rewarding camera motion alone. In Tab. 3, T4GS tracks more accurately than the baselines even though its Gaussians are not pixel-aligned, indicating that it captures more geometrically and temporally coherent scene motion. This is consistent with the tracking consistency analysis in Fig. 1-(d), where the Gaussians of per-pixel methods drift from the ground-truth tracks.

Table 3: **Gaussian motion tracking.**

Method	ADT (PCK)		
	$\Delta t = 2$	$\Delta t = 4$	$\Delta t = 8$
4DGT Xu et al. (2025)	22.96	11.37	6.82
MoVieS Lin et al. (2026)	28.26	23.54	13.03
NeoVerse Yang et al. (2026)	10.43	7.13	5.86
Ours	**41.54**	**28.47**	**20.66**

4.3 Applications

Camera-Controllable VDM. Since T4GS renders novel views with globally consistent motion, it can anchor a camera-controlled VDM that remains consistent on both static and dynamic content. As shown in Tab. 4, our model achieves higher scores despite using a *smaller* model and far *less* training data (4.9K video clips *vs.* 1.37M for NeoVerse). Prior methods either disregard consistency on dynamic content or inherit ghosting and holes from their per-pixel anchors, and both shortcomings propagate into generation. We attribute our advantage at smaller scale to the anchor: since it already carries the global scene motion, the VDM need not infer motion and can focus on appearance.

Table 4: **VBench (Huang et al., 2024) results for novel view generation.** We evaluate 100 videos from DAVIS (Pont-Tuset et al., 2017), using a different target camera trajectory for each video.

Method	Backbone	Sub. Consist.	Back. Consist.	Temp. Flick.	Motion Smooth.	Aesth. Quality	Image Quality
TrajectoryCrafter (Yu et al., 2025)	CogVideoX-Fun-5B (Yang et al., 2024a)	0.8297	0.9000	0.9229	0.9574	0.4876	0.5713
NeoVerse (Yang et al., 2026)	Wan2.1-T2V-14B (Wan et al., 2025)	0.8795	0.9230	0.9506	0.9798	0.5166	0.6392
Ours	Wan2.1-VACE-1.3B (Wan et al., 2025)	**0.9317**	**0.9579**	**0.9687**	**0.9825**	**0.5381**	**0.6398**

Table 5: **Temporally invariant feature.**

Methods	RGB-stacking				DAVIS			
	$< \delta^0$	$< \delta^2$	$< \delta^4$	$< \delta^z_{\text{avg}}$	$< \delta^0$	$< \delta^2$	$< \delta^4$	$< \delta^z_{\text{avg}}$
DINOv3-L	1.3	11.3	65.6	23.2	0.6	6.2	40.7	13.7
DINOv3-L + Ours	**6.9**	**38.3**	**74.4**	**39.3**	**2.2**	**19.8**	**63.1**	**27.1**
VGGT-Tracking	1.1	15.9	63.6	24.8	2.7	17.9	54.0	23.1
VGGT-Tracking + Ours	**4.9**	**28.7**	**68.0**	**33.0**	**4.4**	**23.4**	**68.7**	**30.5**

Table 6: **Dynamic scene understanding.**

Methods	DAVIS				
	mIOU↑	Acc↑	PSNR↑	SSIM↑	LPIPS↓
LSeg (Li et al., 2022)	0.550	0.801	-	-	-
LSM (Fan et al., 2024)	0.615	0.844	14.320	0.310	0.517
C3G (An et al., 2026)	0.345	0.528	18.623	0.376	0.460
Ours	**0.634**	**0.854**	**20.181**	**0.426**	**0.359**

Table 7: **Ablation studies on time embedding.**

Time Embeddings	NVIDIA		TUM-Dynamics		ADT (PCK)	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	$\Delta t = 2$	$\Delta t = 4$
Absolute	20.43	0.400	19.03	0.323	24.9	15.8
Relative Scale	20.41	0.400	19.67	0.303	26.0	16.7
Relative Translate	**20.66**	**0.388**	**20.33**	**0.284**	**34.2**	**22.8**

Table 8: **Ablation studies on loss function.**

Methods	NVIDIA		TUM-Dynamics		ADT (PCK)	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	$\Delta t = 2$	$\Delta t = 4$
Ours	**20.66**	0.388	**20.33**	**0.284**	**34.2**	**22.8**
w/o $\mathcal{L}_{\text{depth}}$, $\mathcal{L}_{\text{normal}}$	20.58	0.389	20.15	0.288	27.3	18.7
w/o $\mathcal{L}_{\text{track}}$, $\mathcal{L}^{\text{3D}}_{\text{flow}}$	20.59	**0.386**	20.20	0.287	23.5	16.5

Point Tracking with 4D Feature Fields. We evaluate our 4D feature field on point tracking following the TAP-Vid protocol (Doersch et al., 2022). We render per-frame features from the field and match points by nearest neighbor in feature space. A well-formed 4D field should yield features that are more temporally and geometrically consistent than the 2D features it was lifted from. As shown in Tab. 5, features rendered from our field outperform the original features of DINOv3 (Siméoni et al., 2025) and VGGT (Wang et al., 2025a).

Dynamic Scene Understanding. Since our framework lifts arbitrary 2D features, it can also lift semantic features from CLIP (Radford et al., 2021) and LSeg (Li et al., 2022), giving each Gaussian a semantic embedding consistent across viewpoints and timestamps and enabling open-vocabulary reasoning (Cho et al., 2024) in 4D. We evaluate on DAVIS (Pont-Tuset et al., 2017) sequences where LSeg produces a non-trivial foreground, a setting that favors LSeg since the sequences are selected by its own predictions. Even so, as shown in Tab. 6, our model outperforms prior feed-forward methods, which are largely limited to static scenes, and even surpasses LSeg, suggesting that cross-frame aggregation refines the semantic priors rather than merely preserving them.

4.4 Ablation Studies

We ablate the time embedding and the loss terms, training every variant for 10K steps. In Tab. 7, absolute timestamps, measured from the first context view, cause the query embedding to vary with t_b, degrading rendering quality, tracking consistency, and temporal interpolation. Anchoring timestamps to the target keeps the query embedding fixed and performs best across all settings, consistent with §3.3. Normalizing timestamps by the interval between the first context view and the target also underperforms, likely because the same temporal offset can map to different embeddings across videos depending on the normalization interval. Tab. 8 ablates each loss term by removing it from the full objective in turn. Removing any of the pseudo-supervision terms from off-the-shelf experts, including geometry loss ($\mathcal{L}_{\text{depth}}$, $\mathcal{L}_{\text{normal}}$) and motion loss ($\mathcal{L}_{\text{track}}$, $\mathcal{L}^{\text{3D}}_{\text{flow}}$), degrades performance, indicating that each term contributes to the final result. We attribute this to the geometric ambiguity of monocular dynamic reconstruction, which photometric supervision alone cannot resolve. The additional ablation studies are discussed in §D.

5 Conclusion

We present T4GS, a feed-forward framework for dynamic scene reconstruction from monocular video that departs from per-pixel Gaussian prediction. A compact set of learnable query tokens aggregates multi-frame features across the full temporal context and decodes 3D Gaussians at any target timestamp, yielding a representation with globally consistent motion. This representation further supports downstream applications, including camera-controllable video generation and 4D feature lifting for point tracking and dynamic scene understanding. Across multiple benchmarks, T4GS matches or outperforms existing feed-forward dynamic 3DGS methods without input camera poses, while using far fewer Gaussians and remaining robust to large temporal gaps.

AI USE STATEMENT

In this work, we used generative AI tools (LLMs) to aid and polish the writing to improve readability and clarity. We have not used generative AI tools for generating synthetic datasets, developing theoretical models, formulating or proving mathematical claims, proposing hypotheses, designing experiments, or interpreting results. Additionally, we used generative AI tools to create or edit software code. We have reviewed all AI-assisted work. We treated all AI outputs only as references and did not adopt any of them as is. Based on these references, the authors selectively revised and incorporated only the necessary parts. Every AI-assisted decision was discussed and made by at least two authors. We take full responsibility for the final content of this work produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we describe the model architecture and loss functions in §3, with further details in §A. Hyperparameters and training settings are provided in the implementation details (§4.1 and §A), and the evaluation protocol is described in §B. The code will be made publicly available.

REFERENCES

Honggyu An, Jin Hyeon Kim, Seonghoon Park, Jaewoo Jung, Jisang Han, Sunghwan Hong, and Seungryong Kim. Cross-view completion models are zero-shot correspondence estimators. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1103–1115, 2025.

Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyunah Ko, Junsu Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Learning compact 3d representations from feed-forward novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 363–373, June 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Matteo Balice, Yanik Kunzi, Chenyangguang Zhang, Matteo Matteucci, Marc Pollefeys, and Sunghwan Hong. No pose, no problem in 4d: Feed-forward dynamic gaussians from unposed multi-view videos. *arXiv preprint arXiv:2605.22190*, 2026.

Aljaz Bozic, Michael Zollhofer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7002–7012, 2020.

Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pp. 690–696. IEEE, 2000.

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*, pp. 19457–19467, 2024.

Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. *Advances in Neural Information Processing Systems*, 34:9011–9023, 2021.

Seokju Cho, Sunghwan Hong, and Seungryong Kim. Cats++: Boosting cost aggregation with convolutions and transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (6):7174–7194, 2022.

Seokju Cho, Heeseong Shin, Sunghwan Hong, Anurag Arnab, Paul Hongsuck Seo, and Seungryong Kim. Cat-seg: Cost aggregation for open-vocabulary semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4113–4123, 2024.

Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14304–14314. IEEE Computer Society, 2021.

Zhiwen Fan, Jian Zhang, Wenyan Cong, Peihao Wang, Renjie Li, Kairun Wen, Shijie Zhou, Achuta Kadambi, Zhangyang Wang, Danfei Xu, et al. Large spatial model: End-to-end unposed images to semantic 3d. *Advances in neural information processing systems*, 37:40212–40229, 2024.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5501–5510, 2022.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12479–12488, 2023.

Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 35: 33768–33780, 2022.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19383–19400, 2024.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Jisang Han, Honggyu An, Jaewoo Jung, Takuya Narihira, Junyoung Seo, Kazumi Fukuda, Chaehyun Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Enhancing 3d reconstruction for dynamic scenes. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Jisang Han, Sunghwan Hong, Jaewoo Jung, Wooseok Jang, Honggyu An, Qianqian Wang, Seungryong Kim, and Chen Feng. Emergent outlier view rejection in visual geometry grounded transformers. *arXiv preprint arXiv:2512.04012*, 2025b.

Sunghwan Hong and Seungryong Kim. Deep matching prior: Test-time optimization for dense correspondence. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9907–9917, 2021.

Sunghwan Hong, Seokju Cho, Seungryong Kim, and Stephen Lin. Integrative feature and cost aggregation with transformers for dense correspondence. *arXiv preprint arXiv:2209.08742*, 2022a.

Sunghwan Hong, Jisu Nam, Seokju Cho, Susung Hong, Sangryul Jeon, Dongbo Min, and Seungryong Kim. Neural matching fields: Implicit representation of matching fields for visual correspondence. *Advances in Neural Information Processing Systems*, 35:13512–13526, 2022b.

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Ranran Huang and Krystian Mikolajczyk. No pose at all: Self-supervised pose-free 3d gaussian splatting from sparse views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 27947–27957, 2025.

Zhening Huang, Hyeonho Jeong, Xuelin Chen, Yulia Gryaditskaya, Tuanfeng Y. Wang, Joan Lasenby, and Chun-Hao Huang. Spacetimepilot: Generative rendering of dynamic scenes across space and time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024.

Junhwa Hur, Charles Herrmann, Songyou Peng, Philipp Henzler, Zeyu Ma, Todd Zickler, and Deqing Sun. UFO-4D: Unposed feedforward 4D reconstruction from two images. In *ICLR*, 2026.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17191–17202, 2025.

Yudong Jin, Tao Xie, Qihang Zhang, Zehong Shen, Zhen Xu, Yujun Shen, Hujun Bao, Xiaowei Zhou, and Yinghao Xu. 4danyone: Create anyone in 4d from a casual monocular video. *arXiv preprint arXiv:2608.20335*, 2026.

Jaewoo Jung, Jisang Han, Jiwon Kang, Seongchan Kim, Min-Seop Kwak, and Seungryong Kim. Self-evolving neural radiance fields. In *ICCV 2025 Workshop on Wild 3D: 3D Modeling, Reconstruction, and Generation in the Wild*, 2025.

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.

Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12912–12921, 2022.

Fanjie Kong, Chen Min, Yiming Nie, Weihuang Chen, Linyue Liu, Liming Chen, Yizhe Li, Yitong Li, Haonan Hou, and Hongbin Sun. FF4WM: Semantically-guided feedforward 4D gaussian world model for dynamic driving scene reconstruction, 2026. SSRN preprint.

Jahyeok Koo, Inès Hyeonsu Kim, Mungyeom Kim, Junghyun Park, Seohyun Park, Jaeyeong Kim, Jung Yi, Seokju Cho, and Seungryong Kim. Mv-tap: Tracking any point in multi-view videos. *arXiv preprint arXiv:2512.02006*, 2025.

Skanda Koppula, Ignacio Rocco, Yi Yang, Joe Heyward, João Carreira, Andrew Zisserman, Gabriel Brostow, and Carl Doersch. Tapvid-3d: A benchmark for tracking any point in 3d. In *NeurIPS*, 2024.

Suryansh Kumar, Yuchao Dai, and Hongdong Li. Monocular dense 3d reconstruction of a complex dynamic scene from two perspective frames. In *Proceedings of the IEEE international conference on computer vision*, pp. 4649–4657, 2017.

Min-Seop Kwak, Jiuhn Song, and Seungryong Kim. Geconerf: Few-shot neural radiance fields via geometric consistency. *arXiv preprint arXiv:2301.10941*, 2023.

Zihang Lai, Eldar Insafutdinov, Edgar Sucar, and Andrea Vedaldi. Cowtracker: Tracking by warping instead of correlation. *arXiv preprint arXiv:2602.04877*, 2026.

Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3d gaussian splatting. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=GMiC4ccyHn.

JoungBin Lee, Jaewoo Jung, Jisang Han, Takuya Narihira, Kazumi Fukuda, Junyoung Seo, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. 3d scene prompting for scene-consistent camera-controllable video generation. *arXiv preprint arXiv:2510.14945*, 2025b.

Jiahui Lei, Yijia Weng, Adam W Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 6165–6177, 2025.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022.

Deqi Li, Shi-Sheng Huang, Zhiyuan Lu, Xinran Duan, and Hua Huang. St-4dgs: Spatial-temporally consistent 4d gaussian splatting for efficient dynamic scene rendering. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024a.

Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20775–20785, 2024b.

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6498–6508, 2021.

Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4273–4284, 2023.

Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, et al. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *arXiv preprint arXiv:2412.03526*, 2024.

Chenguo Lin, Yuchen Lin, Panwang Pan, Yifan Yu, Tao Hu, Honglei Yan, Katerina Fragkiadaki, and Yadong Mu. Movies: Motion-aware 4d dynamic view synthesis in one second. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 2026.

Chieh Hubert Lin, Zhaoyang Lv, Songyin Wu, Zhen Xu, Thu Nguyen-Phuoc, Hung-Yu Tseng, Julian Straub, Numair Khan, Lei Xiao, Ming-Hsuan Yang, et al. Dgs-lrm: Real-time deformable 3d gaussian reconstruction from monocular videos. *arXiv preprint arXiv:2506.09997*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Zhening Liu, Yingdong Hu, Xinjie Zhang, Rui Song, Jiawei Shao, Zehong Lin, and Jun Zhang. Dynamics-aware gaussian splatting streaming towards fast on-the-fly 4d reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pp. 800–809. IEEE, 2024.

Ziqiao Ma, Xuweiyi Chen, Shoubin Yu, Sai Bi, Kai Zhang, Chen Ziwen, Sihan Xu, Jianing Yang, Zexiang Xu, Kalyan Sunkavalli, et al. 4d-lrm: Large space-time reconstruction model from and to any view at any time. *arXiv preprint arXiv:2506.18890*, 2025.

Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A high-resolution high-detail dataset and benchmark for scene flow, optical flow and stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4981–4991, 2023.

Sheng Miao, Jiaxin Huang, Dongfeng Bai, Xu Yan, Hongyu Zhou, Yue Wang, Bingbing Liu, Andreas Geiger, and Yiyi Liao. Evolsplat: Efficient volume-based gaussian splatting for urban view synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11286–11296, 2025.

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.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5480–5490, 2022.

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng Carl Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20133–20143, 2023.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5865–5874, 2021a.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021b.

Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun. Dense monocular depth estimation in complex dynamic scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4058–4066, 2016.

Junyoung Seo, Hyunwook Choi, Minkyung Kwon, Jinhyeok Choi, Siyoon Jin, Gayoung Lee, Junho Kim, JoungBin Lee, Geonmo Gu, Dongyoon Han, et al. Grounding world simulation models in a real-world metropolis. *arXiv preprint arXiv:2603.15583*, 2026.

Qiuhong Shen, Xuanyu Yi, Mingbao Lin, Hanwang Zhang, Shuicheng Yan, and Xinchao Wang. Seeing world dynamics in a nutshell. *arXiv preprint arXiv:2502.03465*, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Jiuhn Song, Seonghoon Park, Honggyu An, Seokju Cho, Min-Seop Kwak, Sungjin Cho, and Seungryong Kim. Därf: Boosting radiance fields from sparse input views with monocular depth adaptation. *Advances in Neural Information Processing Systems*, 36:68458–68470, 2023.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 573–580. IEEE, 2012.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Hanzhang Tu, Zhanfeng Liao, Wei Min, Jiajun Zhang, and Yebin Liu. Tele360: Real-time feed-forward human reconstruction from sparse unposed cameras. *arXiv preprint arXiv:2609.15032*, 2026.

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

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*, pp. 5294–5306, 2025a.

Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9660–9672, 2025b.

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025c.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. Volsplat: Rethinking feed-forward 3d gaussian splatting with voxel-aligned prediction. *arXiv preprint arXiv:2509.19297*, 2025d.

Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21750–21760, 2025e.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20310–20320, 2024.

Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4d gaussian transformer using real-world monocular videos. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8254–8263, 2023.

Yuxue Yang, Lue Fan, Ziqi Shi, Junran Peng, Feng Wang, and Zhaoxiang Zhang. Neoverse: Enhancing 4d world model with in-the-wild monocular videos. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024a.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20331–20341, 2024b.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.

Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5336–5345, 2020.

Haibao Yu, Kuntao Xiao, Jiahang Wang, Ruiyang Hao, Yuxin Huang, Guoran Hu, Haifang Qin, Bowen Jing, Yuntian Bo, and Ping Luo. Recondrive: Fast feed-forward 4d gaussian splatting for autonomous driving scene reconstruction. *arXiv preprint arXiv:2603.07552*, 2026.

Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. Trajectorycrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 100–111, 2025.

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*, pp. 1–19. Springer, 2024.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023.

Shijie Zhou, Hui Ren, Yijia Weng, Shuwang Zhang, Zhen Wang, Dejia Xu, Zhiwen Fan, Suya You, Zhangyang Wang, Leonidas Guibas, et al. Feature4x: Bridging any monocular video to 4d agentic ai with versatile gaussian feature fields. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 14179–14190, 2025.

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.

APPENDIX

This appendix provides additional implementation details and experimental results that could not be included in the main paper due to space constraints. The appendix is organized as follows.

§ A introduces additional implementation details covering architecture, loss functions, and dataset curation and sampling strategies. § B provides details on the evaluation datasets and the comparison protocol used for feed-forward 4D Gaussian methods. § C provides further details and analysis of the failure cases in per-pixel methods. § D reports additional experiments and ablation studies. § E provides further qualitative results. § F provides additional related works. Finally, § G discusses the limitations of our model.

We additionally provide **video results** on in-the-wild videos in the supplementary material, including 3D visualizations of feed-forward 4DGS methods and camera-controllable video generation results. T4GS reconstructs moving objects with globally consistent motion, without the ghosting and holes observed in per-pixel methods, and its renderings serve as motion-aware anchors that let the video diffusion model follow novel camera trajectories while keeping dynamic content consistent.

A ADDITIONAL IMPLEMENTATION DETAILS

A.1 ARCHITECTURE DETAILS

Visual Feature Extractor. Given a monocular video $\mathcal{V} = \{I_t \in \mathbb{R}^{H\times W\times 3}\}_{t=1}^{T}$ with timestamps $\mathcal{T} = \{t \in \mathbb{R}\}_{t=1}^{T}$, where H, W, and T denote the frame height, width, and number of frames, we sample a subset of context frames $\mathcal{V}_c \subseteq \mathcal{V}$ with the corresponding timestamps $\mathcal{T}_c \subseteq \mathcal{T}$. Following prior works (Charatan et al., 2024; An et al., 2026), we extract visual features $\{\mathbf{F}_t \in \mathbb{R}^{h\times w\times d}\}_{t\in\mathcal{T}_c}$, where h, w, and d denote the spatial height, width, and channel dimension, using a geometry-grounded visual encoder $\mathcal{E}(\cdot)$ initialized from VGGT (Wang et al., 2025a):

$$\{\mathbf{F}_t\}_{t\in\mathcal{T}_c} = \mathcal{E}(\{I_t\}_{t\in\mathcal{T}_c}). \tag{7}$$

Temporal Embedding. Given the extracted multi-frame features $\{\mathbf{F}_t\}_{t\in\mathcal{T}_c}$, we introduce N learnable Gaussian queries $\mathbf{Q} \in \mathbb{R}^{N\times d}$, jointly optimized with the decoder. Each query is intended to decode a single 3D Gaussian representing the scene at the target time t_b.

To condition the output on a specific target time, we inject temporal embeddings into both the visual features and the learnable queries. We express all timestamps relative to the target time t_b:

$$\hat{t} = t - t_b, \quad \hat{t}_b = 0, \tag{8}$$

so that the target timestamp maps to 0 and each context timestamp becomes a relative offset. The queries thus receive the same embedding for every target time and only the features shift, which we find better preserves the token-to-Gaussian correspondence (§3.3). This formulation also makes the model invariant to absolute time translation, encouraging generalization across videos with different temporal spans.

The relative timestamps are encoded via sinusoidal positional embeddings $\psi(\cdot)$, projected through a two-layer MLP $\mathcal{H}(\cdot)$, and added to their respective inputs:

$$\hat{\mathbf{F}}_{t|t_b} = \mathbf{F}_t + \mathcal{H}\big(\psi(\hat{t})\big), \quad \hat{\mathbf{Q}} = \mathbf{Q} + \mathcal{H}\big(\psi(\hat{t}_b)\big). \tag{9}$$

Transformer Decoding. The time-conditioned queries and visual features are concatenated along the token dimension and passed through an L-layer Transformer decoder $\mathcal{D}_\mathcal{G}$ with full self-attention:

$$[\bar{\mathbf{Q}}; \bar{\mathbf{F}}] = \mathcal{D}_\mathcal{G}\big([\hat{\mathbf{Q}}; \{\hat{\mathbf{F}}_{t|t_b}\}_{t\in\mathcal{T}_c}]\big), \tag{10}$$

where $\bar{\mathbf{Q}} \in \mathbb{R}^{N\times d}$ and $\bar{\mathbf{F}}$ denote the refined query and visual tokens, respectively. Since queries and visual tokens attend to each other, each query aggregates spatiotemporal information from all context frames while the visual tokens are refined in turn.

Gaussian Head. Each refined query token $\bar{\mathbf{Q}}^i$ is independently mapped to a 3D Gaussian by a single-layer MLP $\mathcal{M}_\mathcal{G}(\cdot)$:

$$\mathbf{G}^i_{t_b} = \mathcal{M}_\mathcal{G}(\bar{\mathbf{Q}}^i), \quad i = 1, \ldots, N. \tag{11}$$

The resulting N Gaussians collectively represent the dynamic scene at the target time t_b and are rendered via differentiable Gaussian splatting (Kerbl et al., 2023).

A.2 Loss Details

Photometric Loss. To propagate the learning signal, we render the predicted Gaussians onto target views $I_t \in \mathcal{V}$, which include context frames with known camera poses π_t. For each target timestamp t, we first estimate the Gaussians $\{\mathbf{G}_t^i\}_{i=1}^N$ using T4GS, then render each pixel p of the target view $\hat{I}_t$ through alpha blending of Gaussian color according to their depth order (Kerbl et al., 2023):

$$\hat{I}_t(p) = \sum_{i=1}^{N} c^i \sigma^i \mathbf{G}_t^{i,\text{2D}}(p) \prod_{j=1}^{i-1} \big(1 - \sigma^j \mathbf{G}_t^{j,\text{2D}}(p)\big), \tag{12}$$

where c^i is the view-dependent color of the i-th Gaussian obtained by decoding spherical harmonics coefficients at the viewing direction, and $\mathbf{G}_t^{i,\text{2D}}$ denotes the 2D screen-space projection of the 3D Gaussian under viewpoint π_t.

The photometric loss combines a pixel-wise mean squared error $\mathcal{L}_{\text{MSE}}$ with the LPIPS loss $\mathcal{L}_{\text{LPIPS}}$ between the rendered image $\hat{I}_t$ and the ground-truth image I_t:

$$\mathcal{L}_{\text{photo}}(t) = \mathcal{L}_{\text{MSE}}(\hat{I}_t, I_t) + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}}(\hat{I}_t, I_t), \tag{13}$$

where λ_{LPIPS} is the weighting coefficient for the LPIPS loss term and is set to 0.05 by default. For Spring and Kubric, where a ground-truth dynamic mask is available, we compute the squared error separately over static and dynamic pixels and weight the dynamic term by a factor of 3.

Depth Loss. To encourage geometrically accurate Gaussian placement, we supervise the rendered depth map against pseudo ground-truth depths. The depth map $\hat{D}_t$ is rendered analogously to the color image via alpha-compositing, replacing color attributes with the z-values of the Gaussian means:

$$\hat{D}_t(p) = \sum_{i=1}^{N} z^i \sigma^i \mathbf{G}_t^{i,\text{2D}} \prod_{j=1}^{i-1} (1 - \sigma^j \mathbf{G}_t^{j,\text{2D}}(p)), \tag{14}$$

where z^i is the z-value of the mean position of Gaussians $\mathbf{G}_t^i$ in the camera coordinate frame.

Since no ground-truth depth is available, we generate pseudo ground-truth depth maps D_t using MoGe-2 (Wang et al., 2025c). As monocular depth predictions suffer from inherent scale-ambiguity, we adopt a scale-shift invariant loss (Song et al., 2023; Li et al., 2024b) rather than l_1 loss:

$$\mathcal{L}_{\text{depth}}(t) = \|(\alpha \hat{D}_t + \beta) - D_t\|, \tag{15}$$

where α and β are scale and shift parameters estimated via least-squares alignment (Wang et al., 2025c) between the rendered and pseudo ground-truth depth as follows:

$$\alpha, \beta = \underset{\alpha,\beta}{\text{argmin}}\, \mathcal{L}_{\text{depth}}(t). \tag{16}$$

Since MoGe-2 provides valid depth for every pixel, the loss is computed over all pixels. The scale α and shift β are obtained in closed form by least squares and detached from the computational graph, so that gradients flow only through $\hat{D}_t$.

Normal Loss. Complementary to depth supervision, we additionally supervise surface normals to enforce local geometric consistency. The pseudo ground-truth normal map is generated from MoGe-2 (Wang et al., 2025c). Since the normal vectors are free from scale ambiguity, we directly apply an l_1 loss without affine alignment.

We derive the rendered normal map $\hat{N}_t$ from the rendered depth map $\hat{D}_t$ by first back-projecting pixels into 3D using camera intrinsics K:

$$\mathbf{P}(u, v) = \hat{D}(u, v) K^{-1} [u, v, 1]^\intercal, \tag{17}$$

where (u, v) denotes the pixel coordinates. Surface normals are then estimated via the cross-product of finite-difference gradients:

$$\hat{N}(u, v) = \text{Norm}\Big(\frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v}\Big). \tag{18}$$

The normal loss $\mathcal{L}_{\text{normal}}$ is computed as:

$$\mathcal{L}_{\text{normal}}(t) = \|\hat{N}_t - N_t\|. \tag{19}$$

Tracking Loss. To suppress temporal flickering and Gaussian drift, we anchor Gaussian trajectories to the predictions of a pretrained point tracker. For each timestamp $t \in \mathcal{T}$, we project the mean μ_t^i of each Gaussian $\mathbf{G}_t^i$ onto the image plane:

$$\hat{\mathbf{p}}_t^i = \Pi\big(K\pi_t\mu_t^i\big), \tag{20}$$

where $\Pi(\cdot)$ denotes perspective division. We then randomly select a reference timestamp $t' \in \mathcal{T}_c$ and feed the projected points $\hat{\mathbf{p}}_{t'}^i$ as queries to the point tracker $\mathcal{P}$ (Lai et al., 2026), which propagates them across the full video:

$$\{\mathbf{p}_t^i, m_t^i\}_{t=1}^T = \mathcal{P}(\mathcal{V}, \hat{\mathbf{p}}_{t'}^i), \tag{21}$$

where $m_t^i \in \{0,1\}$ is set to 1 only when the predicted visibility score times confidence exceeds 0.1, which also masks out points leaving the image. The query points $\hat{\mathbf{p}}_{t'}^i$ and the resulting tracks $\mathbf{p}_t^i$ are detached and serve as fixed targets, so gradients flow only through the projected trajectories $\hat{\mathbf{p}}_t^i$. We set the Huber threshold to $\delta = 2.24$ pixels,

$$\mathcal{L}_{\text{track}} = \sum_i \sum_t m_t^i \cdot \text{Huber}\big(\|\hat{\mathbf{p}}_t^i - \mathbf{p}_t^i\|_2\big). \tag{22}$$

Flow Loss. We further constrain Gaussian motion with 3D flow, so that static regions stay fixed as the target time varies while only dynamic regions move. For Spring and Kubric, we use the ground-truth 3D flow provided by the datasets. Since EgoExo4D provides no ground-truth 3D flow, we generate pseudo ground-truth 3D flow maps from the metric depth of MoGe-2 (Wang et al., 2025c) and the optical flow of RAFT (Teed & Deng, 2020), as described below. All depth and flow targets are aligned to the scale of the predicted Gaussians with a single per-batch scale s_{align}, the median ratio between the depths of the Gaussians and the target depths at their projections, *i.e.*, $\tilde{D}_t = s_{\text{align}} D_t$ and $\tilde{\mathbf{f}}_t = s_{\text{align}}\mathbf{f}_t$.

Given the per-frame depth $D_t \in \mathbb{R}^{H\times W}$ and the optical flow $O_{t\to t+1}$, we first unproject each pixel $\mathbf{x} = (u, v)$ of frame t into its camera coordinate system:

$$\mathbf{X}_t(\mathbf{x}) = D_t(\mathbf{x})\, K^{-1}\tilde{\mathbf{x}}, \qquad \tilde{\mathbf{x}} = [u, v, 1]^\top. \tag{23}$$

The corresponding location in frame $t+1$ is $\mathbf{x}' = \mathbf{x} + O_{t\to t+1}(\mathbf{x})$, at which D_{t+1} is bilinearly sampled to unproject the matched point $\mathbf{X}_{t+1}(\mathbf{x}')$. As the two points lie in different camera coordinate systems, we align them with the relative pose $T_{t+1\to t} = \pi_t\pi_{t+1}^{-1}$ and take their displacement as the 3D flow:

$$\mathbf{f}_t(\mathbf{x}) = T_{t+1\to t}\,\mathbf{X}_{t+1}(\mathbf{x}') - \mathbf{X}_t(\mathbf{x}) \in \mathbb{R}^3. \tag{24}$$

Since the camera motion is cancelled by $T_{t+1\to t}$, $\mathbf{f}_t$ vanishes in static regions and is non-zero only where objects move in the world, making it a useful signal for separating dynamic from static content. We discard unreliable correspondences with a forward–backward consistency mask:

$$V_t(\mathbf{x}) = \mathbb{1}[\|O_{t\to t+1}(\mathbf{x}) + O_{t+1\to t}(\mathbf{x}')\|_2 < \tau_{\text{fb}}], \tag{25}$$

where we set τ_{fb} to 1px.

On the model side, we transform the decoded Gaussian means of two consecutive frames into the camera coordinates of frame t and take their displacement as the predicted 3D flow, $\hat{\mathbf{f}}_t^i = \pi_t\,\mu_{t+1}^i - \pi_t\,\mu_t^i$, where $\pi_t \in SE(3)$ is the world-to-camera extrinsic applied to μ in homogeneous coordinates. Its target $\mathbf{f}_t^i$ is obtained by bilinearly sampling the aligned flow map $\tilde{\mathbf{f}}_t$ at the projected center $\hat{\mathbf{p}}_t^i$. We define a dynamic mask $M_{\text{dyn}}(\mathbf{x}) = \mathbb{1}\big[\|\tilde{\mathbf{f}}_t(\mathbf{x})\|_2 > \tau_{\text{dyn}}\big]$ with $\tau_{\text{dyn}} = 0.005\,s$, where s is the scene scale computed as the median depth of the visible Gaussians. Sampling this mask at $\hat{\mathbf{p}}_t^i$ splits the Gaussians into a dynamic set $\mathcal{S}_{\text{dyn}}$ and a static set.

However, not every Gaussian carries a reliable target. A Gaussian that projects outside the image or onto a pixel rejected by V_t has no valid correspondence, and a Gaussian lying behind the visible surface is occluded, so supervising it with the flow of the surface in front would propagate an erroneous correspondence. We therefore keep only Gaussians whose camera-space depth z_t^i lies within $\tau_{\text{out}} = 10\%$ of the aligned depth at their projection:

$$v_t^i = \mathbb{1}\big[\hat{\mathbf{p}}_t^i \in \Omega\big] \cdot V_t(\hat{\mathbf{p}}_t^i) \cdot \mathbb{1}\Big[\big|z_t^i - \tilde{D}_t(\hat{\mathbf{p}}_t^i)\big| < \tau_{\text{out}}\,\tilde{D}_t(\hat{\mathbf{p}}_t^i)\Big], \tag{26}$$

where Ω denotes the image domain. We then split the visible Gaussians into dynamic and static ones: a Gaussian belongs to $\mathcal{S}_{\text{dyn}}$ if the magnitude of its pseudo ground-truth 3D flow $\mathbf{f}_t^i$, normalized by the scene scale, exceeds a threshold. Dynamic Gaussians are supervised to follow their target flow, while static Gaussians are penalized for any motion, with a weight λ_{static} to prevent the abundant static Gaussians from dominating the loss:

$$\mathcal{L}_{\text{flow}}^{\text{3D}} = \sum_t \Big(\sum_{i \in \mathcal{S}_{\text{dyn}}} v_t^i \big\| \hat{\mathbf{f}}_t^i - \mathbf{f}_t^i \big\| + \lambda_{\text{static}} \sum_{i \notin \mathcal{S}_{\text{dyn}}} v_t^i \big\| \hat{\mathbf{f}}_t^i \big\| \Big). \tag{27}$$

Training Loss. To train our model, we formulate our training loss $\mathcal{L}$ as follows:

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \lambda_{\text{depth}} \mathcal{L}_{\text{depth}} + \lambda_{\text{normal}} \mathcal{L}_{\text{normal}} + \lambda_{\text{track}} \mathcal{L}_{\text{track}} + \lambda_{\text{flow}}^{\text{3D}} \mathcal{L}_{\text{flow}}^{\text{3D}} \tag{28}$$

where λ_{depth}, λ_{normal} and λ_{track} denote the weights of $\mathcal{L}_{\text{depth}}$, $\mathcal{L}_{\text{normal}}$ and $\mathcal{L}_{\text{track}}$, respectively. We set $\lambda_{\text{depth}} = 0.001$, $\lambda_{\text{normal}} = 0.001$, $\lambda_{\text{track}} = 0.1$, $\lambda_{\text{flow}}^{\text{3D}} = 2.0$, and $\lambda_{\text{static}} = 0.0025$.

A.3 Feature Lifting Details

Architecture Details. For feature lifting, we first extract VFM features $\{\mathbf{F}'_t\}_{t \in \mathcal{T}_c}$ from the context frames $\mathcal{V}_c = \{I_t\}_{t \in \mathcal{T}_c}$ using a VFM encoder $\mathcal{E}'(\cdot)$:

$$\{\mathbf{F}'_t\}_{t \in \mathcal{T}_c} = \mathcal{E}'(\{I_t\}_{t \in \mathcal{T}_c}). \tag{29}$$

Since $\mathcal{D}_{\mathcal{F}}$ reuses the attention maps of $\mathcal{D}_{\mathcal{G}}$, we resize each $\mathbf{F}'_t \in \mathbb{R}^{h' \times w' \times d'}$ to the spatial resolution $h \times w$ of the encoder features $\mathbf{F}_t$, so that the two decoders share the same token layout. $\mathcal{D}_{\mathcal{F}}$ has its own N learnable queries $\mathbf{Q}' \in \mathbb{R}^{N \times d'}$ whose dimension matches the VFM features, where the i-th query decodes the feature attribute of the i-th Gaussian at the target time t_b. The relative timestamps are embedded by the time embedding MLP $\mathcal{H}$ shared with $\mathcal{D}_{\mathcal{G}}$ and added to the VFM features and the queries:

$$\hat{\mathbf{F}}'_{t|t_b} = \mathbf{F}'_t + \mathcal{H}\big(\psi(t - t_b)\big), \quad \hat{\mathbf{Q}}' = \mathbf{Q}' + \mathcal{H}\big(\psi(0)\big). \tag{30}$$

We then concatenate them and feed them into the feature decoder $\mathcal{D}_{\mathcal{F}}$, which shares the same architecture as $\mathcal{D}_{\mathcal{G}}$:

$$[\bar{\mathbf{Q}}'; \bar{\mathbf{F}}'] = \mathcal{D}_{\mathcal{F}}\big([\hat{\mathbf{Q}}'; \hat{\mathbf{F}}']; \, \mathbf{A}_{\mathcal{G}}\big), \tag{31}$$

where $\bar{\mathbf{Q}}' \in \mathbb{R}^{N \times d'}$ and $\bar{\mathbf{F}}'$ denote the refined query and feature tokens, respectively. Crucially, $\mathcal{D}_{\mathcal{F}}$ does not compute its own attention; at every layer l, it directly uses the attention map $\mathbf{A}_{\mathcal{G}}^l$ computed by $\mathcal{D}_{\mathcal{G}}$ for the same inputs and target time:

$$\mathsf{Attn}^l = \mathbf{A}_{\mathcal{G}}^l \, V'^l, \quad \mathbf{A}_{\mathcal{G}}^l = \mathsf{Softmax}\left(\frac{Q_{\mathcal{G}}^l K_{\mathcal{G}}^{l\,\mathsf{T}}}{\sqrt{d_k}} \right), \tag{32}$$

where $Q_{\mathcal{G}}^l$ and $K_{\mathcal{G}}^l$ are the query and key projections of $\mathcal{D}_{\mathcal{G}}$, and V'^l is the value projection of $[\hat{\mathbf{Q}}'; \hat{\mathbf{F}}']$ in $\mathcal{D}_{\mathcal{F}}$. Since the i-th row of $\mathbf{A}_{\mathcal{G}}^l$ is the attention of the i-th Gaussian query, the i-th feature query aggregates VFM features from exactly the regions used to decode the i-th Gaussian. Each decoded query token $\bar{\mathbf{Q}}'^i$ is then mapped by a single-layer MLP $\mathcal{M}_{\mathcal{F}}(\cdot)$ to the feature attribute of its Gaussian:

$$\{\mathbf{s}_{t_b}^i\}_{i=1}^N = \mathcal{M}_{\mathcal{F}}(\bar{\mathbf{Q}}'). \tag{33}$$

Attached to the Gaussians $\{\mathbf{G}_{t_b}^i\}_{i=1}^N$, these features form a 4D feature field that is rendered via differentiable Gaussian splatting.

$\mathcal{D}_{\mathcal{G}}$, including its queries, attention projections, and the time embedding MLP $\mathcal{H}$, is kept frozen. All parameters of $\mathcal{D}_{\mathcal{F}}$, namely the queries $\mathbf{Q}'$, value projections, feed-forward layers, and the head $\mathcal{M}_{\mathcal{F}}$, are trained from scratch.

Loss Function. Analogous to the photometric loss, we render the feature attributes onto each target view $I_t \in \mathcal{V}$ by replacing the Gaussian colors with the features in alpha blending:

$$\hat{\mathbf{F}}'_t(p) = \sum_{i=1}^N \mathbf{s}_t^i \sigma^i \mathbf{G}_t^{i,\text{2D}}(p) \prod_{j=1}^{i-1} \big(1 - \sigma^j \mathbf{G}_t^{j,\text{2D}}(p)\big). \tag{34}$$

The feature rendering loss is an ℓ_1 loss between the rendered feature map $\hat{\mathbf{F}}'_t$ and the pseudo ground truth $\mathbf{F}'_t = \mathcal{E}'(I_t)$ extracted by the VFM encoder from the target view:

$$\mathcal{L}_{\text{feat}}(t) = \|\hat{\mathbf{F}}'_t - \mathbf{F}'_t\|_1. \tag{35}$$

Implementation Details. Building on the trained T4GS, we attach a feature decoder $\mathcal{D}_{\mathcal{F}}$ that shares the same architecture as $\mathcal{D}_{\mathcal{G}}$ and is trained from scratch while $\mathcal{D}_{\mathcal{G}}$ remains frozen. To demonstrate generalizability across VFMs, we experiment with DINOv3 (Siméoni et al., 2025), VGGT (Wang et al., 2025a), and LSeg (Li et al., 2022). We adopt the same hyperparameters as T4GS, except that we train $\mathcal{D}_{\mathcal{F}}$ for only 5K steps, since the reused attention patterns are already well suited for aggregating relevant 2D features.

A.4 Camera Controllable VDM

Architecture Details. The overall architecture is illustrated in Fig. 6. The context video $\mathcal{V}_c \in \mathbb{R}^{T_c \times H \times W \times 3}$ with T_c frames and the video $\tilde{\mathcal{V}} \in \mathbb{R}^{T_r \times H \times W \times 3}$ with T_r frames rendered by T4GS along a novel camera trajectory are given as input. Following VACE (Jiang et al., 2025), our VDM consists of the original video DiT (Wan et al., 2025) and a control module whose architecture follows ControlNet (Zhang et al., 2023). The original video DiT takes as input a noisy latent $\mathbf{z}_{t'}$ together with a caption $\mathcal{C}$ generated from the context video by Qwen3-VL-8B-Instruct (Bai et al., 2025).

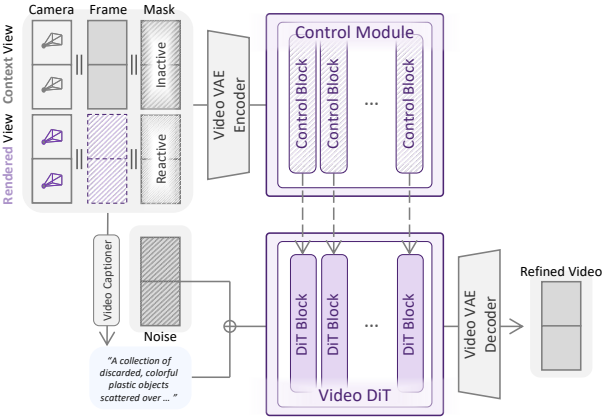


Figure 6: Camera-controllable VDM architecture.

For the control module, we adopt the VACE scheme, which designates the *inactive* component as the frames to be preserved and the *reactive* component as the frames to be regenerated. We treat the full context video as inactive and the rendered video as reactive, setting the masks to $M_c = 0$ for the context video and $M_t = 1$ for the rendered video.

For each of the context-view and target-view streams, we construct inactive and reactive inputs using the corresponding mask and encode them separately with the VAE encoder $\mathcal{E}_{\text{VAE}}(\cdot)$.

Let $\mathcal{V}_t = \tilde{\mathcal{V}}$. For each stream $s \in \{c, t\}$, we define

$$\mathbf{z}^s_{\text{inactive}} = \mathcal{E}_{\text{VAE}}\left(\mathcal{V}_s \odot (1 - M_s)\right), \tag{36}$$

$$\mathbf{z}^s_{\text{reactive}} = \mathcal{E}_{\text{VAE}}\left(\mathcal{V}_s \odot M_s\right). \tag{37}$$

To condition the control module on camera geometry, we represent each camera ray using its six-dimensional Plücker coordinates:

$$\mathbf{p}(\mathbf{o}, \mathbf{d}) = [\mathbf{o} \times \mathbf{d}; \mathbf{d}], \tag{38}$$

where $\mathbf{o}$ and $\mathbf{d}$ denote the ray origin and direction, respectively. Applying this representation to all rays yields the Plücker-ray feature tensors $\mathbf{P}_c$ and $\mathbf{P}_t$ for the context and target camera trajectories, respectively.

For each stream, the inactive and reactive latents are concatenated along the channel dimension with the corresponding mask latent and Plücker-ray features:

$$\mathbf{c}_c = \mathsf{Concat}_{\mathsf{ch}}\left[\mathbf{z}^c_{\text{inactive}}, \mathbf{z}^c_{\text{reactive}}, \mathbf{m}_c, \mathbf{P}_c\right], \tag{39}$$

$$\mathbf{c}_t = \mathsf{Concat}_{\mathsf{ch}}\left[\mathbf{z}^t_{\text{inactive}}, \mathbf{z}^t_{\text{reactive}}, \mathbf{m}_t, \mathbf{P}_t\right]. \tag{40}$$

Here, $\mathbf{m}_c$ and $\mathbf{m}_t$ denote the mask latents for the context-view and target-view streams, respectively.

The context-view and target-view streams are then concatenated along the temporal dimension:

$$\mathbf{c}_{\text{VACE}} = \mathsf{Concat}_{\mathsf{time}}\left[\mathbf{c}_c, \mathbf{c}_t\right]. \tag{41}$$

The resulting conditioning tensor is forwarded to the control module together with the noisy video latent, diffusion timestep, and caption condition:

$$\{\mathbf{f}_l\}_{l=1}^{L_{\mathcal{R}}} = \mathcal{R}_{\text{control}}\left(\mathbf{z}_{t'}, \mathbf{c}_{\text{VACE}}, t', \mathcal{C}\right), \tag{42}$$

where $L_{\mathcal{R}}$ denotes the number of layers in the control module and $\mathbf{f}_l$ denotes the hidden representation at layer l.

These control features are then added to the corresponding hidden states of the original video DiT $\mathcal{R}_{\text{VDM}}$:

$$\tilde{v}_{t'} = \mathcal{R}_{\text{VDM}}\big(\mathbf{z}_{t'}; t', \{\mathbf{f}_l\}_{l=1}^{L_{\mathcal{R}}}, \mathcal{C}\big), \tag{43}$$

where $\mathbf{z}_{t'}$ denotes the intermediate latent at diffusion timestep t', $\tilde{v}_{t'}$ the estimated velocity, and $\mathcal{C}$ the text caption condition.

Loss Function. Following Wan et al. (2025); Jiang et al. (2025), we train with the flow matching loss (Lipman et al., 2022). The intermediate latent at diffusion timestep $t' \in [0, 1]$ is constructed as

$$\mathbf{z}_{t'} = (1 - t')\,\mathbf{z}_0 + t'\,\mathbf{z}_1, \tag{44}$$

where $\mathbf{z}_0$ is the clean latent of the temporally concatenated ground-truth context-view and target-view videos, and $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the noise latent. The flow matching loss is

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\mathbf{z}_0, \mathbf{z}_1, t'}\left[w(t')\left\|\mathcal{R}_{\text{VDM}}\big(\mathbf{z}_{t'}; t', \{\mathbf{f}_l\}_{l=1}^{L_{\mathcal{R}}}, \mathcal{C}\big) - (\mathbf{z}_1 - \mathbf{z}_0)\right\|_{2,\text{tgt}}^2\right], \tag{45}$$

where $\|\cdot\|_{2,\text{tgt}}$ denotes the ℓ_2 norm evaluated over the target-view latent segment only, and $w(t')$ is the timestep-dependent training weight. During inference, the learned velocity field is integrated in the reverse direction to transport noise to data.

Implementation Details. We build on Wan2.1-VACE with 1.3B parameters. The model takes $T_c = 33$ context frames, consisting of 17 observed frames at even timestamps and 16 intermediate-time frames rendered by T4GS, together with $T_r = 33$ target-view conditioning frames rendered by T4GS along the target camera trajectory, and refines the latter into 33 output frames at a resolution of 480×480. We train on the Spring (Mehl et al., 2023) and CamxTime (Huang et al., 2026) datasets using the same sampling strategy as T4GS. We freeze the pretrained video DiT, VAE, text encoder, and the remaining base VACE parameters, and train rank-32 LoRA (Hu et al., 2022) modules in the attention and feed-forward layers of the VACE control module, together with its expanded patch embedding for the additional six-channel Plücker-ray condition. We use AdamW (Loshchilov & Hutter, 2017) with a learning rate of $1\text{e}{-4}$. Training is conducted on 4 NVIDIA H100 GPUs with a per-GPU batch size of 1 and 4 gradient accumulation steps, for a total of 47.6K steps.

A.5 Dataset Details

Monocular Video Dataset. We use Spring (Mehl et al., 2023) and EgoExo4D (Grauman et al., 2024) in the monocular setting. Spring is a high-quality synthetic video dataset with a large proportion of dynamic objects, of which we use the 37 scenes in the training split. From the 2K scenes of EgoExo4D, we use the four exocentric videos, each treated as an independent monocular video, and discard the egocentric view due to its fisheye distortion. We temporally subsample the EgoExo4D videos by a factor of 3 before training. For each sample, we randomly select a start index and take $T = 23$ consecutive frames. Every other frame starting from the first is used as context, yielding $|\mathcal{T}_c| = 12$ context frames with a temporal stride of 2, and the photometric and geometric losses are applied to all 23 frames. The non-context frames, which are never provided as input, fall temporally between context frames and thus require the model to predict the scene at unobserved timestamps. This design encourages temporal interpolation, synthesizing consistent Gaussians at arbitrary intermediate times rather than memorizing observed views.

Multi-view Video Dataset. For multi-view video data, we use Kubric (Greff et al., 2022). Following Koo et al. (2025), we generate 1,144 synthetic scenes with the Kubric rendering engine, each consisting of $M = 4$ videos $\{\mathcal{V}_k\}_{k=1}^{M}$, where $\mathcal{V}_k = \{I_{k,t}\}_{t=1}^{T}$ is captured along the k-th camera

trajectory. During training, a single video $\mathcal{V}_{k'}$ is chosen as the context $\mathcal{V}_c$, and frames from all M videos serve as novel-view rendering targets.

Since a context drawn from a single camera has little overlap with the other target cameras, we apply the multi-view coverage sampling of §3.4 (Eq. 3): among $|\mathcal{T}_c| = 6$ consecutive context timestamps, one is selected at random, and frames from all $M = 4$ cameras at that timestamp are included in the context, while only camera k' is used at the remaining timestamps.

Static Dataset. We use RealEstate10K (Zhou et al., 2018), a dataset of static-scene videos, to anchor static regions. We first pre-train on this dataset alone before introducing dynamic data, and continue to include it alongside the dynamic datasets thereafter. We sample 12 context frames spread over a longer temporal range of the video, with target frames in between, and assign each frame index as its timestamp. Since the scene is static, the rendering at a given target view should remain unchanged regardless of the assigned timestamp. We therefore randomly sample the target timestamp t_b, penalizing drift induced by the temporal conditioning.

Data Mixture. After pre-training on RealEstate10K alone, we train on all datasets jointly, sampling Kubric, EgoExo4D, RealEstate10K, and Spring with probabilities of 35%, 30%, 25%, and 10%, respectively.

B Evaluation Protocol

B.1 Evaluation Datasets

DyCheck. Following NeoVerse (Yang et al., 2026), we evaluate on the 5 scenes that provide multi-view videos with valid camera poses: *apple*, *block*, *paper-windmill*, *spin*, and *teddy*. As context, we use the casually captured handheld camera (Camera 0), sampling 32 frames with a temporal stride of 2 from a fixed 64-frame window in each scene. Novel views are rendered from a static camera (Camera 1) over all 64 frames of the window, spanning the temporal range of the context frames so that evaluation remains in the interpolation regime. Following Shape of Motion (Wang et al., 2025b), all metrics are computed only over co-visible regions, which we determine using LiDAR depth with respect to the 32 context frames, so that methods are not penalized for regions unobservable from the context.

Aria Digital Twin. Following 4DGT (Xu et al., 2025), we use 4 monocular videos from Aria Digital Twin (ADT) (Pan et al., 2023):

- *Apartment_release_multiuser_cook_seq141_M1292*
- *Apartment_release_multiskeleton_party_seq114_M1292*
- *Apartment_release_meal_skeleton_seq135_M1292*
- *Apartment_release_work_skeleton_seq137_M1292*

We sample 16 context frames with a temporal stride of 6 starting from the first frame, and evaluate on 15 target frames with the same stride starting from the fourth frame. All target timestamps are thus interleaved between context frames and lie strictly in the interpolation regime.

TUM-Dynamics. We use all 9 monocular videos in the training split of TUM-Dynamics (Sturm et al., 2012):

- *rgbd_dataset_freiburg2_desk_with_person*
- *rgbd_dataset_freiburg3_sitting_halfsphere*
- *rgbd_dataset_freiburg3_sitting_rpy*
- *rgbd_dataset_freiburg3_sitting_static*
- *rgbd_dataset_freiburg3_sitting_xyz*
- *rgbd_dataset_freiburg3_walking_halfsphere*
- *rgbd_dataset_freiburg3_walking_rpy*
- *rgbd_dataset_freiburg3_walking_static*

– *rgbd_dataset_freiburg3_walking_xyz*

We follow the same sampling as ADT. For the temporal-gap analysis (Tab. 2), we replace the stride of 6 with $\Delta t \in \{2, 4, 6, 8\}$ while keeping target frames interleaved between context frames.

NVIDIA. For the NVIDIA dataset (Yoon et al., 2020), we evaluate on 7 scenes: *Balloon1*, *Balloon2*, *Jumping*, *Playground*, *Skating*, *Truck*, and *Umbrella*. Each scene is captured by 12 synchronized cameras over 12 timestamps. We form a monocular context video by selecting camera k at timestamp k for every other timestamp, corresponding to a temporal stride of 2, and evaluate on all remaining camera–timestamp pairs.

B.2 Baselines

4DGT and MoVieS. 4DGT (Xu et al., 2025) and MoVieS (Lin et al., 2026) are feed-forward 4D Gaussian splatting networks built on the LRM architecture (Zhang et al., 2024), which take RGB frames together with their camera poses and timestamps as input. Since camera poses are given as input, these methods can more easily establish correspondences across frames and do not need to disentangle camera motion from dynamic scene motion, which gives them an inherent advantage over pose-free approaches. Since naively training a per-pixel Gaussian predictor on dynamic scenes struggles to model Gaussian motion and produces ghosting from duplicated Gaussians, both methods introduce dedicated techniques. 4DGT predicts time-varying Gaussian parameters such as velocity and regularizes the lifespan of each Gaussian, which, however, confines each Gaussian to a limited temporal span and hinders modeling global motion. MoVieS handles motion with a deformation field supervised by a flow loss. Both methods further adopt curriculum learning over datasets to stabilize training. For evaluation, we provide ground-truth camera poses, expressed in the coordinate frame of the first context camera, together with the timestamps. For 4DGT, which is trained with metric-scale poses, we use the ground-truth scale as is and convert frame indices to seconds. For MoVieS, whose predictions are in a normalized scale, we calibrate the camera-translation scale using only the input views. Specifically, we first run MoVieS on all context frames, select the translation scale that yields the most consistent predicted depths across consecutive context views, and then rerun MoVieS using the calibrated scale.

NeoVerse. NeoVerse (Yang et al., 2026) is a pose-free feed-forward 4D Gaussian splatting network built on the same VGGT (Wang et al., 2025a) backbone as ours and trained on 19 datasets. To cope with the difficulty of learning motion with per-pixel Gaussians, NeoVerse models each Gaussian's motion as a linear velocity valid only up to the adjacent frames, thereby constraining Gaussian movement to a short temporal range. Since NeoVerse combines a feed-forward 4D Gaussian model with a video diffusion model, we evaluate only its feed-forward 4D Gaussian model, which predicts Gaussians from the input views without any generative prior.

Ours. Unlike pose-required methods, T4GS directly generates a Gaussian field from an input video without camera poses, taking only the context frames and their timestamps as input. It predicts Gaussians in the coordinate frame of the first context camera, while ground-truth poses are used only during training to render the supervised target views relative to that frame. This creates an issue at evaluation time: since the scene scale predicted by the model generally differs from the ground-truth scale, the ground-truth target poses cannot be used directly for rendering. We therefore align the target camera poses to the predicted coordinate frame following the protocol of NoPoSplat (Ye et al., 2024), and apply the same alignment to all pose-free methods for fairness.

B.3 Applications

Camera-Controllable VDM. We evaluate the quality of the generated novel-view videos on VBench (Huang et al., 2024). We first render novel-view videos with T4GS from monocular inputs and then refine them with the VDM. Adapting the evaluation protocol of NeoVerse (Yang et al., 2026), we evaluate 100 unseen DAVIS videos (Pont-Tuset et al., 2017), each with a wobble trajectory derived from its estimated camera motion. All methods are evaluated using the same source videos and target camera trajectories. As baselines, we use TrajectoryCrafter (Yu et al., 2025), which is conditioned on point clouds, and NeoVerse (Yang et al., 2026), which is conditioned on per-pixel 4D Gaussians.

Point Tracking with 4D Feature Fields. We use point tracking as a probe for the temporal consistency of our feature field: if nearest-neighbor matching between per-frame features reproduces the ground-truth tracks, the lifted features are consistent across time. We evaluate on the DAVIS and RGB-Stacking subsets of TAP-Vid (Doersch et al., 2022) and report $< \delta_x$, the fraction of points whose predicted position lies within x pixels of the ground truth. We compare the features before and after lifting for two widely adopted backbones, DINOv3 (Siméoni et al., 2025) and the tracking features of VGGT (Wang et al., 2025a).

Dynamic Scene Understanding. We evaluate on the DAVIS video segmentation dataset (Pont-Tuset et al., 2017), using LSeg (Li et al., 2022) as the backbone and each sequence name as the text label for its foreground object. Since LSeg fails entirely on some sequences, predicting every pixel as background, we restrict evaluation to sequences where LSeg produces a non-trivial foreground. This setting favors LSeg, since the sequences are selected by its own predictions. We compare against the pose-free feed-forward 3D feature-lifting methods LSM (Fan et al., 2024) and C3G (An et al., 2026). We do not compare per-scene optimization method such as Zhou et al. (2025) since they require ground truth poses. For every method, we lift the LSeg features, render them back to the input frames, and compute segmentation masks, reporting mIoU and mAcc.

Table 9: **Analysis of opacity persistence.** We conduct the analysis of Gaussian opacity value with varying the target time. The table values report the average Gaussian opacity.

Methods	Frame index														
	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
4DGT	0.002	0.008	0.024	0.067	0.176	0.381	0.657	0.867	0.692	0.419	0.194	0.081	0.03	0.01	0.003
NeoVerse	0	0	0	0	0	0	0	0.999	0	0	0	0	0	0	0
MoVieS	0.136	0.122	0.151	0.161	0.198	0.316	0.438	0.585	0.501	0.365	0.202	0.157	0.135	0.119	0.124
Ours	0.867	0.859	0.855	0.858	0.851	0.852	0.845	0.842	0.835	0.825	0.817	0.815	0.808	0.805	0.801

Table 10: **Analysis of tracking consistency.** We conduct the analysis of movement of Gaussians compared with 2D Tracker. The table values are the EPE metric (px) between Gaussians and trackers.

Methods	Frame index														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4DGT	0.89	1.62	2.37	3.23	4.11	5.34	6.99	8.73	10.68	12.58	14.88	17.61	20.69	23.79	27.11
NeoVerse	1.11	1.6	-	-	-	-	-	-	-	-	-	-	-	-	-
MoVieS	0.98	1.95	2.9	4.23	5.48	6.8	8.16	9.44	10.81	12.69	14.54	15.41	16.82	17.84	19.07
Ours w/o $\mathcal{L}_{\text{track}}, \mathcal{L}^{\text{3D}}_{\text{flow}}$	1.28	1.54	1.67	1.87	2.02	2.56	2.84	3.01	3.24	3.51	4.04	4.23	4.29	4.53	6.00
Ours	0.85	0.99	1.15	1.19	1.44	1.65	1.73	1.79	1.92	2.12	2.63	3.09	3.38	3.34	3.02

C Analysis

C.1 Analysis of Opacity Persistence

We analyze opacity persistence to expose the view-dependent bias of per-pixel methods, in which each Gaussian is active only near the frame that produced it and therefore cannot account for global scene motion. The analysis is conducted on DyCheck (Gao et al., 2022). We sample 16 input views from a 32-frame sequence with a stride of 2, take the 16th frame of the sequence as the reference view, and average the opacity of all Gaussians decoded from it while varying the target time. Since T4GS does not produce frame-specific Gaussians, we instead measure the average opacity of the full Gaussian set over the same range of target times.

As shown in Fig. 1-(c) and Tab. 9, the opacity of per-pixel methods follows a bell-shaped curve, peaking near the reference timestamp and falling off sharply as the target time moves away, whereas T4GS shows no such decay. This confirms the *view-dependent bias*: each per-pixel Gaussian covers only a short interval around its own frame, which suffices to interpolate between nearby frames, but leaves regions visible only in distant frames unreconstructed. In contrast, our Gaussians persist across the whole sequence, so content observed at any timestamp remains available at every target time. Since a single Gaussian must serve every timestamp, it has to encode the motion of its region over the full sequence rather than a snapshot of one frame, yielding a globally consistent Gaussian field.

C.2 Analysis of Tracking Consistency

We further analyze tracking consistency to assess how well each Gaussian captures the *global scene motion*. Ideally, the mean of each Gaussian should follow the 3D point it represents. Otherwise, it drifts away from its region, and Gaussians decoded from different frames end up at different locations even though they describe the same region, which produces ghosting artifacts.

We evaluate this on DyCheck (Gao et al., 2022) with the same 16 input views, taking the first frame as the query frame. We decode the Gaussians of that frame and compute their tracking field following Shape of Motion (Wang et al., 2025b). A Gaussian that captures the motion of its region should trace the same trajectory as a 2D point track, so we compare the tracking field against pseudo ground-truth tracks from CoWTracker (Lai et al., 2026) using the end-point error (EPE), keeping only points whose product of visibility and confidence exceeds 0.5. Since T4GS decodes a global Gaussian set at each target time rather than frame-specific Gaussians, we compute the tracking field of the full Gaussian set over the same range of target times.

Table 11: **Computational Costs.** † denotes the computational cost of 1 target timestamp.

Methods	# of Gaussians ↓	Inference time (s) ↓	Rendering Speed (FPS) ↑	Peak Memory(GiB) ↓	Gaussian Memory (MiB) ↓
4DGT	272K	0.169	278	7.4	69.6
MoVieS	802K	0.381	40	8.0	205.3
NeoVerse	802K	0.438	313	11.5	205.3
Ours†	**2K**	**0.101**	**651**	**6.8**	**0.1**
Ours	**2K**	0.154	**651**	**6.8**	**0.1**

As shown in Fig. 1-(d) and Tab. 10, the Gaussians of per-pixel methods drift increasingly far from the pseudo ground-truth tracks as the target time advances, whereas T4GS exhibits substantially less drift. We further evaluate a variant of T4GS trained without any motion loss, *i.e.*, without both $\mathcal{L}_{\text{track}}$ and $\mathcal{L}^{\text{3D}}_{\text{flow}}$, under the same protocol. Since this variant never uses CoWTracker during training, it allows a more rigorous comparison that is independent of the tracker used for evaluation. Even without any supervision on Gaussian trajectories, this variant still drifts far less than per-pixel methods, showing that globally consistent trajectories emerge from the compact query-based design itself, while the motion losses further refine them. In contrast, the drift of per-pixel methods explains the ghosting they produce under large motion. Note that NeoVerse also explicitly supervises Gaussian motion, but it propagates each Gaussian only to the adjacent frame, so its trajectories span two frames and never extend across the sequence.

C.3 Computational Costs

We compare the computational cost of T4GS against per-pixel 4D Gaussian splatting methods. As shown in Tab. 11, we measure the cost on 16 input frames decoded at 31 target timestamps, reporting the number of Gaussians, the inference speed and peak memory of the full feed-forward pass, and the rendering speed and memory of the resulting Gaussians. T4GS is the most efficient across all measures, owing to the small number of Gaussians it produces. The cost also grows slowly with the number of target timestamps, since decoding all 31 targets is only marginally more expensive than decoding one, confirming that repeated passes through our lightweight decoder add little to the total inference time.

C.4 Analysis of Direction of Gaussian Movement

Table 12: **Direction of Gaussian Movement.**

Method	Cosine-Similarity $\Delta t = 2$	$\Delta t = 4$	$\Delta t = 8$
4DGT Xu et al. (2025)	<u>0.34</u>	0.20	0.13
MoVieS Lin et al. (2026)	0.31	<u>0.40</u>	<u>0.28</u>
NeoVerse Yang et al. (2026)	0.21	0.24	<u>0.28</u>
Ours	**0.54**	**0.48**	**0.36**

We additionally analyze the direction of Gaussian motion. Following the tracking evaluation protocol of Tab. 3, we compute the cosine similarity between the displacement of each Gaussian across consecutive frames and the ground-truth motion direction. As shown in Tab. 12, T4GS follows the motion direction more accurately than per-pixel methods, indicating that it captures the global scene motion more faithfully. Notably, this holds even though the direction toward the immediately next frame is relatively easy to estimate for per-pixel methods.

D Additional Experiments

D.1 Ablation Study on the Number of Learnable Tokens

We ablate the number of learnable tokens, training each variant for 10K steps. As shown in Tab. 13, 2K Gaussians provide the best trade-off between performance and efficiency: reducing the number to 1K degrades rendering quality, while increasing it to 4K yields only marginal gains. Although more Gaussians can improve rendering quality, we deliberately use a compact set of tokens to impose an explicit representation bottleneck. This encourages each Gaussian to capture meaningful scene content and motion rather than overfit to individual views or timestamps, whereas an excessive number of Gaussians gives the model more freedom to explain each observation independently, potentially weakening cross-view and temporal consistency.

Table 13: **Ablation studies on number of learnable tokens.**

Number of learnable tokens	NVIDIA PSNR ↑	NVIDIA LPIPS ↓	TUM-Dynamics PSNR ↑	TUM-Dynamics LPIPS ↓	ADT (PCK) $\Delta t = 2$	ADT (PCK) $\Delta t = 4$
1,024	20.40	0.454	19.88	0.357	27.9	18.8
2,048	**20.66**	**0.388**	20.33	0.284	**34.2**	**22.8**
4,096	20.64	0.392	**20.39**	**0.279**	29.9	19.8

Table 14: **Ablation studies on dataset curation.**

Stage1 Training	Stage2 Static	Stage2 Multi-view	Multi-view Coverage Sampling	DyCheck PSNR ↑	DyCheck SSIM ↑	DyCheck LPIPS ↓	TUM-Dynamics PSNR ↑	TUM-Dynamics SSIM ↑	TUM-Dynamics LPIPS ↓	ADT (PCK) $\Delta t = 2$	ADT (PCK) $\Delta t = 4$
✗	✓	✓	✓	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
✓	✗	✓	✓	15.78	0.394	0.448	19.82	0.631	0.296	28.0	19.9
✓	✓	✗	✓	15.22	0.376	0.474	**20.38**	0.647	0.288	26.3	15.5
✓	✓	✓	✗	14.99	0.364	0.482	20.08	0.632	0.314	22.6	14.0
✓	✓	✓	✓	**15.89**	**0.402**	**0.439**	20.33	**0.647**	**0.284**	**34.2**	**22.8**

D.2 Ablation Study on Dataset Curation

We ablate our data curation in Tab. 14, covering the static pre-training stage, the dataset combination in the joint training stage, and multi-view coverage sampling.

Static Pre-training. Removing the static pre-training and training directly on the mixture of static and dynamic data makes training unstable and eventually leads to collapse. Without pre-training, each query is free to move anywhere as the target time varies, and nothing anchors it early in training. Pre-training on static data, where the rendering must be identical at any timestamp, provides this anchor before dynamic content is introduced.

Dataset Combination. Without static data in the joint stage, the model struggles to recover static geometry, which is ill-posed from dynamic video alone, and static regions jitter as the target time varies. Without multi-view data, dynamic content must be learned from monocular video alone, leaving the geometry of moving objects underconstrained, whereas observing the same timestamp from several views pins each object to a single consistent position. Both sources are therefore necessary for learning dynamic scenes.

Multi-view Coverage Sampling. Drawing the context from a single trajectory leaves large parts of the other views unobserved, so their renderings contain holes and little multi-view signal reaches the model. Multi-view coverage sampling covers these regions, allowing the multi-view loss to provide meaningful supervision.

E Additional Qualitative Results

E.1 Additional Attention Visualization on T4GS.

We provide additional attention-map visualizations extending Fig. 4. As shown in Fig. 7, our model preserves its emergent attention properties (Hong et al., 2022b;a; Han et al., 2025b) consistently across diverse input images, token positions, and timestamps. Examining the layer-wise attention patterns, we observe a clear functional specialization: Layer 1 attends broadly and consistently to spatially corresponding regions regardless of timestamp, while Layer 2 exhibits temporally aware attention, where the attention intensity increases for tokens closer to the target timestamp and diminishes for those further away. This suggests a natural division of labor within the decoder, where Layer 1 primarily captures spatial correspondences (Hong & Kim, 2021; Cho et al., 2022; 2021; An et al., 2025) and Layer 2 progressively incorporates temporal context to refine the representation toward the target time moment.

E.2 Qualitative Results of VDM.

In Fig. 8, we present qualitative comparisons of camera-controllable video generation on DAVIS (Pont-Tuset et al., 2017) against NeoVerse (Yang et al., 2026) and TrajectoryCrafter (Yu et al., 2025). Under novel camera trajectories, NeoVerse inherits holes and ghosting artifacts from its

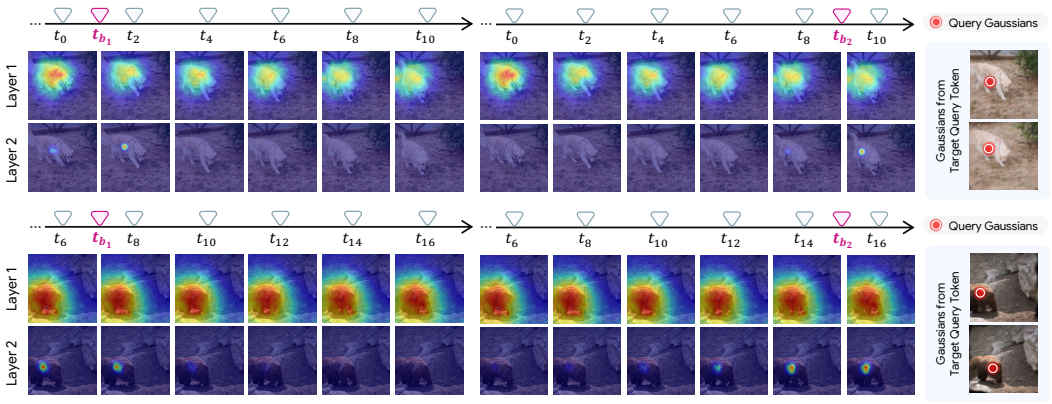


Figure 7: **Attention map visualization in dynamic regions.**

per-pixel Gaussian anchors, which can lead to hallucinated content and fragmented moving objects. TrajectoryCrafter produces plausible novel views but exhibits structured artifacts and distorted object boundaries under large viewpoint changes. In contrast, our VDM leverages globally consistent 4D Gaussian anchors, preserving the structure of both static and dynamic regions while generating coherent motion along the target camera trajectory. Consequently, our method produces temporally more consistent videos with fewer holes, ghosting artifacts, and object discontinuities.

E.3 Qualitative Results of Feature Lifting.

Fig. 9 shows qualitative results of our 4D feature field, comparing DINOv3 (Siméoni et al., 2025) features lifted by T4GS with the original DINOv3 features on DAVIS (Pont-Tuset et al., 2017). Our lifted features exhibit fewer artifacts and greater consistency across views and time, since aggregating multi-frame features suppresses artifacts that appear in individual frames. Moreover, as the features are rendered from Gaussians, they can be produced at arbitrary resolution, yielding finer-grained features than the patch-level DINOv3 outputs. As a result, our feature field better distinguishes semantic regions while remaining temporally consistent.

F Additional Related Works

Towards Addressing View-dependent Bias. The tendency of reconstruction models to overfit to input views is a long-standing challenge in 3D reconstruction, observed in both per-scene optimization (Kim et al., 2022; Niemeyer et al., 2022; Li et al., 2024b) and feed-forward settings (Hong et al., 2024; Huang & Mikolajczyk, 2025). Since photometric reconstruction objectives alone provide insufficient signal for the model to learn globally consistent geometry, existing works incorporate geometric regularizations such as monocular depth priors (Song et al., 2023; Li et al., 2024b), opacity constraints (Kim et al., 2022; Yang et al., 2023), novel view self-supervision (Niemeyer et al., 2022; Kwak et al., 2023; Jung et al., 2025). This challenge is especially pronounced in dynamic scenes, where moving regions inherently break multi-view consistency, necessitating expert geometric models (Wang et al., 2025b;e) or explicit motion constraints (Lei et al., 2025). On the architectural side, recent works show that reducing Gaussian redundancy itself mitigates view-dependent bias: Miao et al. (2025) and Wang et al. (2025d) estimate Gaussians per voxel only when needed, while C3G (An et al., 2026) predicts a compact set of Gaussians through learnable query token.

Additional 4D Reconstruction with Gaussians. Within per-scene optimization, recent works curb the growth of the Gaussian count by clustering or merging primitives so that each region is represented compactly (Lee et al., 2025a), or regularize the optimization to recover motion more reliably (Li et al., 2024a; Liu et al., 2026). Both remain bound by per-scene optimization, which requires lengthy fitting for every new scene. Feed-forward reconstruction has also been demonstrated for driving scenes (Yu et al., 2026; Kong et al., 2026) and human subjects (Tu et al., 2026), but these methods are tailored to a single domain. The concurrent NoPo4D (Balice et al., 2026) per-



Figure 8: **Qualitative comparison of camera-controllable VDMs.**

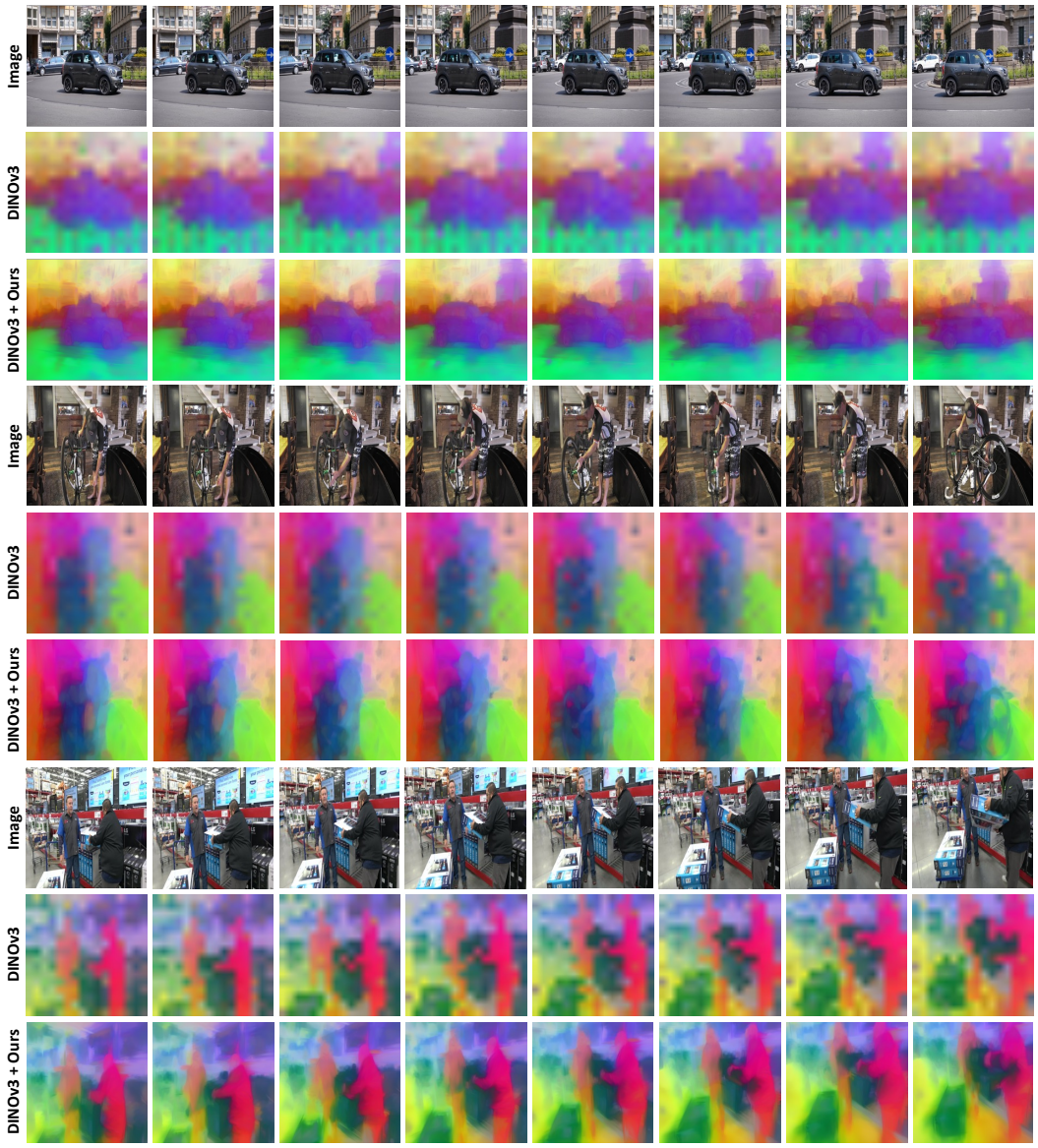


Figure 9: **Qualitative results of the DINOv3 4D feature field.**

forms 4D Gaussian splatting from multi-view rather than monocular video, while we use multi-view data only as training supervision signal, not input and take monocular video as input at inference.

G LIMITATIONS

Since our model reconstructs dynamic scenes solely from the input view distribution, it can only generate Gaussians for regions observed at least once across the context frames. Consequently, when rendering from extreme viewpoints far from the context cameras, or in regions fully occluded in all input views, the model produces holes with zero-valued pixels. While our camera-controllable VDM can fill these regions, it may hallucinate plausible content rather than recover their true appearance. In addition, our model cannot extrapolate beyond the temporal range of the input views: since T4GS is trained only on interpolated timestamps, it fails to predict accurate future Gaussians, which in turn prevents the diffusion module from synthesizing future frames, as it relies on the target views rendered by T4GS.