

---

# ENHANCED VELOCITY FIELD MODELING FOR GAUSSIAN VIDEO RECONSTRUCTION

---

Zhenyang Li<sup>1\*</sup>, Xiaoyang Bai<sup>1\*</sup>, Tongchen Zhang<sup>2</sup>, Pengfei Shen<sup>1</sup>, Weiwei Xu<sup>2</sup>, Yifan Peng<sup>1†</sup>

<sup>1</sup>The University of Hong Kong <sup>2</sup>State Key Laboratory of Computer-aided Design & Computer Graphics, Zhejiang University

## ABSTRACT

High-fidelity 3D video reconstruction is essential for enabling real-time rendering of dynamic scenes with realistic motion in virtual and augmented reality (VR/AR). The deformation field paradigm of 3D Gaussian splatting has achieved near-photorealistic results in video reconstruction due to the great representation capability of deep deformation networks. However, in videos with complex motion and significant scale variations, deformation networks often overfit to irregular Gaussian trajectories, leading to suboptimal visual quality. Moreover, the gradient-based densification strategy designed for static scene reconstruction proves inadequate to address the absence of dynamic content. In light of these challenges, we propose a flow-empowered velocity field modeling scheme tailored for Gaussian video reconstruction, dubbed *FlowGaussian-VR*. It consists of two core components: a *velocity field rendering (VFR)* pipeline which enables optical flow-based optimization, and a *flow-assisted adaptive densification (FAD)* strategy that adjusts the number and size of Gaussians in dynamic regions. We also explore a *temporal velocity refinement (TVR)* post-processing algorithm to further estimate and correct noise in Gaussian trajectories via extended Kalman filtering. We validate our model’s effectiveness on multi-view dynamic reconstruction and novel view synthesis with multiple real-world datasets containing challenging motion scenarios, demonstrating not only notable visual improvements (over 2.5 dB gain in PSNR) and less blurry artifacts in dynamic textures, but also regularized and trackable per-Gaussian trajectories.

**Keywords** Dynamic Scene Reconstruction · Gaussian Video · Velocity Field.

## 1 Introduction

3D scene reconstruction is a crucial step for crafting immersive and realistic experiences in a wide range of augmented and virtual reality (AR/VR) applications [1, 2], including cinema-level virtual modeling [3, 4], autonomous driving [5], robotics [6, 7], and medical surgery [8], as it accurately represents the spatial layouts of 3D environments and readily enables users to interact with and even manipulate scene contents. In this domain, 3D Gaussian Splatting (3DGS) [9] has recently demonstrated remarkable progress in both reconstruction quality and rendering efficiency, outperforming established neural radiance field (NeRF) variants [10]. Noteworthy progress has also been observed in the more challenging dynamic scene reconstruction task [11, 12, 13], where Gaussian Splatting (GS) is also being increasingly adopted to modeling dynamic scenes. Existing dynamic GS frameworks mainly follow two schemes: *deformation-based* methods learn a deformation field [14] to represent 3D motions, while *trajectory-based* methods explicitly model the geometry transformation between consecutive frames [15, 16]. The former scheme, leveraging the exceptional representation capability of neural networks, has generally demonstrated better reconstruction quality across diverse datasets.

We attribute the superior rendering quality of deformation-based methods to two key factors: minimal motion between frames which facilitates model convergence, and large Gaussian population which yields sufficient optimizable parameters during training. However, as shown in Fig. 1, Gaussian trajectories predicted by the deformation network often lack local consistency and struggle to align with the true dynamics of moving objects, leading to suboptimal

---

\*These authors contributed equally.

†e-mail: evanpeng@hku.hk

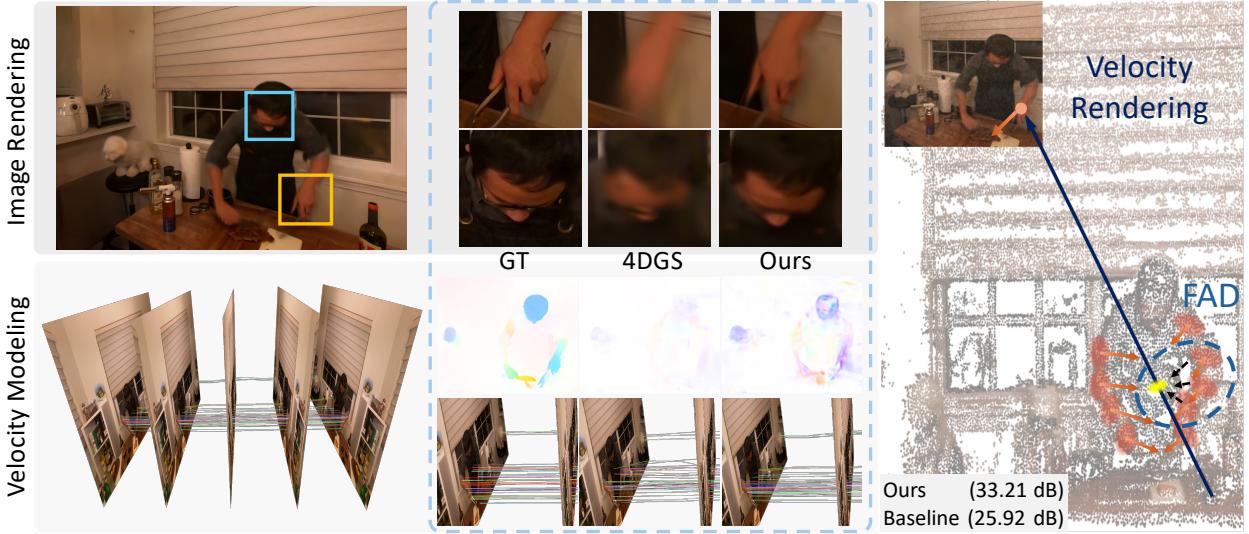


Figure 1: **Left:** Deformation-based 4DGS encounters difficulties in reconstructing scenes and rendering novel views under challenging conditions, such as significant motion and other complex dynamics. Our FlowGaussian-VR exhibits commendable performance on given scenes (e.g., “cut-roasted-beef”). **Middle:** We compare the ground-truth optical flow, the deformation network of baseline (4DGS), and the velocity field rendered by our method. **Right:** We render the velocity field for scene Gaussians, constrain it with flow-based losses, and employ the FAD strategy to add Gaussians for dynamic objects in the canonical space.

video reconstruction quality and restricting the application of these methods to high-quality video datasets with fewer camera fluctuations, small-scale motion, and clear content. Stable and physically accurate Gaussian trajectories ensure both rendering and semantic consistency in dynamic regions during novel view synthesis, resulting in photorealistic outcomes. Therefore, their prospects in fields such as autonomous driving and robotics, where critical applications in semantic segmentation and object tracking demand accurate reconstruction of object trajectories [17], appear limited.

As optical flow [18] has been proven effective in enhancing the camera pose estimation consistency in monocular video reconstruction, we adopt a similar approach to supervising the motion of 3D Gaussians. Nonetheless, we observe that such a methodology cannot be trivially applied to Gaussian representations for two primary reasons. Firstly, the centers of 3D Gaussians do not align with object surfaces. This misalignment may cause conflicts between velocity field-based learning and photometric supervision, leading to misguided optimization. Secondly, the default densification strategy of dynamic GS, inherited from 3DGS, relies on the gradient of photometric losses [11]. We find experimentally that this strategy struggles to capture regions with substantial motion and does not effectively align with the dynamic information present in the scene. Therefore, how to overcome these problems and incorporate motion cues into GS frameworks becomes a critical challenge.

To this purpose, we propose *FlowGaussian-VR*, a velocity field modeling pipeline that incorporates optical flow as the ground truth for 3D Gaussian dynamics to enhance their temporal controllability and interpretability. We render 2D velocity fields with differentiable rasterization for each Gaussian and supervise them using outputs from models such as RAFT [19]. Consequently, we introduce three losses based on the rendered velocity fields to facilitate the alignment of Gaussian attributes with real-world dynamics captured by estimated optical flows. The *windowed velocity error* ( $\mathcal{L}_{\text{win}}$ ) employs a temporal sliding window to optimize rendered velocity fields across multiple frames. The *flow warping error* ( $\mathcal{L}_{\text{warp}}$ ) computes the discrepancy between rendered images and warped ground truths from the next frame. Additionally, we leverage SAM-v2 [20] to segment dynamic regions for each video and compute the *dynamic rendering loss* ( $\mathcal{L}_{\text{dyn}}$ ) to further refine the model’s representation of dynamic scene components.

While velocity field-based optimization effectively controls the trajectories of Gaussian centers, the combination of flow-based losses and the original photometric loss may still fall behind object motion when the scene undergoes significant changes between frames. To mitigate this issue, we introduce a *flow-assisted adaptive densification* (FAD) strategy that adds Gaussians by identifying challenging dynamic regions on the 2D frame instead of cloning and splitting existing ones. Compared to the conventional densification algorithm, this design helps to complete scene components that are missing from initialization based on a single video frame. Finally, we explore the possibility of leveraging the

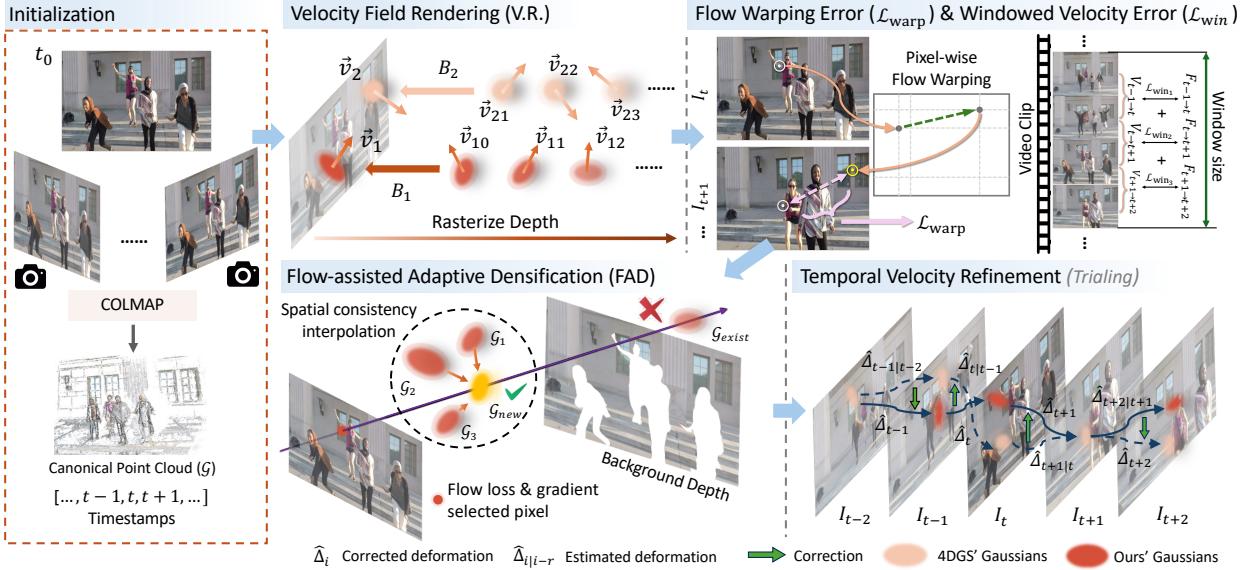


Figure 2: **Overview of our FlowGaussian-VR pipeline.** The velocity attribute is introduced into each Gaussian within the dynamic 3D GS pipeline to ensure that the motion of Gaussians closely mimics real-world physical motion. To achieve this, we impose several constraints on the rendered velocity:  $\mathcal{L}_{\text{warp}}$  ensures temporal color consistency by aligning the predicted velocity with sequential frame data;  $\mathcal{L}_{\text{win}}$  maximizes the supervision effect of ground truth optical flow on velocity predictions;  $\mathcal{L}_{\text{dyn}}$  enhances rendering quality and velocity accuracy for dynamic regions. Finally, to globally refine the predicted velocity across the temporal domain, we explore a TVR post-processing strategy, which integrates the traditional Extended Kalman Filter (EKF).

Extended Kalman Filter (EKF) [21] as a post-processing *temporal velocity refinement (TVR)* technique for Gaussian motion estimation.

We conduct a comprehensive evaluation of our model on challenging datasets: Nvidia-long [22], and Neu3D [23]. Experimental results indicate that our FlowGaussian-VR pipeline, when integrated with the 4DGGS [14] baseline, improves the overall PSNR of novel view rendering by approximately 2.5 dB across challenging scenes, with a gain of over 2.0 dB on dynamic regions. Additionally, we compare our method with alternative approaches following the same scheme, namely 4D-GS [24] and SC-GS [25], yielding superior performance. We further demonstrate that the TVR strategy effectively smoothens Gaussian trajectories and aligns them more closely with ground truth optical flows, substantiating the potential of a multi-stage learning framework for dynamic GS.

In summary, our contributions are as follows:

- We propose a velocity field modeling pipeline, *FlowGaussian-VR*, which facilitates optical flow-based optimization via velocity field rendering in dynamic Gaussian splatting models to enhance the temporal behavior of Gaussian instances.
- We introduce the FAD strategy to address challenges of optimizing Gaussians in dynamic scenes with abrupt changes. By integrating flow-based supervision into its pipeline, FAD enhances the robustness of dynamic GS video processing.
- We select challenging datasets comprising real-world captured videos to experimentally demonstrate the superior performance of FlowGaussian-VR on complex dynamic scenes.

## 2 Related Work

### 2.1 Video Reconstruction

Neural video reconstruction methods [26, 27] have significantly advanced the modeling and rendering of dynamic scenes [28, 29], with progress in monocular videos [30, 31], time-synchronized videos [32], sparse camera views [33], and stereo cameras [34]. In dynamic settings, NeRF-based approaches have made notable strides, particularly for video reconstruction. Some models represent dynamic scenes as time-conditional radiance fields [35, 36, 37], or use

deformation networks separate from the canonical radiance field [38]. Others, such as DyNeRF [23], Nerfies [39], and HyperNerf [40], encode temporal information into latent representations. For trajectory learning, approaches like DynamicNeRF [41] and NSFF [42] regress optical flow from spacetime coordinates, while DynIBaR [43] models trajectory fields using DCT bases. However, NeRF representations are computationally intensive, posing challenges for reconstructing complex dynamic scenes in videos.

## 2.2 Dynamic 3D Gaussian Splatting

The emergence of 3DGs [9, 44, 45] has significantly improved training speeds and reduced memory usage in video reconstruction models. Dynamic 3DGs variants primarily fall into two categories: those with time-dependent Gaussian parameters [11, 12] and those employing deformation fields [46, 13]. The latter offers a lightweight network, an initial point cloud, and fast training. These advantages make them well-suited for 3D video streaming, compression, and web-based demonstrations, while directly applying the previous type of dynamic 3DGs in these applications may struggle to handle the substantial storage concern. However, many 3DGs methods rely on datasets with minimal motion blur, slow object motion, and without flickers. When interference factors are present, dynamic 3DGs [11] performs multiview 3D optimization on the point clouds of each frame, but the high number of iterations significantly increases training time. A more efficient approach applies deformation to the initial frame instead. Yet, optimizing the deformation field solely based on point cloud and temporal features [47] lacks explicit constraints [48], reducing its controllability and adherence to physical motion laws. Without well-defined motion dynamics, the model struggles to accurately capture per-instance motion within the point cloud.

## 2.3 Flow-assisted Gaussians

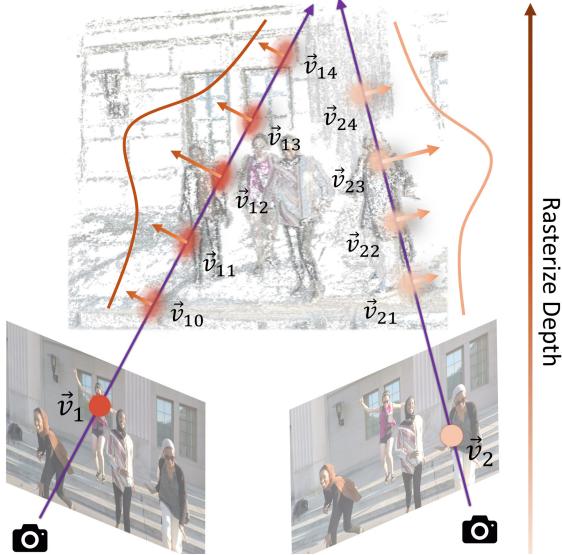
Several studies [49, 50, 51, 52] have explored integrating dynamic flows, such as velocity fields, with dynamic GS. MODGS [53] leverages off-the-shelf depth [54] and flow [19] estimators to compute the 3D flow between frames, initializing the deformation network for point cloud prediction. However, it is constrained by the limitations of depth estimation and struggles to adapt to Gaussian changes during densification and pruning, reducing the effectiveness of flow supervision. Gaussian-Flow [15] and Shape of Motion [16] improve temporal motion modeling by introducing motion bases into the deformation network but avoid directly supervising Gaussian parameters or trajectories. These methods also overlook the impact of deformation updates on the rasterizer’s rendering, as Gaussian attributes are indirectly modified through geometric reshaping [55] rather than being directly controlled. Allowing the rasterizer to render velocity fields provides direct supervision of Gaussian properties, enhancing both deformation controllability and temporal motion representation. More relevant to our research is MotionGS [50], which decouples foreground Gaussians from background ones and supervises object motion with pseudo 2D optical flow. However, we observe in practice that these Gaussians still undergo large, erratic displacements during optimization, causing numerous background Gaussians to drift into the foreground and corrupt its rendering. Furthermore, MotionGS only controls the center position of each Gaussian, while overlooking and constraining the optimization space of other attributes. Such a design prevents it from reaching globally optimal solution, especially in fast-moving scenes. Our method overcomes this drawback by directly rasterizing velocity fields from dynamic scenes.

## 3 Gaussian Video with Velocity Field Modeling

As evidenced in Fig. 1, although deformation-based dynamic GS methods such as 4DGS [14] can generate visually appealing rendering results, their per-Gaussian motion estimation lacks constraints and local coherence. In light of this, we introduce FlowGaussian-VR, whose pipeline is illustrated in Fig. 2. It comprises a dynamic GS backbone (Sec. 3.1), multiple flow-based constraints empowered by velocity field rendering (Sec. 3.2 & 3.3), and the flow-assisted adaptive densification strategy (Sec. 3.4). The proposed pipeline centers around the allocation of 2D velocity attributes (e.g., 2D velocity vector) to each Gaussian point and the subsequent modeling of motion dynamics with this enhanced mapping. We name this mechanism *velocity field modeling* throughout the manuscript. In addition, we discuss our exploration on the Extended Kalman Filter (EKF)-aided temporal velocity refinement post-processing in Sec. 5 and the supplementary material.

### 3.1 Preliminary: 4D Gaussian Splatting (4DGS)

In 4DGS [14], a dynamic 3D scene is represented by a set of Gaussians  $\mathcal{G}$ . Each Gaussian in the scene is defined as  $G_i = (\mu_i, \Sigma_i, c_i, \sigma_i)$ , with  $\mu_i \in \mathbb{R}^3$  being the center coordinates,  $\Sigma_i \in \mathbb{R}^{3 \times 3}$  being the covariance matrix controlling the spread and orientation,  $c_i \in \mathbb{R}^3$  being the color representation, and  $\sigma_i \in [0, 1]$  being the opacity value. For simplicity of presentation, we omit the Gaussian index  $i$  hereafter. To capture temporal changes, 4DGS applies a deformation field  $D(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R} \rightarrow \mathbb{R}^3$  to each Gaussian center  $\mu_0$  in the canonical space, predicting its position at time  $t$



**Figure 3: Velocity Rendering in 3D Gaussian Rasterization.** Gaussian velocities are alpha-blended along each projection ray. Through multi-view optimization, these Gaussian velocities progressively converge toward the actual motion of the object.

as  $\mu_t = \mu_0 + D(\mu_0, t)$ . The deformation model  $D$  is trained end-to-end with the Gaussian parameters using image losses. During rendering,  $\mu_t$  is projected onto the 2D plane as  $\mu'_t$  and the intensity at pixel  $\mathbf{p}$  is computed following the formulation in 3DGS:

$$I(\mathbf{p}) = \sum_{i=1}^N \alpha_i c_i \prod_{j=1}^i (1 - \alpha_j), \quad (1)$$

where  $\alpha_i = \sigma_i \cdot \exp \left[ -\frac{1}{2} (\mathbf{p} - \mu'_i)^T \Sigma_i^{-1} (\mathbf{p} - \mu'_i) \right]$ .

Inspired by the alpha blending technique in Eq. 1, we propose the velocity field rendering algorithm below to obtain 2D representations of scene motion through differentiable rasterization.

### 3.2 Velocity Field Rendering

Given the point cloud  $\mathbf{P}_0$  reconstructed by COLMAP [56] and two adjacent timestamps  $\{t, t + \delta t\}$ , we obtain the velocity field in a manner similar to the pixel color rendering in Eq. (1) by projecting the deformed point clouds,  $\mathbf{P}_t$  and  $\mathbf{P}_{t+\delta t}$ , onto the pixel plane, and calculating their displacement over  $\delta t$ .

Specifically, we first transform a 3D point cloud  $\mathbf{P}$  from world coordinates to camera coordinates by applying the camera's extrinsic parameters, i.e. the rotation matrix  $\mathbf{R}$  and translation vector  $\mathbf{T}$ . Then, we project the 3D camera coordinates onto the 2D image plane by utilizing the camera's intrinsic matrix  $\mathbf{K}$ , which depends on the focal lengths and optical center of the camera. Ultimately, the 2D pixel coordinates  $\mathbf{p}$  is obtained by normalizing the homogeneous coordinates. In such a way, we enable the rendering of 2D velocity fields from the velocity attributes of Gaussians and the subsequent optimization of Gaussian parameters to capture motion dynamics effectively, completing the velocity field modeling pipeline.

**Velocity Field Rasterization.** Each Gaussian in the 3D point cloud is assigned a velocity vector that represents its movement between adjacent frames (Fig. 3). The projected 2D velocity vector allows us to rasterize the temporal motion information at each pixel, akin to color rasterization in the original 3DGS formulation, and to enable a more explicit control of Gaussian attributes by the deformation network. Analogous to Eq. (1), the velocity field  $V(\mathbf{p})$  rendering process is formulated as:

$$V(\mathbf{p}) = \sum_{i=1}^N \alpha_i v_i \prod_{j=1}^i (1 - \alpha_j). \quad (2)$$

By definition, the projected velocity serves as a reasonable estimation of the rendered optical flow, assuming that a pixel is predominantly influenced by Gaussians that contribute to its color, and its radiance is carried away by the motion of those Gaussians. Therefore, the velocity field on the pixel is a sum of the velocity of dominant Gaussians, weighed by their densities and visibilities. Alternatively, we may render two consecutive frames and apply optical flow estimation on the outcome directly. However, both experimental findings and theoretical analysis suggest that it is not as effective as our splatting-based approach. **Additional details are provided in the supplementary materials.**

The newly introduced velocity attributes of Gaussians can be optimized via backpropagation through the rasterization process. To utilize this property and optimize the velocity on Gaussians, we design specific losses for velocity field rendering to complement the conventional photometric loss.

### 3.3 Losses on Rendered Velocity

**Windowed Velocity Error ( $\mathcal{L}_{\text{win}}$ ).** To refine the rendered velocity field at each iteration, we employ a temporal sliding window to sample multiple video frames during training. This approach ensures sufficient supervision and improves the model’s consistency in predicting point cloud deformations across different timestamps. *This sliding window operation is applied to all losses in this subsection.* Specifically, given the rendered velocity field image  $\hat{\mathbf{V}}_i$  at timestamp  $i$ , we perform  $\tau + 1$  rendering processes within the same iteration, which generates  $\tau$  additional rendered velocity images, denoted as  $\hat{\mathbf{V}}_i(\tau) = \{\mathbf{V}_{i+k}\}_{k=1}^{\tau}$ , that span  $k$  timestamps after  $i$ . The windowed velocity error ( $\mathcal{L}_{\text{win}}$ ) is calculated between  $\hat{\mathbf{V}}_i(\tau)$  and ground truth optical flows  $\tilde{\mathbf{V}}_i(\tau)$  at those timestamps. Formally:

$$\mathcal{L}_{\text{win}} = \|\hat{\mathbf{V}}_i(\tau) - \tilde{\mathbf{V}}_i(\tau)\|_1. \quad (3)$$

This loss ensures that the predicted velocity aligns with the actual scene dynamics across multiple consecutive frames.

**Flow Warping Error ( $\mathcal{L}_{\text{warp}}$ ).** During training, the rendered velocity field registers the correspondence between RGB values of adjacent frames. To enhance this temporal correspondence, we propose the flow warping error ( $\mathcal{L}_{\text{warp}}$ ), which supervises frame-wise alignment by warping rendered RGB values in the next frame to the current frame using the rendered velocity field and minimizing its discrepancy from ground truth, as defined by:

$$\mathcal{L}_{\text{warp}} = \|W_{t+\delta t \rightarrow t}(\hat{\mathbf{I}}_{t+\delta t}, \hat{\mathbf{V}}_{t+\delta t \rightarrow t}) - \tilde{\mathbf{I}}_t\|_1, \quad (4)$$

where  $\hat{\mathbf{I}}$  and  $\tilde{\mathbf{I}}$  represent rendered and ground truth images respectively, and  $W_{t+\delta t \rightarrow t}$  denotes the warping operation from timestamp  $t + \delta t$  to  $t$ .  $\mathcal{L}_{\text{warp}}$  provides another pathway of aligning the predicted velocity field with the actual motion dynamics observed in the scene, and helps to rectify any inaccuracies in the predicted motion, thereby improving the consistency of the deformation field and enhancing the overall fidelity of reconstruction.

**Dynamic Rendering Loss ( $\mathcal{L}_{\text{dyn}}$ ).** To improve the rendering accuracy of dynamic objects in the scene, we perform dynamic foreground segmentation to isolate moving objects from the background and compute masked image error:

$$\mathcal{L}_{\text{dyn}} = \|\mathbf{I} \odot \hat{\mathbf{M}}_{\text{dyn}}, \tilde{\mathbf{I}} \odot \mathbf{M}_{\text{dyn}}\|_1, \quad (5)$$

where  $\mathbf{M}_{\text{dyn}}$  represents the dynamic foreground segmentation mask, and  $\odot$  denotes element-wise multiplication. This loss emphasizes dynamic regions in the scene and focuses the optimization process on the motion and appearance of moving objects.

### 3.4 Flow-assisted Adaptive Densification (FAD)

Examining the experimental results of the baseline reveals its inability to handle challenging scenarios with significant object dynamics, where Gaussians are often absent in such regions. While the rendering and supervision of the velocity field effectively influence the positions and parameters of existing Gaussians, regions without Gaussians remain unoptimized. To address this limitation, we retain the 3DGS densification strategy while aiding it with the FAD strategy (Fig. 4). Unlike conventional densification that clones and splits existing Gaussians, FAD leverages flow-based losses and their gradients to directly add Gaussians to dynamic regions in the canonical space. It aims to compensate for inadequate Gaussians in dynamic objects by aligning their positions with underlying motion dynamics, thereby achieving more accurate and temporally consistent representations. Specifically, it lifts prominent pixels in the velocity field, which are correlated to dynamic scene elements, into the 3D space and constructs 3D Gaussians of dynamic objects correspondingly, ultimately enhancing the model’s representation and alignment with scene motion.

**Flow-assisted Pixel Selection.** We use the loss map  $\mathcal{L}_{\text{win}}$  and the gradient map  $\nabla \mathcal{L}_{\text{win}}$  from Section 3.2 to identify pixel positions where additional Gaussians are needed. We define a threshold  $\epsilon$  to select pixel locations that exhibit sufficiently large losses and gradients:  $\mathcal{P} = \{\mathbf{p} : \mathcal{L}_{\text{win}}(\mathbf{p}) > \epsilon \text{ and } \nabla \mathcal{L}_{\text{win}}(\mathbf{p}) > \epsilon\}$ . Since the loss and gradient maps

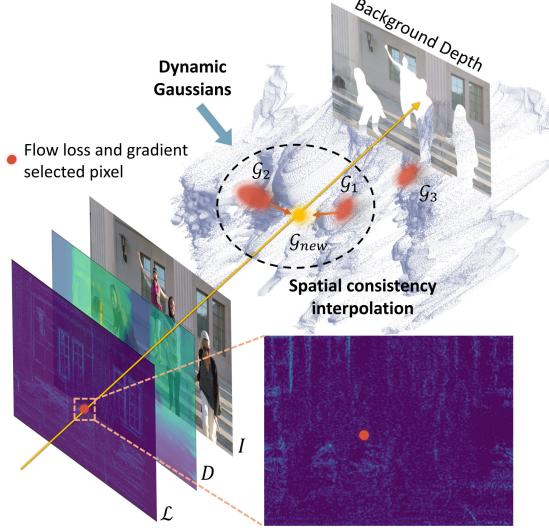


Figure 4: **Flow-assisted adaptive densification (FAD)** incorporates flow gradients and loss maps to identify pixel locations requiring densification. These pixels are lifted to the current frame’s 3D space using rendered depth and then transformed back into the canonical space via deformation-induced displacement. The attributes of these new Gaussians are calculated through kNN-weighted interpolation.

may contain non-zero pixels in the background, we employ an off-the-shelf segmentation model SAM-v2 [20] to extract and exclude background pixels from  $\mathcal{P}$ .

**Lifting to 3D Space.** We lift the filtered pixels to 3D world coordinates  $\mathcal{P}'$  using the rendered depth map  $\mathbf{Z}$ , camera intrinsics  $\mathbf{K}$ , and camera extrinsics  $[\mathbf{R}, \mathbf{T}]$ , inverting the procedure described in Sec. 3.2. To lift a point  $(x, y)$  from 2D space to 3D space, we use the intrinsic and extrinsic camera parameters, as well as the projection model. The general relationship between the 2D image coordinates and the 3D world coordinates can be expressed as:

$$\mathcal{P}' = (x_{3D}, y_{3D}, z_{3D})^T = \mathbf{K}[\mathbf{R}, \mathbf{T}] \cdot (x_{2D}, y_{2D}, 1)^T \cdot Z, \quad (6)$$

where  $(x_{2D}, y_{2D})$  are the 2D pixel coordinates in the image plane,  $(x_{3D}, y_{3D}, z_{3D})$  are the corresponding 3D coordinates in the world space,  $\mathbf{K}$  is the camera intrinsic matrix, which includes parameters such as focal length and principal point,  $Z$  is the depth value corresponding to the point in the 3D world.

This equation assumes that we have the depth value  $Z$  for the 3D point, which can be obtained through depth sensing or estimated through stereo vision techniques. Once the 3D coordinates are determined, we can perform operations such as motion tracking and dynamic scene reconstruction in 3D space.

**Farthest Point Sampling (FPS).** To keep the number of added points manageable, we employ the established FPS [57, 58] to downsample the candidates with a ratio  $r$ . FPS works by iteratively selecting points that are farthest from those already selected in a point cloud. This process ensures that the selected subset of candidates, denoted as  $\mathcal{P}''$ , spreads evenly across the space.

**Spatial Consistency Interpolation.** After downsampling, to ensure consistency of the new Gaussians’ attributes in the canonical space, we use the K-nearest neighbors (kNN) algorithm to find the set of existing Gaussians surrounding each candidate in  $\mathcal{P}''$ . To mitigate the spatial discontinuity of Gaussians parameters, we further obtain the radius  $r_c$  on the pixel plane for each new Gaussian using differentiable rasterization and lift it to 3D to determine whether the neighboring points found by kNN lie within the candidate Gaussian’s spatial range. Points outside the radius indicate that the attributes between two Gaussians may not be continuous and thus cannot be used for attribute interpolation. To be more specific, we know that the FAD process operates on the frame at the current timestamp during training. The deformation network  $D$  is achieved by transforming the point cloud  $\mathbf{P}_0$  in the canonical space to obtain the current point cloud  $\mathbf{P}_t$ .

$$\mathbf{P}_t = \mathbf{P}_0 + D(\mathbf{P}_0, t). \quad (7)$$

Given that the pixels selected by  $\mathcal{L}_{warp}$  and  $\mathcal{L}_{win}$  are projected into the 3D space through depth to obtain new Gaussians  $\mathcal{G}_f$ , we use kNN to find the top-k nearest neighbors from the original Gaussians surrounding  $\mathcal{G}_f$ . The process can be described as follows:

$$\mathcal{G}_c = \text{kNN}(\mathbf{P}_t, \mathcal{G}_f, k), \text{ if } |\mathbf{P}_t - \mathcal{G}_f| < r_c, \quad (8)$$

where  $r_c$  refers to the 3D range of each Gaussian in  $\mathcal{G}_f$  and we finally obtain  $\mathcal{G}_c$  as newly densified Gaussians.

Therefore, the new Gaussians  $\mathcal{G}_c$  added at the current moment need to be mapped back to the canonical space  $\mathcal{G}_c^{\text{cano}}$  by reversing the offset calculated through the deformation:

$$\mathcal{G}_c^{\text{cano}} = \mathcal{G}_c - D(\mathbf{P}_0, t). \quad (9)$$

Table 1: **Quantitative evaluation on the Nvidia dataset** [22]. The resolution and video length of each scene vary. “Face.” and “Pg.” refer to dynamicFace and Playground. [PSNR (DPSNR) $\uparrow$ , SSIM $\uparrow$ , LPIPS $\downarrow$ ] are reported. Same applies to Table 2.

| Scene    | 4DGS                          | 4D-GS                         | SC-GS                         | MotionGS                      | Ours                                 |
|----------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|--------------------------------------|
| Balloon1 | 20.51 (21.24) / 0.619 / 0.317 | 23.06 (21.78) / 0.740 / 0.210 | 18.17 (16.94) / 0.666 / 0.440 | 20.72 (19.97) / 0.611 / 0.472 | <b>24.50 (24.54)</b> / 0.757 / 0.290 |
| Skating  | 26.80 (16.35) / 0.885 / 0.203 | 28.06 (18.32) / 0.872 / 0.131 | 18.09 (7.86) / 0.529 / 0.535  | 23.21 (12.94) / 0.834 / 0.370 | <b>29.91 (19.56)</b> / 0.916 / 0.180 |
| Face.    | 19.47 (24.58) / 0.783 / 0.185 | 18.61 (22.40) / 0.801 / 0.153 | 12.38 (9.25) / 0.332 / 0.546  | 15.15 (20.67) / 0.560 / 0.446 | <b>21.82 (27.20)</b> / 0.854 / 0.042 |
| Jumping  | 24.33 (17.75) / 0.847 / 0.246 | 23.80 (18.25) / 0.803 / 0.199 | 13.37 (7.96) / 0.368 / 0.639  | 20.53 (14.31) / 0.771 / 0.403 | <b>27.89 (21.07)</b> / 0.887 / 0.216 |
| Truck    | 25.18 (22.19) / 0.805 / 0.279 | 24.11 (20.67) / 0.791 / 0.260 | 16.87 (12.95) / 0.486 / 0.612 | 20.47 (18.68) / 0.698 / 0.498 | <b>26.22 (23.96)</b> / 0.835 / 0.255 |
| Pg.      | 19.98 (16.65) / 0.602 / 0.319 | 21.66 (16.92) / 0.757 / 0.179 | 12.27 (13.12) / 0.197 / 0.613 | 18.07 (15.62) / 0.448 / 0.533 | <b>22.18 (18.15)</b> / 0.695 / 0.281 |
| Umbrella | 22.85 (21.24) / 0.527 / 0.416 | 24.01 (21.84) / 0.574 / 0.346 | 6.76 (4.65) / 0.000 / 0.719   | 20.80 (19.19) / 0.471 / 0.625 | <b>24.11 (22.32)</b> / 0.599 / 0.366 |
| Avg.     | 22.73 (20.00) / 0.724 / 0.280 | 23.33 (20.03) / 0.763 / 0.211 | 13.99 (10.39) / 0.368 / 0.586 | 19.85 (17.34) / 0.628 / 0.478 | <b>25.23 (22.40)</b> / 0.792 / 0.232 |

Table 2: **Quantitative evaluation on the Neu3D dataset** [23] with a resolution of  $1,352 \times 1,014$ .

| Scene   | 4DGS                          | 4D-GS                                | SC-GS                         | MotionGS                      | Ours                                 |
|---------|-------------------------------|--------------------------------------|-------------------------------|-------------------------------|--------------------------------------|
| martini | 22.47 (20.97) / 0.810 / 0.240 | 23.85 (21.22) / 0.861 / 0.219        | 13.33 (9.24) / 0.306 / 0.615  | 21.05 (20.17) / 0.805 / 0.405 | <b>24.19 (23.69)</b> / 0.856 / 0.206 |
| spinach | 24.92 (22.88) / 0.831 / 0.222 | 26.81 (26.76) / 0.833 / 0.261        | 17.11 (15.89) / 0.432 / 0.531 | 25.94 (23.20) / 0.864 / 0.333 | <b>27.06 (25.40)</b> / 0.881 / 0.186 |
| beef    | 26.61 (25.08) / 0.881 / 0.192 | 28.48 (28.79) / 0.916 / 0.163        | 16.42 (15.74) / 0.414 / 0.552 | 21.55 (20.37) / 0.832 / 0.367 | <b>29.46 (28.58)</b> / 0.924 / 0.166 |
| salmon  | 22.95 (21.45) / 0.844 / 0.209 | 24.79 (23.24) / 0.896 / 0.171        | 8.23 (6.79) / 0.186 / 0.650   | 22.49 (21.62) / 0.813 / 0.388 | <b>25.67 (24.79)</b> / 0.883 / 0.180 |
| flame   | 25.70 (23.49) / 0.851 / 0.218 | <b>29.35</b> (26.37) / 0.951 / 0.143 | 6.71 (5.98) / 0.054 / 0.654   | 25.92 (23.04) / 0.878 / 0.323 | 28.86 (26.59) / 0.927 / 0.157        |
| sear    | 26.46 (26.12) / 0.868 / 0.199 | 26.94 (29.36) / 0.914 / 0.163        | 6.72 (6.39) / 0.052 / 0.660   | 24.96 (26.13) / 0.865 / 0.347 | <b>28.54 (29.49)</b> / 0.922 / 0.156 |
| Avg.    | 24.85 (23.33) / 0.847 / 0.213 | 26.70 (25.96) / 0.895 / 0.187        | 11.42 (10.01) / 0.241 / 0.610 | 23.65 (22.42) / 0.982 / 0.361 | <b>27.30 (26.47)</b> / 0.899 / 0.175 |

## 4 Temporal Velocity Refinement (TVR) with Extended Kalman Filtering (EKF)

While deformation network-based methods can establish temporal correlations, they do not inherently ensure smooth and physically consistent motion. In light of this, we here propose a *temporal velocity refinement (TVR)* process following the “refinement as a post-processing step” philosophy, whose impact on motion prediction will be explored in the subsequent experiment section.

**Extended Kalman Filtering.** As an optimal recursive estimation technique in control theory, Kalman filter (KF) [59, 60] estimates the internal states  $\mathbf{x}$  of a linear dynamic system in the presence of noisy observations  $\mathbf{z}$ . The system is described using the state-space equations below:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \end{aligned} \quad (10)$$

where  $\mathbf{A}$  and  $\mathbf{H}$  are the state transition matrix and the observation matrix, and  $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$  and  $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$  are noise terms. For the task of temporal velocity refinement, the input term  $\mathbf{B}\mathbf{u}_k$  is not needed and is omitted hereafter. To generalize this formulation to nonlinear systems, extended Kalman filter [61, 62] replaces  $\mathbf{A}$  and  $\mathbf{H}$  with nonlinear functions  $\mathbf{f}$  and  $\mathbf{h}$  whose local linearity at  $\mathbf{x}$  is characterized by Jacobian matrices  $\mathbf{J}_\mathbf{f}(\mathbf{x})$  and  $\mathbf{J}_\mathbf{h}(\mathbf{x})$  respectively. Then, we iterate between the *forecast step* and the *assimilation step* to progressively refine  $\mathbf{x}_k$  given previous estimation  $\mathbf{x}_{k-1}$  and current observation  $\mathbf{z}_k$  for each time step  $k$ . More details on EKF are provided in the **supplementary materials**.

Intuitively, EKF imposes a stronger correction of  $\mathbf{x}_k^f = \mathbf{f}(\mathbf{x}_{k-1})$  when the accumulated covariance  $\mathbf{P}_k$  grows larger, thereby smoothing the state sequence and reducing noisy or zigzagging patterns. Therefore, it stands as a promising tool for the refinement of Gaussian velocity. Applying EKF to the FlowGaussian-VR pipeline, we define  $\mathbf{f}$  and  $\mathbf{h}$  as:

$$\begin{aligned} \mathbf{f}(\mathbf{x}_{t-1}) &= D(D^{-1}(\mathbf{x}_{t-1}, t-1), t), \\ \mathbf{h}(\mathbf{x}) &= \text{VRaster}(\mathbf{x}), \end{aligned} \quad (11)$$

where  $D$  is the deformation network and  $\text{VRaster}(\cdot)$  is the velocity rasterization process described Section 3.1. Since both  $D$  and  $\text{VRaster}(\cdot)$  are intractable, we have to assume their local linearity and approximate the partial gradients as:

$$\begin{aligned} \mathbf{J}_\mathbf{f}(\mathbf{x}_{t-1}) &= \frac{\partial \mathbf{f}(\mathbf{x}_{t-1})}{\partial \mathbf{x}} \approx \frac{\mathbf{f}(\mathbf{x}_{t-1} + \delta \mathbf{x}) - \mathbf{f}(\mathbf{x}_{t-1})}{\delta \mathbf{x}}, \\ \mathbf{J}_\mathbf{h}(\mathbf{x}_t) &= \frac{\partial \mathbf{h}(\mathbf{x}_t)}{\partial \mathbf{x}} \approx \frac{\mathbf{h}(\mathbf{x}_t + \delta \mathbf{x}) - \mathbf{h}(\mathbf{x}_t)}{\delta \mathbf{x}}. \end{aligned} \quad (12)$$

Furthermore, since  $D$  is non-invertible and directly obtaining  $\mathbf{f}(\mathbf{x}_{t-1} + \delta\mathbf{x})$  is impossible, we take a detour by considering the Jacobian of  $D$  ( $\mathbf{J}_D$ ):  $\mathbf{J}_f(\mathbf{x}_{t-1}) = \mathbf{J}_D(\mathbf{x}_t) \cdot \mathbf{J}_D(\mathbf{x}_{t-1})^{-1}$  from the inverse function theorem. A more detailed calculation is provided in the **supplementary materials**.

**Surface Gaussian filtering.** In an image rendered from 3D Gaussians, multiple Gaussians could project to the same pixel, yet only those on the surface of objects contribute to the optical flow value of that pixel. Therefore, we filter out each Gaussian whose projected depth is greater than the rendered depth value of its corresponding pixel at each time step, so that occluded Gaussians are not mistakenly updated. Note that we still run the forecast step for those excluded Gaussians, while skipping the assimilation step by directly using  $\mathbf{x}_k^f$  and  $\mathbf{P}_k^f$  as  $\mathbf{x}_k$  and  $\mathbf{P}_k$ .

**Accurate flow map localization.** Another challenge of TVR is to select the correct optical flow observation for each Gaussian. While using the projected coordinates at each time step sounds plausible, this approach links the reliability of  $\mathbf{z}_k$  to the accuracy of  $\mathbf{x}_{k-1}$ . That is, for an erroneous  $\mathbf{x}_{k-1}$ , the selected observation will also deviate from the ground truth flow value, which further causes the assimilated  $\mathbf{x}_k$  to have an even larger error and finally leads to catastrophic error accumulation. To resolve this problem, we assume that Gaussian locations are reliable only at the first frame, and for the  $k^{\text{th}}$  time step,  $\mathbf{z}_k$  is calculated as:  $\mathbf{z}_k = \mathbf{F}(\mathbf{p}_0 + \sum_{i=0}^{k-1} \mathbf{z}_i)$ , where  $\mathbf{F}(\mathbf{p})$  denotes the optical flow value at coordinates  $\mathbf{p}$ , and  $\mathbf{p}_0$  is the projected 2D location of  $\mathbf{x}_0$ . That is, the coordinates used to index the flow map from the second frame on are obtained via adding up the flow values from all previous time steps.

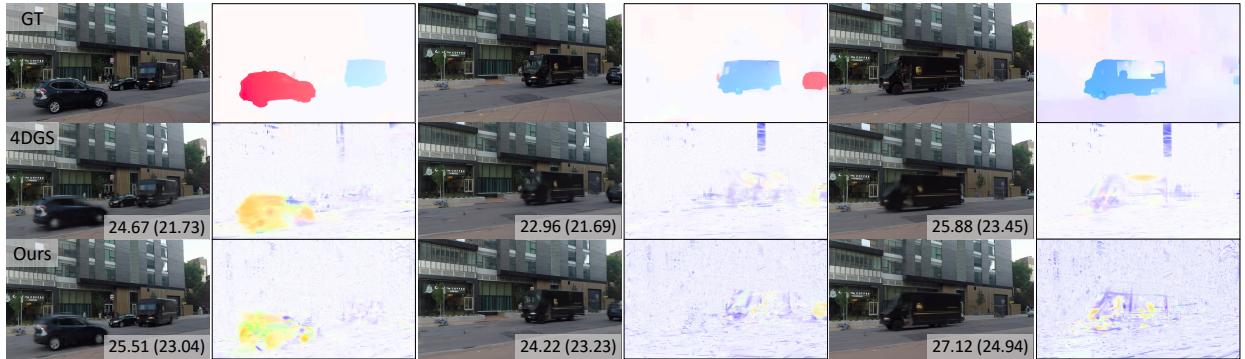


Figure 5: **Novel view synthesis evaluation on the Nvidia “Truck” scene.** We compare our work with the retrained baseline (4DGS) over a few timesteps. As each timestep, we show the rendered novel view image with PSNR (DPSNR) in dB, and the estimated velocity field (color gamma-tuned for visualization).

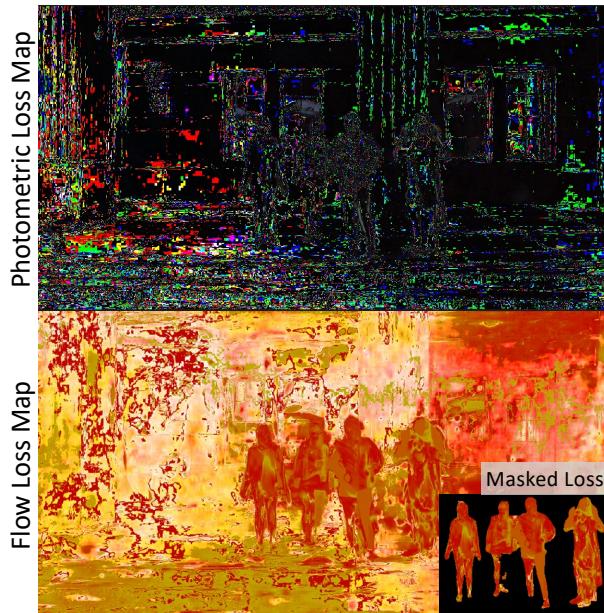


Figure 6: **Comparison of photometric and flow-based losses.** We visualize the pixel-wise photometric loss and our proposed flow-based loss for a given frame. We observe that  $\mathcal{L}_{\text{warp}}$  and  $\mathcal{L}_{\text{win}}$  are primarily concentrated on foreground pixels.

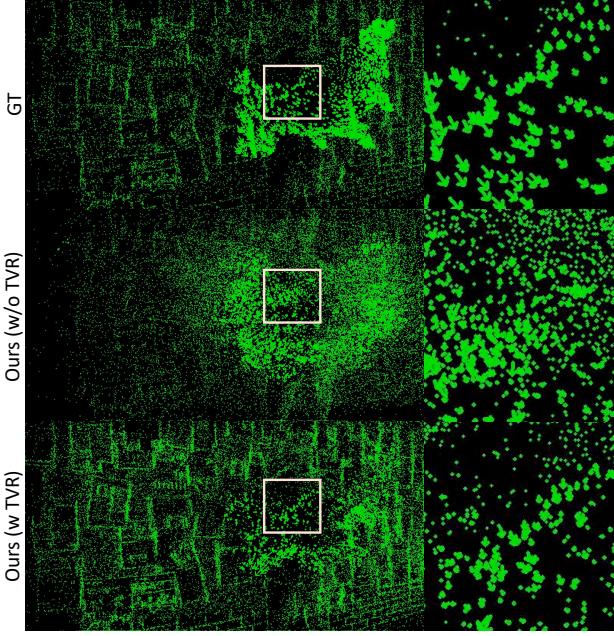


Figure 7: Visualization of the effectiveness of Temporal Velocity Refinement (TVR) trajectory correction. The figure depicts the 3D Gaussian motion rendered as a 2D velocity field, with arrows indicating the velocity vectors between consecutive frames.

## 5 Experiment

### 5.1 Initialization and Implementation Details

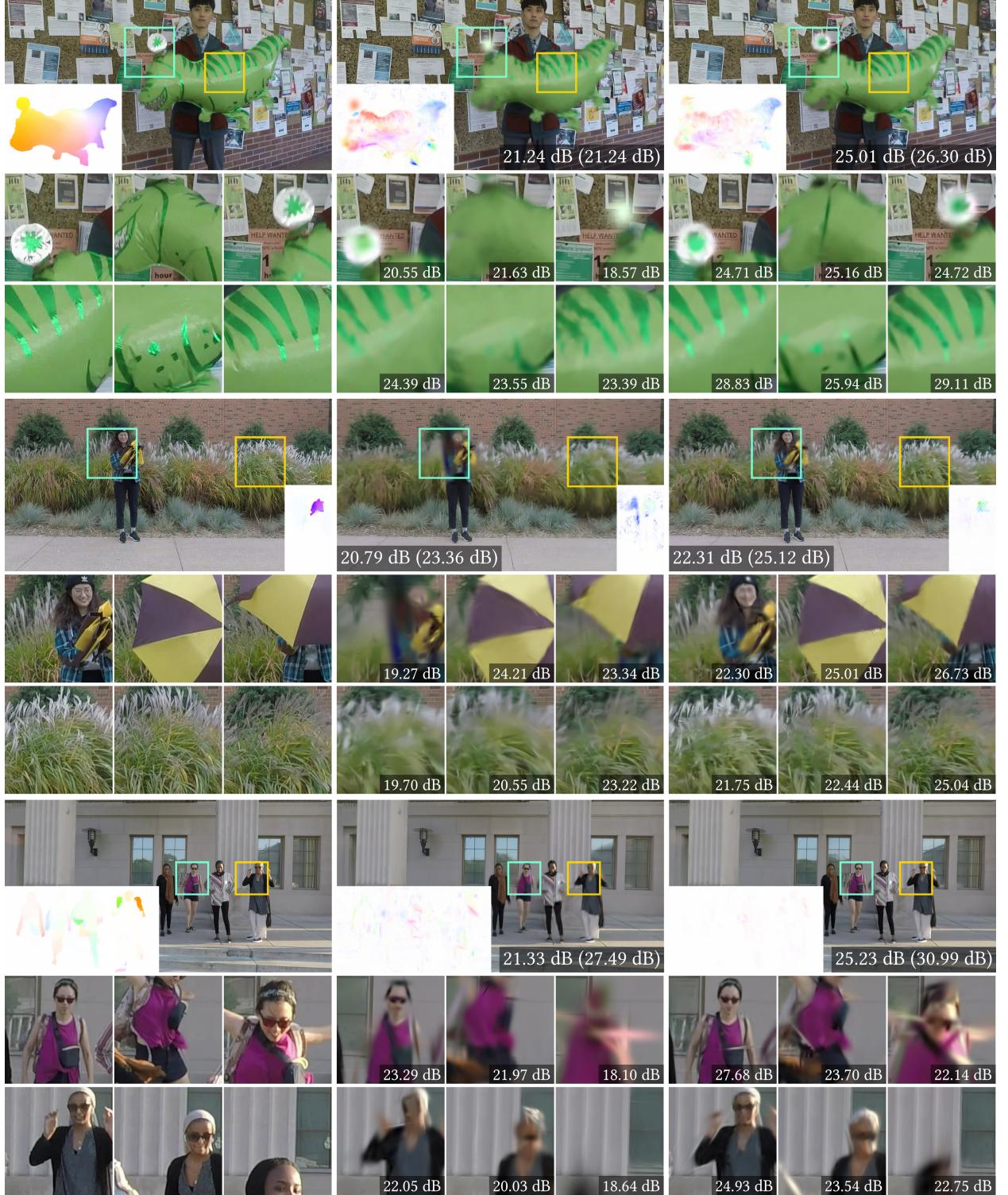
Similar to the initialization strategy in 3DGS [63], reconstruction quality improves when the initial scene point cloud is derived from SfM [64]. Following 4DGS [13], the first frame of each video sequence is used as the initialization frame, and COLMAP [56] is applied to generate Gaussian centers. Ground truth optical flows are obtained using RAFT [19], and dynamic foreground segmentation masks are manually selected with SAM-2 [20].

We conducted quantitative evaluations on the Nvidia-long [22] dataset, a benchmark for dynamic scenes, pre-processed and made publicly available by prior works [43]. This dataset includes 7 scene videos, each containing 90–210 frames with 12 camera views per frame. The Neu3D dataset [23], captured with 15–20 static cameras over extended periods with complex motions, features 6 selected scene sequences of 300 frames each. These datasets comprise high-resolution videos of dynamic scenes, offering challenges such as occlusions, rapid motion, and large-scale changes.

Our FlowGaussian-VR pipeline builds upon 4DGS [14], and compares with 4DGS, 4D-GS [24], SC-GS [25], and MotionGS [50]. All training is performed on an NVIDIA RTX 4090 GPU. More experimental details can be found in the **supplementary material**. Notably, we set the FPS point cloud downsampling ratio in FAD (Sec 3.4) to 0.01 for Nvidia-long and 0.001 for Neu3D.

### 5.2 Results

The quantitative results on the Nvidia-long [22] and Neu3D [23] datasets are summarized in Table 1 and Table 2, respectively. A qualitative comparison between FlowGaussian-VR and the 4DGS baseline can be found in Fig. 8 and Fig. 9. In addition to standard metrics such as PSNR, SSIM, and LPIPS, we report dynamic PSNR (DPSNR) to specifically evaluate the rendering quality of dynamic foreground pixels. D-PSNR computes PSNR only in dynamic regions, identified by SAM-2, to assess reconstruction quality in motion areas. Our experimental results demonstrate that FlowGaussian-VR significantly outperforms all three baselines in novel view synthesis. The average PSNR metrics across all scenes in the Neu3D and Nvidia-long datasets are improved by 2.45 dB and 2.5 dB, respectively, compared to the best-performing 4DGS method. For dynamic scene rendering, our method achieves average improvements of 3.14 dB and 2.4 dB in DPSNR over 4DGS. These metrics validate that our method effectively optimizes both static and dynamic components of the scene, and the gain in DPSNR is more pronounced compared to the overall PSNR, highlighting the effectiveness of FlowGaussian-VR on capturing and reconstructing dynamic scene contents.



**Figure 8: Novel view synthesis evaluation on the Nvidia dataset.** We present three scenes: “Balloon1”, “Umbrella”, and “Jumping” and compare our work with retrained 4DGS. The 3 primary columns show results of the ground truth, baseline, and our method. For each scene, ground truth optical flows (first column) and rendered velocity fields (other columns) are presented in false color. The second and third rows within each scene show insets of 2 subareas, from left to right yielding 3 different timestamps. The PSNR showing at corners is calculated using the entire image.

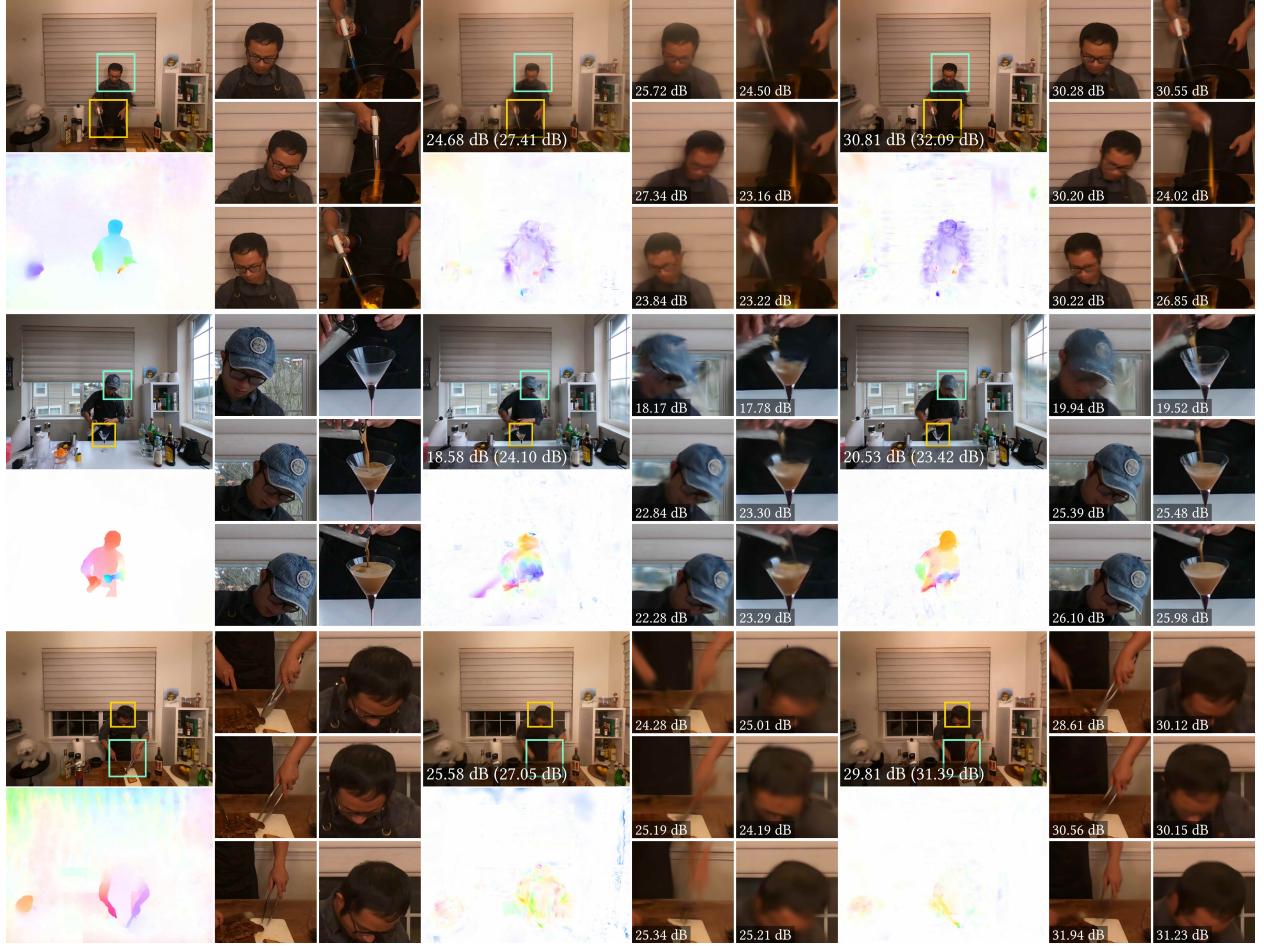


Figure 9: **Novel view synthesis evaluation on the Neu3d dataset.** We visualize the “flame”, “martini”, and “beef” scenes and compare our approach with the retrained 4DGS. We rearranged the inset placement from figure 8 to suit the aspect ratio of this dataset, that the 3 main columns from left to right are still the results of the ground truth, 4DGS, and our method, and the insets from top to bottoms within each group represents 3 different time stamps. The colored image on the bottom-left of each group is optical flow for the ground truth, and velocity field rendering for the others accordingly.

Table 3: **Effectiveness assessment of Velocity Rendering (V.R.), FAD,  $\mathcal{L}_{\text{warp}}$ , and  $\mathcal{L}_{\text{win}}$  on dynamic scene reconstruction.** “N” refers to the number of Gaussians in the final point cloud.

|      | V.R. | FAD | $\mathcal{L}_{\text{warp}}$ | $\mathcal{L}_{\text{win}}$ | PSNR / SSIM / LPIPS   | N    |
|------|------|-----|-----------------------------|----------------------------|-----------------------|------|
| 4DGS | ✗    | ✗   | ✗                           | ✗                          | 20.51 / 0.619 / 0.317 | 214k |
|      | ✓    | ✓   | ✓                           | ✗                          | 22.69 / 0.703 / 0.312 | 89k  |
|      | ✓    | ✓   | ✗                           | ✓                          | 22.47 / 0.718 / 0.305 | 75k  |
|      | ✓    | ✓   | ✓                           | ✓                          | 24.50 / 0.757 / 0.290 | 141k |

We further provide a close-up inspection of velocity fields for synthesized novel views on the “Truck” scene to demonstrate the effectiveness of our method in Fig. 5. In this scene, the primary reconstruction challenges lie in the significant scale variations between the truck and the car, motion blur of the car, and fine-grained details such as text on the truck. As shown in the rendered Gaussian velocity field, our method produces gradually enlarging Gaussians on the truck’s front as it approaches, aligning well with the physical motion. In contrast, 4DGS introduces inconsistent Gaussian replacements that violate geometric continuity. Additionally, the RGB rendering results further confirm that Gaussians adhering to plausible motion dynamics yield more realistic visual quality.

Table 4: **Effectiveness assessment of sliding window size for  $\mathcal{L}_{\text{win}}$**  on final rendering quality. Metric reported is PSNR (dB)↑.

| Number of adjacent frames | 2 (4DGS) | 4     | 6     | 8     |
|---------------------------|----------|-------|-------|-------|
| Jumping                   | 25.33    | 25.63 | 26.45 | 27.89 |
| Coffee-martini            | 22.47    | 22.68 | 23.23 | 24.19 |

### 5.3 Ablation Study

For ablation studies, we chose one representative scene from each dataset based on the level of difficulty: the relative easy “martini” from Neu3D and the hard “jumping” from Nvidia-long. We validate the effectiveness of velocity rendering,  $\mathcal{L}_{\text{warp}}$ , and  $\mathcal{L}_{\text{win}}$  in reconstructing dynamic scenes and report the results in Table 3.

**Ablation on Sliding Window Size.** We study the impact of the sliding window size in  $\mathcal{L}_{\text{win}}$  on the reconstruction performance, and present the results in Table 4. We observe that a larger window size results in stronger temporal consistency and thus incurs more accurate velocity estimation of moving objects. Such benefits are attributed to the temporal window overlapping between adjacent iterations, bringing stability to the training process.

**Loss Map Visualization.** We visualize the loss maps of the  $\mathcal{L}_1$  photometric loss and flow-based losses for a rendered frame in Fig. 6. We note that  $\mathcal{L}_1$  does not emphasize dynamic regions and therefore misleads conventional densification algorithms to prioritize static backgrounds, potentially resulting in inaccurate reconstructions of dynamic elements. On the other hand, our  $\mathcal{L}_{\text{win}}$  and  $\mathcal{L}_{\text{warp}}$  generally yield larger loss values for dynamic pixels, making them more suitable for guiding the densification steps in dynamic scenes. By additionally applying a dynamic mask to filter out irrelevant losses from the background, we further ensure that all Gaussian candidates in FAD correspond to foreground objects.

**Evaluation of TVR.** By applying the temporal velocity refinement (TVR) with EKF to the “Balloon1” scene from Nvidia-long, we observe that the PSNR on novel views decreases from 24.50 to 24.32 (-0.18 change) and the DPSNR decreases from 24.54 to 24.17 (-0.37 change). While TVR induces a slight deterioration in render quality, we yet observe that the accuracy of Gaussians’ offsets (motion), and subsequently the modeling of scene dynamics, has notably improved, as shown in Fig. 7. We argue that the drop in visual quality is due to TVR not being able to optimize Gaussian attributes other than the center coordinates.

## 6 Conclusion and Analysis

In this work, we present an enhanced flow-based velocity field modeling technique to improve Gaussian deformation for video reconstruction. By expanding the capabilities of the 3DGS rasterizer to accommodate velocity field rendering, we introduce a velocity attribute to each Gaussian and incorporate multiple constraints —  $\mathcal{L}_{\text{win}}$ ,  $\mathcal{L}_{\text{warp}}$ , and  $\mathcal{L}_{\text{dyn}}$  — to ensure temporal consistency and smoothness in both RGB rendering and trajectory predictions. Additionally, we develop FAD, a method that actively adds Gaussians in challenging dynamic regions. Extensive experiments conducted on challenging datasets illustrate that our model demonstrates improved alignment with real motion and reduced motion blur, leading to clearer textures for smoother motions. We believe our approach holds significant potential for achieving more accurate and temporally consistent dynamic scene reconstruction in applications such as VR/AR, gaming, and cinematic content creation.

**Limitations and future work.** Our exploration of temporal velocity refinement using EKF is still in its infant stages, yet it has shown promise in refining Gaussian center trajectories alongside other attributes, and is worth further research and experimentation. The current model encounters challenges in accurately predicting motion trajectories under complex and rapid motion, particularly in scenarios involving abrupt object changes (e.g., abrupt appearance or size variations), for example, the “Jumping” scene in Nvidia-long. As shown in Fig. 8, the visual quality and PSNR of reconstruction decrease for later timestamps, when persons in the scene move closer and existing Gaussians are no longer sufficient to fully represent their fine details. Future improvements in dynamic Gaussian control and adaptive optimization strategies could possibly address these issues. Additionally, maintaining Gaussian attribute consistency across views during significant camera perspective changes (e.g., 360-degree rotations [65]) remains a challenge. Lastly, the sliding window strategy for loss computation leads to increased training time. Addressing these aspects presents opportunities for future improvements in velocity-based modeling.

## References

- [1] Navid Ashrafi, Francesco Vona, Carina Ringsdorf, Christian Hertel, Luca Toni, Sarina Kailer, Alice Bartels, Tanja Kojic, and Jan-Niklas Voigt-Antons. Enhancing job interview preparation through immersive experiences using

- photorealistic, ai-powered metahuman avatars. In *2024 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 345–346. IEEE, 2024.
- [2] Shawheen Alipour, Audrey Rah, Bishnu Karki, Justin Burris, Leslie Coward, and Tony Liao. Enhancing physics education: Designing customized virtual reality for teaching crystalline structures. In *2024 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 549–552. IEEE, 2024.
  - [3] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024.
  - [4] Weiliang Chen, Fangfu Liu, Diankun Wu, Haowen Sun, Haixu Song, and Yueqi Duan. Dreamcinema: Cinematic transfer with free camera and 3d character. *arXiv preprint arXiv:2408.12601*, 2024.
  - [5] Xiaoyu Zhou, Zhiwei Lin, Xiaojun Shan, Yongtao Wang, Deqing Sun, and Ming-Hsuan Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21634–21643, 2024.
  - [6] Xiaokun Pan, Gan Huang, Ziyang Zhang, Jinyu Li, Hujun Bao, and Guofeng Zhang. Robust collaborative visual-inertial slam for mobile augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
  - [7] Zhiming Hu, Zheming Yin, Daniel Haeufle, Syn Schmitt, and Andreas Bulling. Hoimotion: Forecasting human motion during human-object interactions using egocentric 3d object bounding boxes. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
  - [8] Mine Dastan, Michele Fiorentino, Elias D Walter, Christian Diegritz, Antonio E Uva, Ulrich Eck, and Nassir Navab. Co-designing dynamic mixed reality drill positioning widgets: A collaborative approach with dentists in a realistic setup. *IEEE Transactions on Visualization and Computer Graphics*, 2024.
  - [9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.
  - [10] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
  - [11] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.
  - [12] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024.
  - [13] 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*, pages 20310–20320, 2024.
  - [14] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024.
  - [15] Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wencho Ma, Le Chen, Danhang Tang, and Ulrich Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365*, 2024.
  - [16] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024.
  - [17] Yanjun Huang, Jiatong Du, Ziru Yang, Zewei Zhou, Lin Zhang, and Hong Chen. A survey on trajectory-prediction methods for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(3):652–674, 2022.
  - [18] Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R Oswald, Andreas Geiger, and Marc Pollefeys. Nicer-slam: Neural implicit scene encoding for rgb slam. *arXiv preprint arXiv:2302.03594*, 2023.
  - [19] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.
  - [20] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

- [21] Keisuke Fujii. Extended kalman filter. *Refernce Manual*, 14:41, 2013.
- [22] 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*, pages 5336–5345, 2020.
- [23] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022.
- [24] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024.
- [25] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024.
- [26] Chaoyang Wang, Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural prior for trajectory estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6532–6542, 2022.
- [27] 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*, pages 7002–7012, 2020.
- [28] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019.
- [29] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (ToG)*, 35(4):1–13, 2016.
- [30] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1611–1621, 2021.
- [31] Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. Consistent depth of moving objects in video. *ACM Transactions on Graphics (ToG)*, 40(4):1–12, 2021.
- [32] Seoha Kim, Jeongmin Bae, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Sync-nerf: Generalizing dynamic nerfs to unsynchronized videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2777–2785, 2024.
- [33] Guangcong, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [34] Fabio Tosi, Alessio Tonioni, Daniele De Gregorio, and Matteo Poggi. Nerf-supervised deep stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 855–866, June 2023.
- [35] 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)*, pages 14304–14314. IEEE Computer Society, 2021.
- [36] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021.
- [37] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9421–9431, 2021.
- [38] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.
- [39] 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*, pages 5865–5874, 2021.
- [40] 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*, 2021.

- [41] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.
- [42] 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*, pages 6498–6508, 2021.
- [43] 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*, pages 4273–4284, 2023.
- [44] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.
- [45] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19447–19456, 2024.
- [46] 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*, pages 20331–20341, 2024.
- [47] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.
- [48] Alvaro Cuno, Claudio Esperança, Antonio Oliveira, and Paulo Roma Cavalcanti. 3d as-rigid-as-possible deformations using mls. In *Proceedings of the 27th computer graphics international conference*, pages 115–122. Citeseer, 2007.
- [49] Zhiyang Guo, Wengang Zhou, Li Li, Min Wang, and Houqiang Li. Motion-aware 3d gaussian splatting for efficient dynamic scene reconstruction. *arXiv preprint arXiv:2403.11447*, 2024.
- [50] Ruijie Zhu, Yanzhe Liang, Hanzhi Chang, Jiacheng Deng, Jiahao Lu, Wenfei Yang, Tianzhu Zhang, and Yongdong Zhang. Motions: Exploring explicit motion guidance for deformable 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37:101790–101817, 2024.
- [51] Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics (TOG)*, 43(6):1–18, 2024.
- [52] Shizun Wang, Xingyi Yang, Qihong Shen, Zhenxiang Jiang, and Xinchao Wang. Gflow: Recovering 4d world from monocular video. *arXiv preprint arXiv:2405.18426*, 2024.
- [53] Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lv, Peng Wang, Wenping Wang, and Junhui Hou. Modgs: Dynamic gaussian splatting from causally-captured monocular videos. *arXiv preprint arXiv:2406.00434*, 2024.
- [54] Xiao Fu, Wei Yin, Mu Hu, Kaixuan Wang, Yuexin Ma, Ping Tan, Shaojie Shen, Dahua Lin, and Xiaoxiao Long. Geowizard: Unleashing the diffusion priors for 3d geometry estimation from a single image. In *European Conference on Computer Vision*, pages 241–258. Springer, 2025.
- [55] Fabio Tosi, Youmin Zhang, Ziren Gong, Erik Sandström, Stefano Mattoccia, Martin R Oswald, and Matteo Poggi. How nerfs and 3d gaussian splatting are reshaping slam: a survey. *arXiv preprint arXiv:2402.13255*, 4, 2024.
- [56] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [57] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. The farthest point strategy for progressive image sampling. *IEEE transactions on image processing*, 6(9):1305–1315, 1997.
- [58] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588, 2020.
- [59] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.
- [60] Yifan Zhan, Zhuoxiao Li, Muyao Niu, Zhihang Zhong, Shohei Nobuhara, Ko Nishino, and Yinqiang Zheng. Kfd-nerf: Rethinking dynamic nerf with kalman filter. *arXiv preprint arXiv:2407.13185*, 2024.
- [61] Gabriel A Terejanu et al. Extended kalman filter tutorial. *University at Buffalo*, 27, 2008.
- [62] Chang Ho Kang, Chan Gook Park, and Jin Woo Song. An adaptive complementary kalman filter using fuzzy logic for a hybrid head tracker system. *IEEE Transactions on Instrumentation and Measurement*, 65(9):2163–2173, 2016.

- [63] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [64] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III* 14, pages 501–518. Springer, 2016.
- [65] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.