REON-NVS: Real-Time Online Novel-View Synthesis from Sparse-View Videos

Daeyeon Kim, Jinhyeok Kim, Gangmin Kwon,
Seungjoo Shin, and Sunghyun Cho

POSTECH
{dkim011006, jh4011, gmkwon99, seungjoo.shin, s.cho}@postech.ac.kr

Fig. 1: We present REON-NVS, a feedforward online NVS framework that performs pose estimation and novel-view reconstruction from sparse-view video streams in real-time. The right figure is evaluated on the Neural 3D Video [20] dataset.

Abstract. Recent advances in online reconstruction of dynamic scenes demonstrate impressive visual quality, showing potential for real-world applications such as VR content streaming. Yet, existing online reconstruction methods still require a large number of input views and time-consuming iterative optimization. Moreover, reliable pose estimation, a prerequisite for NVS, introduces additional delay. In this paper, we present REON-NVS, a feedforward online NVS framework for sparse-view input video streams, which goes from pose estimation to novel-view reconstruction in real time. REON-NVS comprises two main components. First, our method leverages a feedforward pose estimator and mapper that replace conventional camera pose optimization pipelines. Second, we design a scene reconstructor built upon a state-space model (SSM), which efficiently synthesizes temporally consistent novel-view images by exploiting information from previous frames. To train and evaluate our approach in realistic in-the-wild streaming scenarios, we introduce a new multi-view dynamic scene dataset of 150 dynamic scenes captured with moving cameras. Extensive experiments demonstrate that REON-NVS achieves high visual quality while operating in real time (32 FPS), validating its applicability in real-world scenarios.

1 Introduction

Reconstructing dynamic scenes from sparse-view videos plays a crucial role in applications such as free-viewpoint video (FVV), augmented reality (AR), and virtual reality (VR). By enabling users to freely navigate and observe photo-realistic scenes from arbitrary viewpoints, dynamic scene reconstruction opens new possibilities for next-generation interactive media that go beyond traditional static content.

Recently, Neural Radiance Fields (NeRFs) [26] and 3D Gaussian Splatting (3DGS) [15], which reconstruct static scenes through iterative optimization, have revolutionized the field of novel-view synthesis (NVS). Following this breakthrough, subsequent studies [22, 28, 29, 35, 39] have extended to dynamic scene reconstruction. While promising, many of them require offline optimization of the entire video sequence before inference, making them unsuitable for real-time streaming scenarios where video frames arrive continuously, such as live broadcasting or virtual meetings.

In response, several approaches [10, 19, 34, 38] explored online reconstruction of dynamic scenes from streaming inputs, enabling incremental scene updates in more realistic, real-world settings. However, these methods still face two practical challenges. First, they often rely on a large number of input viewpoints for stable reconstruction, limiting their applicability in sparse-view streaming scenarios. Second, as illustrated in Fig. 1, they process incoming frames much slower than the typical 30 FPS frame rate, leading to accumulated delay over time. Furthermore, they often rely on external camera pose estimation pipelines such as COLMAP [32], which introduce additional delay. Such reliance on numerous viewpoints and accumulated delay hinder the real-time applicability of online novel-view synthesis, particularly in scenarios that demand instantaneous interaction.

To address these challenges, we propose **REON-NVS**, a real-time online NVS framework for dynamic scenes from sparse-view input streams. Specifically, REON-NVS achieves 32 FPS on a single A100 GPU with two input views, surpassing the standard 30 FPS frame rate. To enable such real-time and delay-free processing, our framework consists of two major components: (1) a pose estimator and a mapper that infer camera poses directly from input frames without iterative optimization, and (2) a scene reconstructor that efficiently synthesizes temporally consistent novel views from sparse-view inputs. These two components are jointly designed to operate in a fully online manner, processing streaming video frames sequentially while preserving spatial fidelity and temporal coherence.

The pose estimator and mapper replace conventional time-consuming pose pipelines (e.g., COLMAP [32]) by estimating poses in a single feedforward pass. Specifically, inspired by RayZer [12], REON-NVS employs self-supervised learning and adopts a latent pose representation strategy, where camera poses are predicted within a latent space that emerges during training. This lightweight design enables fast pose estimation while maintaining reliable performance under real-time streaming conditions. Meanwhile, the scene reconstructor employs

a state-space model (SSM) to efficiently aggregate information from past frames, propagating temporal context to ensure consistent reconstruction. By directly synthesizing novel-view frames without relying on explicit geometric reconstruction, REON-NVS achieves both computational efficiency and photorealistic rendering quality.

In addition, we introduce a new multi-view dynamic scene dataset designed for both training and evaluation of multi-view NVS. The dataset contains 150 diverse real-world scenes with synchronized moving cameras and dynamic content, serving as a challenging benchmark for learning and assessing streaming-based reconstruction. Extensive experiments demonstrate that REON-NVS achieves superior visual quality under sparse-view inputs, outperforming existing streamable methods by +3.3 dB and feedforward baselines by up to +1.7 dB PSNR on our benchmarks, while operating in real time (32 FPS).

We summarize our contributions as follows:

- We introduce REON-NVS, the first framework to tackle online novel-view synthesis of dynamic scenes from unposed multi-camera streams. It performs pose estimation, reconstruction, and rendering in a single forward pass.
- REON-NVS replaces conventional time-consuming pose estimation by predicting the camera pose of each input video frame in a feedforward manner.
- To achieve high-quality, temporally consistent NVS in real time, we propose a novel scene reconstructor, built upon an SSM architecture.
- We release a novel multi-view dynamic scene dataset of 150 dynamic real-world scenes captured with moving cameras, enabling research under realistic in-the-wild streaming scenarios.

2 Related Work

NVS in Static Scenes. NVS has advanced rapidly with the development of diverse scene representations, including NeRF [26], Instant-NGP [27], and 3DGS [15]. These methods reconstruct static scenes via per-scene optimization, yielding high-fidelity novel views once the scene representation has been trained. While achieving impressive quality, their reliance on iterative optimization makes them unsuitable for real-world scenarios requiring immediate reconstruction and rendering.

To avoid scene-specific optimization, recent work explores feedforward architectures that directly predict 3D representations for static scenes [3, 5, 36, 40, 44]. While feedforward approaches generalize across scenes and offer fast inference without iterative optimization, Jin *et al.* [13] observe that the use of explicit 3D representations and rendering equations can introduce constraints on representational flexibility, which may limit novel-view synthesis quality. In response, recent works avoid explicit 3D structures, directly synthesizing novel-view images in a geometry-free manner [13, 31]. These geometry-free methods circumvent errors introduced by explicit geometry, resulting in improved synthesis quality. Taking it a step further, RayZer [12] removes the need for ground-truth pose supervision by implicitly modeling viewpoint information through self-supervised learning,

although this implicit representation does not offer explicit controllability. Motivated by these works, we extend geometry-free, self-supervised approaches to online reconstruction of dynamic scenes.

NVS in Dynamic Scenes. Extending static scene reconstruction to dynamic scenes has been an active area of research. Monocular approaches [22, 28, 29, 39] model scene deformations over time, enabling novel-view synthesis from single-camera videos but often struggle with occlusions and limited viewpoint diversity. Multi-camera methods [17, 21, 35] leverage synchronized inputs to capture richer geometry and motion, yet still rely on offline training for the entire video sequence. More recently, BTimer [23] and MoVieS [24] introduced feed-forward NVS frameworks for dynamic scenes captured with a monocular camera. Both methods leverage temporal attention within each temporal window, where BTimer predicts independent 3D Gaussians at each bullet timestamp and MoVieS explicitly regresses per-pixel 3D motion displacements. However, their Transformer-based temporal modeling scales poorly with sequence length, restricting them to fixed-size windows. As a result, while intra-window temporal coherence is maintained, no mechanism exists to enforce consistency across windows, limiting their practicality for continuous and interactive video sequences.

Some prior works address these issues by enabling continuous video processing, synthesizing novel views in an online manner as new frames arrive [10, 19, 34, 38]. These methods incrementally update 3D representations, removing the need to re-optimize the entire scene and enabling online updates. Although more efficient, they still rely on iterative optimization, which prevents them from achieving true real-time performance and limits their applicability in interactive streaming scenarios. Another line of work adopts restrictive setups, such as depth sensors (e.g., RGB-D) [9, 11]. While these constraints can improve reconstruction speed and accuracy, they reduce flexibility and hinder deployment in uncontrolled environments. In contrast, our method directly takes multi-view video streams and produces novel-view frames in a feedforward manner, without per-frame camera pose optimization in the streaming loop or restrictive setups, while running in real time (>30 FPS) in sparse-view settings.

3 REON-NVS

Fig. 2 shows an overview of REON-NVS. At each time step t, REON-NVS receives multi-view input frames $\{I_t^k\}$ and a target camera pose p_t^T. Here, $k = 1, 2, \ldots, n$ indexes the input views, and the superscript T denotes the target view. Given the input, REON-NVS synthesizes a novel-view image I_t^T corresponding to p_t^T by leveraging both the information from multi-view observations at time t and temporal information propagated from previous time steps. To achieve temporally consistent NVS in real time from unposed input frames, REON-NVS comprises two main stages: pose estimation and scene reconstruction. In the following, we detail each stage of our framework, along with the dataset and training strategy.

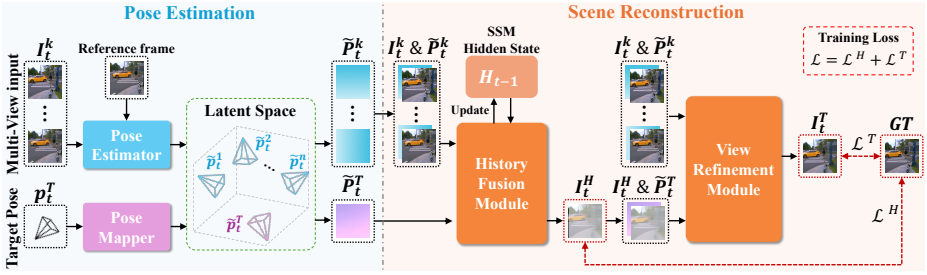

Fig. 2: REON-NVS framework. At each time step t, REON-NVS receives sparse-view video frames $\{I_t^k\}$ along with a user-specified target viewpoint p_t^T. (**Left**) REON-NVS first estimates the latent poses of the input frames $\{\tilde{p}_t^k\}$ and the target pose $\tilde{p}_t^T$ relative to the reference frame I_r in the latent space. (**Right**) REON-NVS synthesizes the target view frame I_t^T by leveraging the latent poses $\{\tilde{p}_t^k\}$ and $\tilde{p}_t^T$. The history fusion module exploits the SSM hidden state as temporal memory, incorporating information from previous frames to generate a coarse prediction I_t^H, which is then refined by the view refinement module into the final output I_t^T.

3.1 Pose Estimation and Mapping

Following prior approaches [12,16], our framework predicts relative camera poses. For this, we designate the first-frame image I_0^1 as the canonical reference image I_r, whose camera pose is defined as an identity rotation with zero translation. Subsequent poses are then inferred relative to this canonical reference, forming a consistent coordinate system throughout the sequence. Building upon this, we introduce two neural modules to enable real-time pose processing: a pose estimator that infers latent camera poses from input frames, and a pose mapper that bridges them with physical camera poses.

Pose Estimator. To avoid per-scene pose optimization, we adopt a lightweight feedforward pose estimator. Given an input frame I_t^k and a reference frame I_r, the pose predictor $\mathcal{P}$ predicts the camera pose of I_t^k relative to I_r:

$$\tilde{p}_t^k = \mathcal{P}\left(I_t^k, I_r\right). \tag{1}$$

The predicted pose $\tilde{p}_t^k$ is a 9-dimensional pose vector, comprising 3D translation and 6D rotation [42], represented in latent space.

As discussed in RayZer [12] and previous works [1, 4, 18, 42], supervising a feedforward estimator with ground-truth poses is unstable due to the low-dimensional and highly constrained nature of $SE(3)$ poses. Furthermore, COLMAP failure cases can lead to noisy pose annotations, which degrade pose-supervised pipelines. Therefore, REON-NVS adopts the self-supervised learning strategy of RayZer [12]. Instead of training with GT camera poses, the pose estimator is trained using RGB reconstruction loss between the ground-truth target frame and the synthesized image from the latent pose $\tilde{p}_t^k$ (see Sec. 3.2). This self-supervision strategy allows the network to learn a latent pose space that naturally emerges during training, resulting in more accurate camera pose estimation even with a compact model size [12]. Note that the latent camera pose $\tilde{p}_t^k$

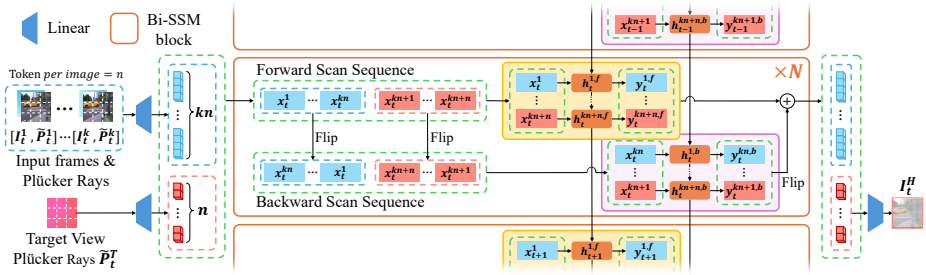

Fig. 3: Overall structure of the History Fusion Module. The History Fusion Module consists of N stacked Bi-SSM blocks. For the backward scan, the input view tokens and target view tokens are flipped in place, ensuring that the target token is updated only after all other tokens have been processed.

is not physically interpretable, but rather a latent representation that emerges from the self-supervised reconstruction process.

Pose Mapper. The pose estimator assigns each input frame a latent pose that is not physically interpretable. In contrast, the target camera pose p_t^T is specified in the physical pose space, leading to a mismatch between the two representations. This mismatch prevents direct viewpoint control using physical poses [12]. To enable controllable novel-view synthesis, it is therefore necessary to bridge the physical and latent pose space.

Accordingly, we introduce a pose mapper that translates a physical pose p into its corresponding latent pose $\tilde{p}$. In our framework, target camera poses are defined as physical poses in the 3D space of the reference frame I_r. However, specifying a camera pose in 3D space inherently suffers from scale ambiguity: the same image projection can arise from either a large-scale scene viewed from a distant camera or a small-scale scene viewed from a nearby one. This ambiguity affects both physical and latent pose spaces, potentially leading to scale mismatches between them.

To resolve this issue, we provide the pose mapper with a pose pair $(\tilde{p}', p')$ serving as a reference example between the two spaces. The pose pair $(\tilde{p}', p')$ represents the camera pose for the initial frame I_0^2 relative to I_r, expressed in the latent and physical spaces, respectively. Formally, the pose mapper predicts a latent pose $\tilde{p}$ from a given physical pose p as:

$$\tilde{p} = \mathcal{M}\Big(p, (\tilde{p}', p')\Big), \tag{2}$$

where $\mathcal{M}$ denotes the mapper implemented as a multi-layer perceptron (MLP). $\tilde{p}'$ is obtained from the pose estimator $\mathcal{P}$, i.e., $\tilde{p}' = \mathcal{P}(I_0^2, I_r)$, and p' is computed using a physical camera pose estimation method (e.g., DA3 [25]). Note that the physical poses p and p' share the same scale, ensuring consistent mapping between the two spaces.

3.2 Scene Reconstruction

As shown in Fig. 2, REON-NVS synthesizes the target-view frame I_t^T from the input frames $\{I_t^k\}$, their latent poses $\{\tilde{p}_t^k\}$, and the target latent pose $\tilde{p}_t^T$. Synthesizing faithful novel-view videos requires exploiting not only the current time-step frames but also preceding ones, as past observations are essential for maintaining temporal consistency. Meanwhile, real-time online NVS demands a computationally efficient architecture capable of processing multiple images under a limited budget.

To address these requirements, we adopt an SSM-based architecture that balances temporal modeling and efficiency. Within this framework, we design two key components: the history fusion module and the view refinement module. The history fusion module integrates temporal information from past frames to generate an initial target-view frame, which is subsequently refined by the view refinement module to produce the final high-quality output. In the following, we describe the history fusion and view refinement modules in detail.

History Fusion. Fig. 3 illustrates the overall structure of the history fusion module. At each time step t, the history fusion module takes the multi-view input frames $\{I_t^k\}$, their associated latent poses $\{\tilde{p}_t^k\}$, and the target latent pose $\tilde{p}_t^T$ as input, and produces a target-view image I_t^H. To exploit past observations, the history fusion module leverages information from previous frames in the form of hidden states of the SSM layers and outputs updated hidden states. We denote the sets of hidden states updated at time step t as H_t.

Formally, the operation of the history fusion module is expressed as:

$$I_t^H, H_t = \mathcal{H}\left(\{[I_t^k, \tilde{P}_t^k]\}, \tilde{P}_t^T, H_{t-1}\right), \tag{3}$$

where $\mathcal{H}$ indicates the history fusion module. $\tilde{P}_t^k$ and $\tilde{P}_t^T$ are pixel-wise Plücker [30] ray embeddings derived from latent poses $\tilde{p}_t^k$ and $\tilde{p}_t^T$, respectively, and $[\cdot,\cdot]$ indicates channel-wise concatenation. Following prior work [12, 13], we compute the 6-dimensional pixel-wise Plücker ray embeddings from the predicted latent poses.

As shown in Fig. 3, the module consists of a stack of N Bi-SSM blocks. To process k multi-view input images, the module first tokenizes the concatenations of input frames and their Plücker ray embeddings $[I_t^k, \tilde{P}_t^k]$ by splitting each of them into non-overlapping patches [8] and projecting each patch with a linear layer. This yields kn tokens, where n is the number of tokens per image. In addition, the target-view Plücker ray embedding $\tilde{P}_t^T$ is tokenized into n patches and linearly projected in the same manner. The resulting $kn + n$ tokens are then processed through the Bi-SSM blocks. Finally, the last n tokens from the output of the N-th Bi-SSM block are mapped to RGB values via a linear layer, producing the synthesized image I_t^H.

The Bi-SSM block builds upon Mamba2 [7], and adopts a bidirectional scanning strategy inspired by Vision Mamba [43] to efficiently model context in unordered 2D inputs. Given an input token sequence, as illustrated in Fig. 3,

we construct a backward sequence by flipping the input image and target view tokens in place, ensuring that the target token is processed only after all input image tokens have been processed. These two sequences, forward and backward, are independently processed through parallel SSM branches.

To leverage temporal information, we propagate SSM hidden states across time steps. Specifically, for the forward scan, the initial hidden state $h_t^{1,f}$ at time step t is obtained from the final hidden state from the previous time step, $h_{t-1}^{kn+n,f}$. Each Bi-SSM block maintains two hidden states, one for the forward branch and the other for the backward branch, resulting in $2N$ hidden states across the history fusion module. These collectively form the set H_t in Eq. (3).

Thanks to this hidden state mechanism, SSMs can effectively model long-range dependencies with substantially lower computational cost than Transformers, which require full-sequence attention. By incorporating bidirectional processing, the Bi-SSM block further improves context modeling for 2D image inputs. For additional architectural details, refer to the supplementary material.

View Refinement. While the history fusion module aggregates temporal information to ensure consistency across frames, this accumulation process can blur fine details and degrade edge sharpness, particularly under excessive compression of information in the SSM hidden states. To alleviate these issues, we introduce a view refinement module that eschews temporal cues and instead refines the reconstruction using only the current multi-view inputs.

Formally, the operation of the view refinement module is defined as:

$$I_t^T = \mathcal{R}\left(\{[I_t^k, \tilde{P}_t^k]\}, [I_t^H, \tilde{P}_t^T]\right), \tag{4}$$

where $\mathcal{R}$ indicates the view refinement module. To balance parameter efficiency, practical applicability, and reconstruction quality, the view refinement module adopts a hybrid architecture combining Bi-SSM blocks and Transformer blocks.

Similar to the history fusion module, the view refinement module begins by tokenizing the input images $\{I_t^k\}$ and the coarse reconstruction I_t^H, along with their associated Plücker ray embeddings. Each image-embedding pair is split into non-overlapping patches and projected via linear layers, yielding $kn + n$ tokens in total. The resulting token sequence is then processed through a stack of Bi-SSM and Transformer blocks. Finally, the last n output tokens are mapped to RGB values via a linear layer, producing the refined target-view image I_t^T.

Empirically, a single Bi-SSM block requires only about half the parameters of a Transformer block, enabling us to build a deeper refinement pipeline under the same parameter budget. The additional depth offers more opportunities to progressively refine coarse predictions, resulting in higher-quality reconstructions. However, the recurrent smoothing nature of SSMs limits their ability to recover fine-grained local details. Thus, rather than stacking only Bi-SSM blocks, we adopt the Transformer–SSM hybrid architecture to build a deeper pipeline while remaining parameter efficient. Further architectural details of the view refinement module can be found in the supplementary material.

Table 1: Quantitative comparison on Neural 3D Video. Since IGS [38] trains its network on four scenes of Neural 3D Video, we compare IGS only on the remaining two scenes (*cut roasted beef, sear steak*) and additionally report our results on the same two scenes, denoted as †.

Method	Pose	Initialization	FPS↑	PSNR↑	SSIM↑	LPIPS↓	Flicker↓
3DGS [15]	✓	✓	0.1	19.22	0.532	0.394	85.10
3DGStream [34]	✓	✓	0.6	20.38	0.580	0.536	14.88
HiCoM [10]	✓	✓	0.7	19.63	0.583	0.420	13.96
DepthSplat [36]	✓	✗	9.7	19.81	0.692	0.324	19.26
RayZer [12]	✗	✗	19	21.97	0.596	0.287	24.16
Ours	✗	✗	**32**	**23.63**	**0.715**	**0.199**	**8.97**
IGS [38]†	✓	✓	5.5	21.23	0.653	0.425	**6.40**
Ours†	✗	✗	**32**	**24.62**	**0.737**	**0.191**	8.76

3.3 Multi-View Dynamic Scene Dataset

For training and evaluation of our approach, we introduce a new multi-view dynamic scene dataset. Existing real-world multi-view dynamic scene datasets for novel-view synthesis such as Neural 3D Video [20] (6 scenes), Immersive Light Field Video [2] (15 scenes), and SelfCap [37] (6 scenes) often contain only a few scenes and are captured with fixed cameras, making them less suitable for exploring practical streaming scenarios where both cameras and objects are moving. To address this limitation, our dataset consists of 150 dynamic real-world scenes captured with a handheld three-camera rig, targeting more realistic streaming conditions. The dataset is recorded at a resolution of 1920×1200 and 30 FPS, providing 3 synchronized views per frame with an average sequence length of 338 frames per scene. We will publicly release the dataset for future research. More capture details are provided in the supplementary material.

3.4 Training

To train our framework, we adopt a two-phase training strategy. In the first phase, the pose estimator and scene reconstruction modules are jointly trained in a self-supervised manner without ground-truth pose supervision. At each iteration, the model reconstructs consecutive novel-view frames and is trained using only image reconstruction loss. Since the pose mapper is not yet available, the target latent pose $\tilde{p}_t^T$ is computed by applying the pose estimator to the ground-truth target frame. Although pose supervision is not provided, the pose estimator learns a latent pose space through its contribution to reconstruction quality. In the second phase, we train the pose mapper in a supervised manner while all other modules are frozen, enabling physical poses to be mapped into the latent space learned in the first phase.

We use a photometric loss $\mathcal{L}_{\text{photo}}$ and a flicker loss $\mathcal{L}_{\text{flicker}}$ to encourage photometric consistency and temporal coherence between the predicted and ground-truth target-view frames. Specifically, we define our training loss as:

$$\mathcal{L}(I_t^T, I_t^{GT}) = \mathcal{L}_{\text{photo}}(I_t^T, I_t^{GT}) + \lambda_{\text{flicker}} \mathcal{L}_{\text{flicker}}(I_t^T, I_t^{GT}), \tag{5}$$

Fig. 4: Qualitative comparison on Neural 3D Video.

where I_t^{GT} is the ground-truth target-view frame corresponding to the predicted target-view frame I_t^T, and λ_{flicker} is a weight for the flicker loss. Following prior work [12, 13], we define the photometric loss as:

$$\mathcal{L}_{\text{photo}} = \mathcal{L}_{\text{MSE}}(I_t^T, I_t^{GT}) + \lambda_{\text{per}}\mathcal{L}_{\text{per}}(I_t^T, I_t^{GT}), \tag{6}$$

where $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{per}}$ are an MSE loss and a perceptual loss [14], respectively, and λ_{per} is a weight for the perceptual loss.

Since our framework processes video sequences, ensuring temporal consistency is critical for stable outputs. In practice, temporal incoherence typically appears as abrupt changes in high-level structures rather than raw pixel values. Following previous video-based work [6], we encourage temporal consistency by matching the VGG feature [33] differences between consecutive frames in both prediction and ground truth. Specifically, we define the flicker loss $\mathcal{L}_{\text{flicker}}$ as:

$$\mathcal{L}_{\text{flicker}} = \left\| [\phi(I_t^T) - \phi(I_{t-1}^T)] - [\phi(I_t^{GT}) - \phi(I_{t-1}^{GT})] \right\|_2, \tag{7}$$

where ϕ denotes a pretrained VGG network. The flicker loss computes the feature differences between two consecutive frames in both ground-truth and prediction, and enforces these differences to be similar, ensuring consistent temporal transitions. Since both the history fusion and view refinement modules produce target-view predictions, we apply the training loss to both outputs: $\mathcal{L}(I_t^H, I_t^{GT}) + \mathcal{L}(I_t^T, I_t^{GT})$.

In the first phase, we train our framework on RealEstate10K [41], which consists of approximately 80K indoor and outdoor video clips, for 100K iterations. Since RealEstate10K contains only static scenes, it provides a simpler setting that stabilizes training and allows the model to learn generalizable multi-view reconstruction from large-scale static data. Specifically, the model learns to reconstruct consecutive 8-frame video sequences, generating novel-view frames at each time step.

In the second phase, we fine-tune the model for an additional 20K iterations to reconstruct longer 64-frame video sequences on a mixture of static and dynamic datasets. RealEstate10K serves as the static dataset for co-training, while

Table 2: Quantitative comparison on our multi-view dynamic scene dataset.

Method	Pose	Initialization	FPS↑	PSNR↑	SSIM↑	LPIPS↓	Flicker↓
3DGStream [34]	✓	✓	0.6	19.52	0.491	0.501	50.24
HiCoM [10]	✓	✓	0.7	18.19	0.441	0.599	62.88
IGS [38]	✓	✓	5.5	19.96	0.466	0.508	62.12
DepthSplat [36]	✓	✗	9.7	19.72	0.561	0.386	49.51
RayZer [12]	✗	✗	19	21.57	0.503	0.434	52.66
Ours	✗	✗	**32**	**23.26**	**0.617**	**0.274**	**45.45**

the dynamic data is drawn from our dataset together with SelfCap [37]. This mixture of static and dynamic supervision improves robustness under dynamic scenes while maintaining stability from large-scale static multi-view data. We then train the pose mapper for 20K iterations with a batch size of 64, using only the photometric loss. Additional training details are provided in the supplementary material.

4 Experiments

4.1 Evaluation Protocol

We evaluate our method under two different streaming setups: fixed-camera and moving-camera. In the fixed-camera setup, we use the Neural 3D Video dataset [20], which consists of six dynamic multi-view indoor scenes captured with static cameras. In the moving-camera setup, we use the test split of our proposed streaming dataset consisting of 30 real-world dynamic scenes recorded with handheld moving cameras. For each scene in both datasets, we use two input views, each with a 300-frame sequence at a resolution of 256.

We compare our method against existing streamable approaches, including 3DGStream [34], HiCoM [10], and IGS [38], as well as per-frame reconstruction of 3DGS [15]. This comparison demonstrates the limitations of existing pipelines under sparse-view streaming conditions. We additionally compare with the per-frame reconstruction using DepthSplat [36], a feedforward model that predicts 3D Gaussians from multi-view images. Unlike the above streamable methods, DepthSplat does not require per-scene initialization, while still relying on pre-computed poses and an explicit 3D representation. Finally, we compare with the per-frame synthesis using RayZer [12], a feedforward NVS method that directly predicts novel-view images from unposed inputs. As RayZer operates under the same geometry-free conditions without ground-truth pose supervision, this serves as our most direct baseline.

For evaluation metrics, we use PSNR, SSIM, and LPIPS to measure the rendering quality. We also use a flicker loss to assess temporal consistency, as defined in Eq. (7). For runtime speed, we record frames per second (FPS) for every method. The FPS of 3DGStream, HiCoM, and IGS is reported excluding Gaussian initialization and pose estimation time, and that of DepthSplat excludes pose estimation time. The FPS of Ours and RayZer is measured for the full feedforward pipeline, from input images to novel-view synthesis, including

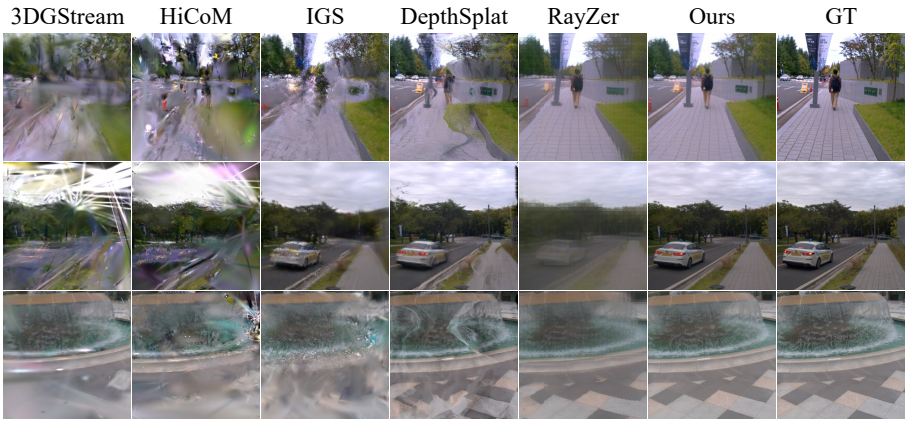

Fig. 5: Qualitative comparison on our dataset. Our method shows strong visual quality under in-the-wild streaming conditions, while prior streamable methods struggle to handle changing backgrounds or newly emerging objects that were not reconstructed during Gaussian initialization.

pose estimation and mappings. All experiments, both quantitative and qualitative evaluations, are conducted on a single NVIDIA A100 GPU. Ablation studies are performed on the RealEstate10K [41] test split of pixelSplat [3]. More evaluation details are provided in the supplementary material.

4.2 Streaming Benchmark with Fixed Cameras

We compare REON-NVS across all six scenes of Neural 3D Video [20]. As shown in Tab. 1, our method achieves the highest novel-view synthesis performance while running in real time at 32 FPS. Furthermore, unlike existing streamable approaches, our method requires no offline preprocessing over the full video sequence, such as physical pose estimation or 3D Gaussian initialization. DepthSplat [36] relies on ground-truth camera poses, yet REON-NVS achieves higher reconstruction quality, suggesting that our geometry-free formulation is highly competitive. Compared to RayZer [12], which shares our geometry-free design without pose supervision, REON-NVS achieves +1.6 dB higher PSNR and significantly lower flicker. This is because RayZer compresses inputs into a latent representation and lacks temporal modeling, whereas REON-NVS directly leverages input images with SSM-based temporal memory. As illustrated in Fig. 4, REON-NVS produces high-quality, photorealistic results, whereas prior streamable methods often fail to synthesize realistic frames from the same sparse-view inputs.

4.3 Streaming Benchmark with Moving Cameras

We compare REON-NVS on our dataset containing 30 dynamic scenes captured with handheld moving cameras. This benchmark features real-world challenges

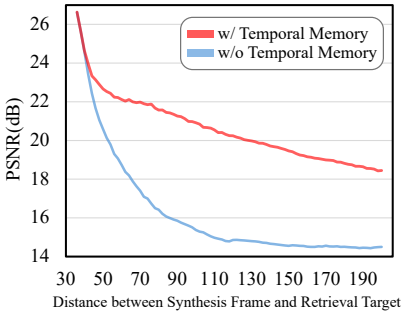

Fig. 6: Retrieval task performance. Using the SSM hidden state as temporal memory, our method achieves stable PSNR across hundreds of frames without relying on 3D structures and incurring extra computation.

Fig. 7: Qualitative result of retrieval task. Our model leverages the hidden state of SSM as temporal memory, enabling the reconstruction of current inputs' out-of-view regions from past observations.

such as camera shaking, dynamic lighting, and scene content changes over time. As shown in Tab. 2, our method achieves superior FPS and visual quality under in-the-wild streaming scenarios. Moreover, as illustrated in Fig. 5, prior streamable methods based on 3D Gaussians often fail to handle newly appearing objects and backgrounds that arise from camera motion or object movement, since these regions were not contained in the initialized 3D Gaussians. In contrast, our framework operates in a geometry-free, per-frame feedforward manner, maintaining robustness to newly visible regions. DepthSplat also achieves lower quality despite using ground-truth poses, consistent with our observation in the fixed-camera setting. Compared to RayZer, REON-NVS also achieves +1.7 dB higher PSNR with lower flicker, confirming the advantage of directly leveraging input images with temporal modeling.

4.4 Ablation Study

Temporal Memory. To directly evaluate the ability to leverage temporal information from past frames, we compare our scene reconstructor with a variant that disables temporal hidden-state propagation in the history fusion module. For evaluation, we design a retrieval task where the target viewpoint revisits regions observed in earlier frames but no longer visible in the current input views.

As shown in Fig. 6, our framework can leverage information from hundreds of past timesteps, maintaining stable PSNR compared to the variant without temporal hidden-state propagation. Fig. 7 further illustrates that our model can faithfully recover regions that are no longer visible in the current inputs by recalling past observations. Notably, this is achieved in real time without relying on 3D structures or incurring additional computational cost.

Table 3: Ablation study on the flicker loss.

Dataset	Neural 3D Video [20]				Our Dataset				RealEstate10K [41]			
	PSNR↑	SSIM↑	LPIPS↓	Flicker↓	PSNR↑	SSIM↑	LPIPS↓	Flicker↓	PSNR↑	SSIM↑	LPIPS↓	Flicker↓
w/o $\mathcal{L}_{\text{flicker}}$	23.41	0.706	0.211	10.27	23.09	0.601	0.295	46.30	26.67	0.874	0.119	20.96
w/ $\mathcal{L}_{\text{flicker}}$	**23.63**	**0.715**	**0.199**	**8.97**	**23.26**	**0.617**	**0.274**	**45.45**	**27.22**	**0.882**	**0.112**	**19.91**

Flicker Loss. In Tab. 3, we compare our model trained with and without the flicker loss $\mathcal{L}_{\text{flicker}}$ to evaluate its contribution to temporal stability. We evaluate on both dynamic and static datasets, including Neural 3D Video [20], our proposed dataset, and RealEstate10K [41]. The results show that incorporating the flicker loss enables our framework to capture temporal interactions more effectively, thereby reducing flicker and improving reconstruction quality.

Bidirectional SSM. We ablate various SSM architectures in Tab. 4. Specifically, we evaluate the reconstruction performance of the history fusion module, which is composed entirely of Bi-SSM blocks. Here, 'Token Merge' denotes the strategy used to merge two unidirectionally scanned tokens for bidirectional scanning, while 'Cross' refers to alternating the bidirectional scanning direction between rows and columns in each Bi-SSM block.

Table 4: Reconstruction performance of the history fusion module with various SSM architectures. *Token Merge* denotes the method of merging two unidirectional scans, while *Cross* denotes alternating scan directions across Bi-SSM blocks.

Bidirection	Cross	Token Merge		PSNR↑
		Concat	Add	
				19.79
✓		✓		20.82
✓			✓	21.60
✓	✓	✓		21.08
✓	✓		✓	**22.04**

The results show that the bidirectional scanning strategy significantly improves NVS performance compared to the unidirectional baseline, the same as vanilla Mamba2 [7]. Moreover, alternating the scanning direction in each SSM block further enhances reconstruction quality without incurring additional computational cost. The best performance is achieved with add-based token merging, which outperforms concatenation. More details on our SSM architecture are provided in the supplementary material.

5 Conclusion

In this paper, we introduced REON-NVS, a novel feedforward NVS framework for online reconstruction of dynamic scenes from sparse-view input video streams. Our method addresses the key challenges that prevent existing streamable approaches from being practical in unrestricted streaming scenarios: the need for a large number of viewpoints and reliance on time-consuming external pose estimation. To this end, we perform both pose estimation and scene reconstruction in a fully feedforward manner, achieving real-time performance at 32 FPS. We also capture a new dynamic multi-view video dataset of 150 real-world scenes to explore realistic streaming scenarios, which also provides a challenging benchmark for evaluating streamable NVS methods. Our experiments verify that REON-NVS maintains comparable visual quality while operating in real time, demonstrating practical applicability to real-world streaming scenarios.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant (No. RS-2026-25492695) and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2026-25517417, Development of Compression, Reconstruction, and Rendering Technologies for Free-viewpoint Media; No. RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH)).

References

1. Brégier, R.: Deep regression on manifolds: A 3D rotation case study. In: 3DV (2021)
2. Broxton, M., Flynn, J., Overbeck, R., Erickson, D., Hedman, P., Duvall, M., Dourgarian, J., Busch, J., Whalen, M., Debevec, P.: Immersive light field video with a layered mesh representation. ACM TOG **39**(4), 86–1 (2020)
3. Charatan, D., Li, S.L., Tagliasacchi, A., Sitzmann, V.: pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In: CVPR (2024)
4. Chen, J., Yin, Y., Birdal, T., Chen, B., Guibas, L.J., Wang, H.: Projective manifold gradient layer for deep rotation regression. In: CVPR (2022)
5. Chen, Y., Xu, H., Zheng, C., Zhuang, B., Pollefeys, M., Geiger, A., Cham, T.J., Cai, J.: MVSplat: Efficient 3D Gaussian Splatting from sparse multi-view images. In: ECCV (2024)
6. Chu, M., Xie, Y., Mayer, J., Leal-Taixe, L., Thuerey, N.: Learning temporal coherence via self-supervision for GAN-based video generation (TecoGAN). ACM TOG **39**(4) (2020)
7. Dao, T., Gu, A.: Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In: ICML (2024)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
9. Fink, L., Rückert, D., Franke, L., Keinert, J., Stamminger, M.: LiveNVS: Neural view synthesis on live RGB-D streams. In: SIGGRAPH Asia (2023)
10. Gao, Q., Meng, J., Wen, C., Chen, J., Zhang, J.: HiCoM: Hierarchical coherent motion for dynamic streamable scenes with 3D Gaussian Splatting. In: NeurIPS (2024)
11. Ha, H., Xiao, L., Richardt, C., Nguyen-Phuoc, T., Kim, C., Kim, M.H., Lanman, D., Khan, N.: Geometry-guided online 3D video synthesis with multi-view temporal consistency. In: CVPR (2025)
12. Jiang, H., Tan, H., Wang, P., Jin, H., Zhao, Y., Bi, S., Zhang, K., Luan, F., Sunkavalli, K., Huang, Q., Pavlakos, G.: RayZer: A self-supervised large view synthesis model. In: ICCV (2025)
13. Jin, H., Jiang, H., Tan, H., Zhang, K., Bi, S., Zhang, T., Luan, F., Snavely, N., Xu, Z.: LVSM: A large view synthesis model with minimal 3D inductive bias. In: ICLR (2025)
14. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)

15. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for real-time radiance field rendering. ACM TOG **42**(4), 1–14 (2023)
16. Lai, Z., Liu, S., Efros, A.A., Wang, X.: Video autoencoder: Self-supervised disentanglement of static 3D structure and motion. In: ICCV (2021)
17. Lee, J., Won, C., Jung, H., Bae, I., Jeon, H.G.: Fully explicit dynamic Gaussian Splatting. In: NeurIPS (2024)
18. Levinson, J., Esteves, C., Chen, K., Snavely, N., Kanazawa, A., Rostamizadeh, A., Makadia, A.: An analysis of SVD for deep rotation estimation. In: NeurIPS (2020)
19. Li, L., Shen, Z., Wang, Z., Shen, L., Tan, P.: Streaming radiance fields for 3D video synthesis. In: NeurIPS (2022)
20. Li, T., Slavcheva, M., Zollhoefer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., Lv, Z.: Neural 3D video synthesis from multi-view video. In: ICCV (2022)
21. Li, Z., Chen, Z., Li, Z., Xu, Y.: Spacetime Gaussian feature Splatting for real-time dynamic view synthesis. In: CVPR (2024)
22. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR (2021)
23. Liang, H., Ren, J., Mirzaei, A., Torralba, A., Liu, Z., Gilitschenski, I., Fidler, S., Oztireli, C., Ling, H., Gojcic, Z., Huang, J.: Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. In: NeurIPS (2026)
24. Lin, C., Lin, Y., Pan, P., Yu, Y., Yan, H., Fragkiadaki, K., Mu, Y.: MoVieS: Motion-aware 4D dynamic view synthesis in one second. In: CVPR (2026)
25. Lin, H., Chen, S., Liew, J.H., Chen, D.Y., Li, Z., Shi, G., Feng, J., Kang, B.: Depth Anything 3: Recovering the visual space from any views. In: ICLR (2026)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
27. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM TOG **41**(4), 1–15 (2022)
28. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV (2021)
29. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: HyperNeRF: A higher-dimensional representation for topologically varying neural radiance fields. ACM TOG **40**(6), 238:1–238:12 (2021)
30. Plucker, J.: XVII. on a new geometry of space. Philosophical Transactions of the Royal Society of London (155), 725–791 (1865)
31. Sajjadi, M.S., Meyer, H., Pot, E., Bergmann, U., Greff, K., Radwan, N., Vora, S., Lučić, M., Duckworth, D., Dosovitskiy, A., et al.: Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In: CVPR (2022)
32. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
34. Sun, J., Jiao, H., Li, G., Zhang, Z., Zhao, L., Xing, W.: 3DGStream: On-the-fly training of 3D Gaussians for efficient streaming of photo-realistic free-viewpoint videos. In: CVPR (2024)
35. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4D Gaussian Splatting for real-time dynamic scene rendering. In: CVPR (2024)
36. Xu, H., Peng, S., Wang, F., Blum, H., Barath, D., Geiger, A., Pollefeys, M.: DepthSplat: Connecting Gaussian Splatting and depth. In: CVPR (2025)

37. Xu, Z., Xu, Y., Yu, Z., Peng, S., Sun, J., Bao, H., Zhou, X.: Representing long volumetric video with temporal Gaussian hierarchy. ACM TOG **43**(6), 1–18 (2024)
38. Yan, J., Peng, R., Wang, Z., Tang, L., Yang, J., Liang, J., Wu, J., Wang, R.: Instant Gaussian stream: Fast and generalizable streaming of dynamic scene reconstruction via Gaussian Splatting. In: CVPR (2025)
39. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In: CVPR (2024)
40. Ye, B., Liu, S., Xu, H., Li, X., Pollefeys, M., Yang, M., Peng, S.: No pose, no problem: Surprisingly simple 3D Gaussian splats from sparse unposed images. In: ICLR (2025)
41. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM TOG (2018)
42. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR (2019)
43. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision Mamba: Efficient visual representation learning with bidirectional state space model. In: ICML (2024)
44. Ziwen, C., Tan, H., Zhang, K., Bi, S., Luan, F., Hong, Y., Fuxin, L., Xu, Z.: Long-LRM: Long-sequence large reconstruction model for wide-coverage Gaussian splats. In: ICCV (2025)