Phase-DGS: Phase-guided Dynamic Gaussian Splatting from Unsynchronized Multi-view Video

Hosung Jeon[†1] **Jun Young Jeong**[1] **Sangwoon Kwak**[†1] **Joonsoo Kim**[1]
Sangmin Kim[2] **Jaesik Park**[2] **Won-Sik Cheong**[1] **Hyon-Gon Choo**[1]

[1]Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea
[2]Seoul National University, Seoul, Republic of Korea

{h.jeon, jyj0120, s.kwak, joonsoo.kim, wscheong, hgchoo}@etri.re.kr
{sm.kim, jaesik.park}@snu.ac.kr

Abstract

Spatiotemporal scene reconstruction with Dynamic Gaussian Splatting (DGS) fundamentally depends on perfect temporal synchronization across all cameras, an assumption rarely achievable in practice. Industry-standard synchronization requires expensive specialized hardware and complex setup, while data-driven alternatives including audio-based alignment, geometry matching, and pose tracking remain limited by restrictive environmental or scene-specific assumptions. To overcome these limitations, we propose Phase-DGS, a framework that replaces conventional timestamps with a semantic phase representing each frame's inherent state within the underlying motion cycle. Frames capturing the same motion state share the same phase regardless of which camera records them or when, naturally establishing spatiotemporal correspondence across unsynchronized cameras. We validate Phase-DGS under temporal offsets, frame drops, and camera freezes, achieving up to +8.5 dB PSNR improvement over baselines while enabling seamless integration with existing DGS backbones including RealTime-4DGS and FreeTimeGS.

1 Introduction

In recent years, radiance field representations such as Neural Radiance Fields (NeRF) [6, 21, 37] and 3D Gaussian Splatting (3DGS) [11, 18, 42] have been extended to dynamic scenes, leading to Dynamic NeRF (DNeRF) and Dynamic Gaussian Splatting (DGS), which have emerged as a leading paradigm for spatiotemporal scene reconstruction. The latest of these methods [15, 24, 34, 35] reconstruct complex real-world scenes with striking realism, enabling applications spanning from autonomous navigation to immersive content creation. However, such results come with a prerequisite: perfect temporal synchronization across all cameras, with each camera capturing the scene at exactly the same instant, frame-by-frame [12, 14, 39]. Without such synchronization, observations from different viewpoints no longer correspond to the same scene instant, making it ill-posed to learn a consistent radiance field under a shared timestamp. This spatiotemporal incoherence inevitably produces floaters, severe ghosting, and geometric distortion in rendered views.

A straightforward approach is to enforce camera synchronization during the capture stage through standardized solutions, providing the radiance field with already-aligned frames that guarantee spatiotemporal coherence. Trigger-based methods such as Genlock [27, 30, 31] synchronize all camera shutters via a master sync signal, achieving high precision. However, depending on the capture environment, trigger signals can be degraded by attenuation and noise, and maintaining high-

[†]Co-corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

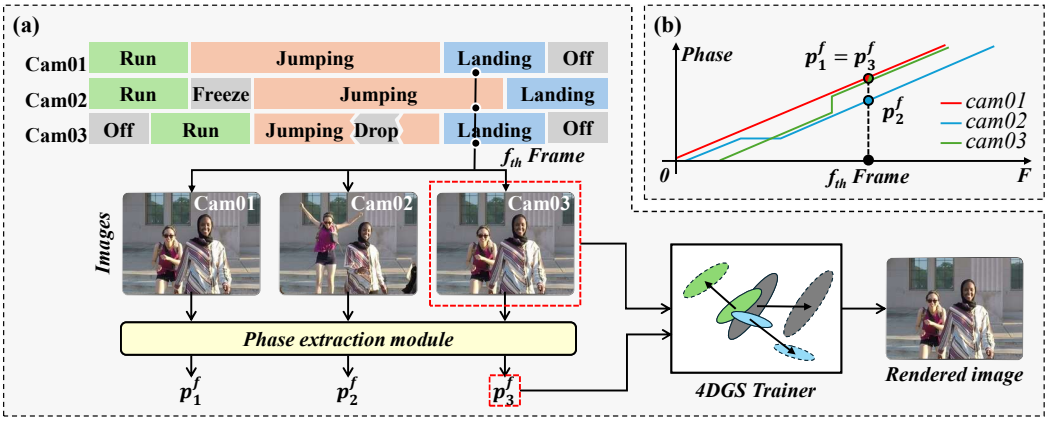


Figure 1: Conceptual diagram of Phase-DGS. (a) In an unsynchronized multi-view dataset, images sharing the frame index are actually captured at different time instants. The proposed phase extraction module estimates an inherent semantic phase from each input image, which is then used to train 4DGS. (b) The graph shows the estimated phases across cameras. When a sequence is offset, the phase shifts along the *x-axis* direction; when frozen, the phase is predicted parallel to the *x-axis*; and when dropped, the phase is predicted parallel to the *y-axis*.

quality synchronization requires expensive specialized hardware and complex wiring. Protocol-based methods such as the Precision Time Protocol (PTP) [9] and Network Time Protocol (NTP) [1] avoid dedicated wiring and offer synchronization at reasonable cost, but rely heavily on network bandwidth and stability. When transmission delays or packet losses occur, frames can be dropped, or cameras may suffer temporal drift in which their time axes gradually diverge from one another.

Beyond capture-stage synchronization, recent works have explored training radiance fields with improved spatiotemporal coherence by using cues extracted from the captured data itself. Audio-based alignment methods [2, 8, 28, 29, 33] synchronize cameras by matching audio signals across microphones. However, they rely on a carefully controlled acoustic environment, a condition that rarely holds in open-capture settings. Geometry- and feature-matching methods [5, 7, 17, 36] estimate the degree of unsynchrony by finding spatiotemporal correspondences across views, but they struggle with weak scene texture or large viewpoint differences. Human- and pose-tracking approaches [3, 10, 25, 32, 40] align frames by tracking human body poses across cameras, but such reliance restricts their use to human-centric scenarios where a person remains visible across all frames. Reconstruction-coupled methods [12, 14, 39] jointly optimize sub-frame-level temporal misalignment together with the radiance field while most assign only a single offset per camera. Furthermore, these methods require camera pose information, which is typically obtained from synchronized data or provided as known calibration in the dataset. In unsynchronized recordings, however, temporal misalignment makes accurate camera pose estimation difficult.

Instead of estimating absolute temporal offsets to synchronize videos, we infer an inherent semantic phase for each image that represents its position within the dynamic instance. Based on this idea, we propose Phase-DGS, a framework that integrates a phase extraction module into 4D Gaussian Splatting (4DGS) to learn a radiance field with high spatiotemporal coherence, where images sharing the same phase naturally establish spatiotemporal correspondence regardless of which camera captures them or when. As shown in Fig. 1, even when cameras are unsynchronized and their frame indices no longer correspond to the same instant, our phase extraction module assigns the same inherent semantic phase to images capturing the same motion state, such as the "jumping" pose, across different cameras. The estimated phases then replace absolute timestamps as input to 4DGS, allowing it to reconstruct spatiotemporally coherent rendered images. Its modular design further enables plug-and-play integration with state-of-the-art 4DGS backbones, as validated on RealTime-4DGS [35] and FreeTimeGS [34]. Experimentally, we validate that the proposed method optimizes 4DGS with enhanced spatiotemporal coherence under various frame-level errors, including temporal offsets, frame drops, and camera freezes. Our method achieves up to +8.5 dB improvement in PSNR over baseline methods under unsynchronized conditions.

2 Related Works

Existing synchronization approaches fall into several categories, each with its own limitations. Hardware- and software-based methods such as Genlock, Timecode, and PTP offer high precision but provide no post-hoc correction for timing errors induced by signal drift, and their cost remains a barrier to practical deployment. **Audio-based methods** [2, 8, 29, 28, 33] align videos through shared acoustic events but rely on a controlled acoustic environment and suffer from non-negligible audio-visual latency in large-scale setups, where the finite speed of sound introduces meaningful error. **Geometry and feature matching methods** [5, 7, 16, 20] estimate the amount of unsynchrony by finding spatiotemporal correspondences across views, and they are highly dependent on the quality of feature extraction and degrade considerably under the occlusion of tracked features. **Approaches based on human or pose tracking** [3, 10, 25, 32, 40] align frames by tracking human body poses across cameras, and strictly require the presence of humans in the scene. It substantially limits their applicability to common real-world environments. **Learning-based methods** such as Sync-NeRF [12, 14, 39] jointly optimize temporal offsets as learnable parameters by using rendering quality as a supervisory signal, yet they assume continuous frame availability across all views and fundamentally cannot recover from per-camera frame drops and freezes that arise independently across cameras.

3 Method

3.1 Overview

Phase-DGS replaces the absolute timestamp used in conventional 4DGS with a per-frame phase derived from the image content. We observe that each frame possesses an inherent semantic phase within video sequences, independent of its absolute timestamp. For instance, in a sit-to-stand motion, the "fully seated" pose corresponds to the same phase regardless of which camera captures it or at what moment. Based on this observation, Phase-DGS extracts a phase p_i^f from each camera image u_i^f and uses it instead of the timestamp t to supervise 4DGS training. Note that f and i represent the frame and camera indices, respectively. Since the phase is derived from the image content, it naturally accommodates unsynchronized capture conditions such as frame drops, frame freezes, and temporal offsets between cameras.

3.2 Phase Extraction Module

We introduce the phase projector module, which extracts a phase p_i^f from each camera image u_i^f using a pretrained Vision Transformer (ViT) encoder [4, 23] followed by a lightweight MLP.

Feature Generation. The input image u_i^f of the f-th frame of the i-th camera is coarsely cropped to a motion-intensive region of interest (ROI) and resized to match the input resolution of the ViT encoder. Then, the cropped image is passed through a pretrained ViT encoder, which partitions it into $H \times W$ patches and generates a token for each patch. We spatially aggregate all patch tokens by averaging to obtain a d-dimensional global feature vector $z_i^f \in \mathbb{R}^d$ that summarizes the visual content of the image:

$$z_i^f = \frac{1}{HW} \sum_{n=1}^{H \times W} \left[\mathrm{ViT}_{\mathrm{enc}} \left(u_i^f \right) \right]_n . \tag{1}$$

Phase Projector. The global feature vector z_i^f is subsequently passed through a lightweight MLP that consists of three fully connected layers, which produces a normalized phase value $p_i^f \in [0, 1]$,

$$p_i^f = \mathrm{MLP} \left(z_i^f \right) . \tag{2}$$

Notably, the phase projector is trained without any explicit phase supervision; instead, it is optimized through the rendering loss together with the phase consistency and monotonic losses (see Sections 3.4 and 3.6), enabling unsupervised phase extraction directly from visual observations.

3.3 Feature Bank

The feature bank $\mathcal{B}$ serves as a learnable reference repository that enforces cross-camera phase coherence, ensuring that identical motion states share the same phase across all views by enabling the phase projector to map each input feature to a phase through comparison against stored references. The feature bank is defined as $\mathcal{B} \in \mathbb{R}^{I\times(F\times k)\times d}$, where I is the number of cameras, F is the video length, k is an upsampling factor, and d is the dimension of the feature. $k > 1$ is designed to handle videos with varying frame rates or staggered frame timestamps on some cameras, providing finer temporal resolution in the feature bank. Since the DINOv2 features are extracted independently from each image and do not contain temporal information, we make the feature bank learnable rather than fine-tuning the entire ViT encoder for every scene. Inspired by temporal prototype learning [22], which aligns multiple videos through learned prototypes, the feature bank is optimized jointly with the phase projector to adapt to the scene, so that frames capturing the same motion state are assigned the same phase across all views. When the feature bank is fixed, which corresponds to a fixed-memory alternative, the accuracy of phase estimation drops substantially (see Section 4.5).

3.4 Loss Function

Phase-DGS is trained with a combination of three loss functions,

$$\mathcal{L} = \lambda_{phase}\mathcal{L}_{phase} + \lambda_{mono}\mathcal{L}_{mono} + \lambda_{render}\mathcal{L}_{render}. \tag{3}$$

Phase Consistency Loss. $\mathcal{L}_{phase}$ supervises the phase projector so that it retrieves features with matching phases from the feature bank. Given an input image u_i^f, the extracted feature z_i^f is passed through the phase projector to obtain the phase p_i^f. The entire i-th camera array of the feature bank, $\mathcal{B}_i \in \mathbb{R}^{(F\times k)\times d}$, is then passed through the phase projector to generate a phase map $\mathcal{P}_i \in \mathbb{R}^{(F\times k)\times 1}$. An attention weight w is computed by applying softmax to the distance between the phase map and p_i^f,

$$w = \text{softmax}\left(-\left|\mathcal{P}_i - p_i^f\right|\right). \tag{4}$$

The reference feature $\hat{z}_i^f$ that shares the same phase as p_i^f is then retrieved through a weighted sum over the feature bank,

$$\hat{z}_i^f = \sum w \cdot \mathcal{B}_i. \tag{5}$$

The loss is defined as the MSE between z_i^f and $\hat{z}_i^f$,

$$\mathcal{L}_{phase} = \text{MSE}\left(z_i^f, \hat{z}_i^f\right). \tag{6}$$

Monotonic Loss. $\mathcal{L}_{mono}$ promotes the phase map generated from the feature bank to increase monotonically along the temporal axis. The entire feature bank is passed through the phase projector to produce $\mathcal{P} \in \mathbb{R}^{I\times(F\times k)\times 1}$, and the difference between adjacent phases is computed for each camera,

$$\Delta_i^f = \mathcal{P}_i^{f+1} - \mathcal{P}_i^f, \quad f = 0, 1, \ldots, Fk-2. \tag{7}$$

A penalty is applied whenever the phase difference falls below the minimum increment $a = \frac{1}{F\times k}$,

$$\mathcal{L}_{mono} = \sum_i \sum_f \text{relu}\left(-\Delta_i^f + \frac{1}{Fk}\right). \tag{8}$$

This term encourages the phase projector to learn temporally consistent phases across the feature bank.

Rendering Loss. $\mathcal{L}_{render}$ optimizes the rendering quality of 4DGS by minimizing the visual discrepancy between the rendered images and the ground-truth observations.

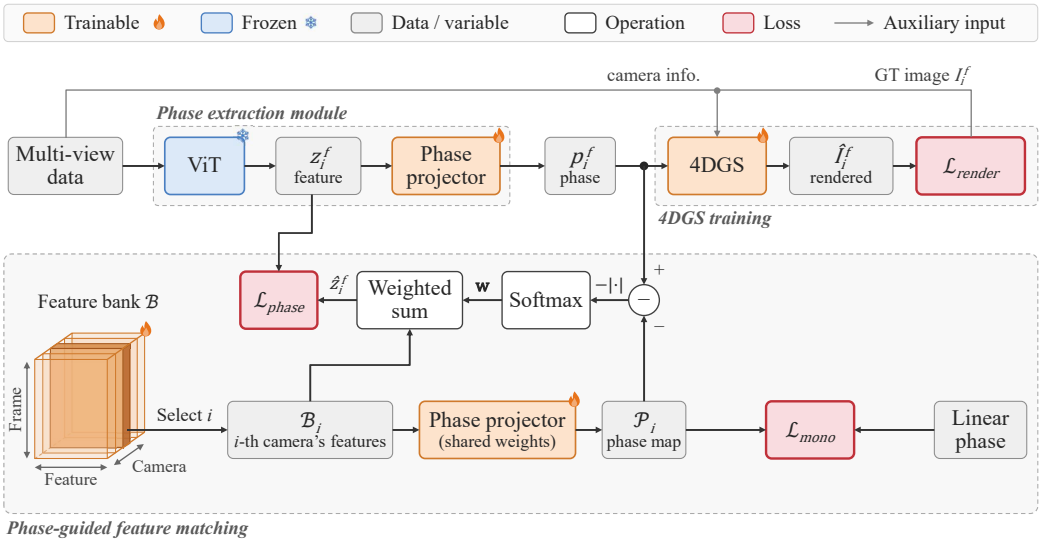


Figure 2: Training pipeline of Phase-DGS. Multi-view data provides camera parameters (view matrix, focal length, resolution) and an RGB image. The phase projector module predicts a phase value from the RGB image, replacing the timestamp input of the 4DGS backbone. The feature bank serves as a learnable reference repository that enforces cross-camera phase coherence, so that identical motion states share the same phase across all views. Training is guided by three losses: a Rendering loss between the rendered and input images, a monotonic loss, and a phase consistency loss.

3.5 Integration with Dynamic Gaussian Splatting

Existing 4DGS methods model the deformation of 3D Gaussians over time by taking the absolute timestamp t of each frame as input [35, 34]. However, as discussed above, absolute timestamps are inherently unreliable in real-world capture environments because of inter-camera synchronization errors, frame drops, and frame freezes.

To address this issue, we replace the timestamp t in 4DGS model $\mathcal{G}$ with the phase p_i^f extracted by Phase-DGS, as shown in Fig. 1. Since p_i^f is extracted independently from the image content of each camera, it naturally compensates for inter-camera synchronization errors, frame drops, and frame freezes without requiring any explicit timestamp correction. In other words, whereas conventional 4DGS optimizes Gaussians along the temporal axis, Phase-DGS optimizes Gaussians along the phase axis,

$$\mathcal{G}(p_i^f) \leftarrow \mathcal{G}(t). \tag{9}$$

Note that the phase is designed as a one-dimensional scalar because it plays the same role as the timestamp t. Since existing 4DGS methods take a scalar timestamp as input [35, 34], the phase can be integrated in a plug-and-play manner without modifying the backbone. Furthermore, all objects within a frame are captured at the same instant, so a single phase per frame is sufficient to represent the temporal state of the scene regardless of the number of moving objects. The motion of each object, including complex motions and multi-object interactions, is still modeled by the deformation of the 4DGS backbone, as in the timestamp-based setting.

3.6 Training

Phase-DGS is trained end-to-end by jointly optimizing a plug-and-play 4DGS backbone as shown in Fig. 2 (e.g., RealTime-4DGS [35], FreeTimeGS [34]), the phase projector, and the feature bank under the combined loss in Eq. (3). The ViT encoder remains frozen throughout the training, while all other modules are updated by gradient descent. In each iteration, a frame u_i^f is sampled from the camera i and passed through the ViT encoder and the phase projector to produce the phase p_i^f. The estimated phase replaces the absolute timestamp t as the temporal coordinate for the 4DGS backbone, and all three losses are back-propagated simultaneously.

4 Experimental results

4.1 Implementation Details.

Phase Extraction Module. For the ViT encoder, we adopt DINOv2 [23] with the `dinov2_vitb14` configuration. The phase projector (see Section 3.2) MLP follows a $d \to 256 \to 128 \to 64 \to 1$ architecture, in which each hidden layer is activated by GELU and the final output is normalized to $[0, 1]$ by a Sigmoid activation.

4DGS Backbones. We primarily adopt FreeTimeGS [34], which we re-implement based on the original paper, and we additionally validate plug-and-play compatibility with RealTime-4DGS [35]. These two methods constitute the state of the art in 4DGS and represent the *explicit* and *implicit* paradigms of modeling Gaussian motion, respectively.

Training Details. Phase-DGS is trained with the Adam optimizer and a batch size of one camera per iteration. The shared hyperparameters across both backbones are a learning rate of 1×10^{-4} for the phase projector, a learning rate of 1×10^{-3} for the feature bank, $\lambda_{render} = 1.0$, and $\lambda_{phase} = 0.1$. The backbone-specific configurations are as follows. FreeTimeGS is trained for 500,000 iterations with an upsampling factor $k = 2$ and $\lambda_{mono} = 0.1$, while RealTime-4DGS is trained for 60,000 iterations with $k = 3$ and $\lambda_{mono} = 0.01$. Compared with the FreeTimeGS backbone alone, Phase-DGS requires approximately 30% more GPU memory and $2.1\times$ longer training time per iteration, while the rendering cost at inference remains the same (see Section A).

Datasets. We evaluated Phase-DGS on two representative datasets for dynamic scene reconstruction, each featuring distinct camera configurations to ensure a comprehensive assessment. The Nvidia Dynamic Scene Dataset [41] consists of 12 front-facing Full HD cameras, and SelfCap [38] provides 24 convergent 4K cameras, with sequences of approximately 100 and 300 frames, respectively. For the Nvidia Dynamic Scene Dataset, we use three scenes: Jumping, Playground, and Umbrella. For SelfCap, we use Corgi, Yoga, and Bike. For both datasets, all images are downsampled to 960×540. To validate robustness under various unsynchronized conditions, we evaluated on three data configurations. The **Synced** condition uses the provided dataset as is and assumes synchronized cameras. The **Shifted** condition introduces inter-camera temporal offsets of about 20–30 frames by randomly shifting the starting frame of each camera. The **Dropped** condition simulates severe real-world asynchrony with 10% random frame drops and 5–10 frames freeze, which vary independently across cameras.

Train and Test View Selection. Since Phase-DGS improves the optimization quality of the radiance field, we select a single camera located at the outermost spatial position in each scene as the test view. This view has the largest viewpoint difference from the train views, where artifacts such as spatial incoherence caused by inaccurate timestamps appear more prominently.

4.2 Baselines

We compare Phase-DGS to the following baselines. **4DGS w/o Sync** trains the 4DGS backbone directly using absolute timestamps without any synchronization and serves as a lower-bound reference. **VisualSync** [16] performs video synchronization as a preprocessing step, after which the 4DGS backbone is trained on the aligned sequences. **Sync-NeRF** [12] jointly optimizes temporal offsets together with neural radiance field reconstruction in an end-to-end manner.

4.3 Quantitative Results

As shown in Tab. 1, the Synced condition achieves the highest performance in all datasets (all experimental results are shown in Tab. A4 and A5, and see evaluation detail in Section B). In contrast, the Shifted and Dropped conditions exhibit different tendencies depending on the dataset. A noteworthy observation is that RealTime-4DGS [35], which implicitly represents motion, is relatively more robust to unsynchronized conditions than FreeTimeGS [34], which explicitly represents motion. Due to this property, the proposed Phase-DGS shows only a marginal improvement in PSNR when combined with the RealTime-4DGS backbone. In contrast, when combined with the FreeTimeGS backbone, which uses an explicit motion representation, Phase-DGS achieves a substantial improvement in quantitative metrics. FreeTimeGS is therefore particularly sensitive to timestamp inaccuracy, and under such conditions the phase-based supervision of Phase-DGS achieves the greatest effect.

Table 1: Average quantitative comparison on the **SelfCap** and **Nvidia Dynamic Scene** datasets under synchronized (Synced) and unsynchronized (Shifted, Dropped) conditions. ↑: higher is better, ↓: lower is better. Green: best among methods sharing the same backbone and condition. **Red**: best across all methods under the same condition.

			SelfCap			Nvidia		
Condition	**Backbone**	**Method**	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Synced	K-Planes	Sync-NeRF	-	-	-	28.889	0.942	0.048
	RealTime-4DGS	-	30.858	0.970	0.030	29.735	0.952	0.045
		Ours	32.145	0.973	0.026	28.938	0.946	0.061
	FreeTimeGS	-	34.181	0.984	0.010	31.167	0.963	0.022
		VisualSync	33.446	0.983	0.013	30.429	0.958	0.025
		Ours	34.727	0.985	0.012	30.416	0.959	0.030
Shifted	K-Planes	Sync-NeRF	-	-	-	28.492	0.940	0.049
	RealTime-4DGS	-	27.614	0.956	0.039	23.083	0.915	0.084
		Ours	28.288	0.961	0.037	26.655	0.937	0.064
	FreeTimeGS	-	27.170	0.955	0.026	23.785	0.918	0.068
		VisualSync	29.611	0.967	0.018	27.958	0.950	0.033
		Ours	32.947	0.981	0.014	29.836	0.956	0.037
Dropped	K-Planes	Sync-NeRF	-	-	-	26.683	0.929	0.064
	RealTime-4DGS	-	26.679	0.959	0.036	25.547	0.931	0.068
		Ours	29.638	0.966	0.032	28.263	0.944	0.056
	FreeTimeGS	-	27.894	0.962	0.025	25.313	0.929	0.055
		VisualSync	29.739	0.972	0.019	28.057	0.943	0.039
		Ours	34.379	0.984	0.012	30.665	0.961	0.030

Since Sync-NeRF [12] is retrained under capture conditions that differ from those in the original paper, both the COLMAP-based pose initialization and hyperparameter settings additionally affect the reconstruction quality under all conditions. In particular, the SelfCap dataset [38] proves incompatible with the default configuration of Sync-NeRF, which was originally designed for forward-facing captures such as the Nvidia Dynamic Scene dataset [41], resulting in notably degraded reconstruction quality. Furthermore, the Dropped condition falls outside the design scope of Sync-NeRF, which assumes only real-valued sub-frame temporal offsets. Consequently, Sync-NeRF also yields poor performance under the Dropped condition.

VisualSync [16] successfully corrects synchronization errors before training in the Shifted conditions, yielding reconstruction quality comparable to that of the backbone under the Synced condition. However, the Dropped condition disrupts the temporal consistency of feature correspondences, preventing VisualSync from reliably estimating epipolar geometry. In particular, even under the Synced condition, VisualSync introduces an unexpected frame offset of approximately 3% of the total frame count after synchronization, which decreases the reconstruction quality.

Under the Synced condition, Phase-DGS improves the average PSNR on SelfCap by 0.55 dB and 1.29 dB with the FreeTimeGS and RealTime-4DGS backbones, respectively, while it shows a slightly lower PSNR on the Nvidia Dynamic Scene dataset. In SelfCap, the convergent multi-camera setup introduces greater viewpoint diversity, making spatiotemporal coherence a more critical factor for reconstruction quality. As a result, accurate synchronization leads to a more pronounced improvement in rendering quality. In contrast, in the Nvidia Dynamic Scene dataset, the simpler forward-facing configuration allows the backbone to converge quickly in terms of image quality, whereas Phase-DGS requires additional iterations for phase learning within the same training budget, resulting in a slightly lower PSNR than the backbone under the Synced condition. We note that Phase-DGS is designed to provide robustness to synchronization errors rather than to improve reconstruction under perfect synchronization. Since the synchronization status of real-world captures is often unknown, it is noteworthy that Phase-DGS remains within 0.8 dB of the backbone under the Synced condition, while improving the PSNR by up to 6.5 dB under the Dropped condition (SelfCap, FreeTimeGS).

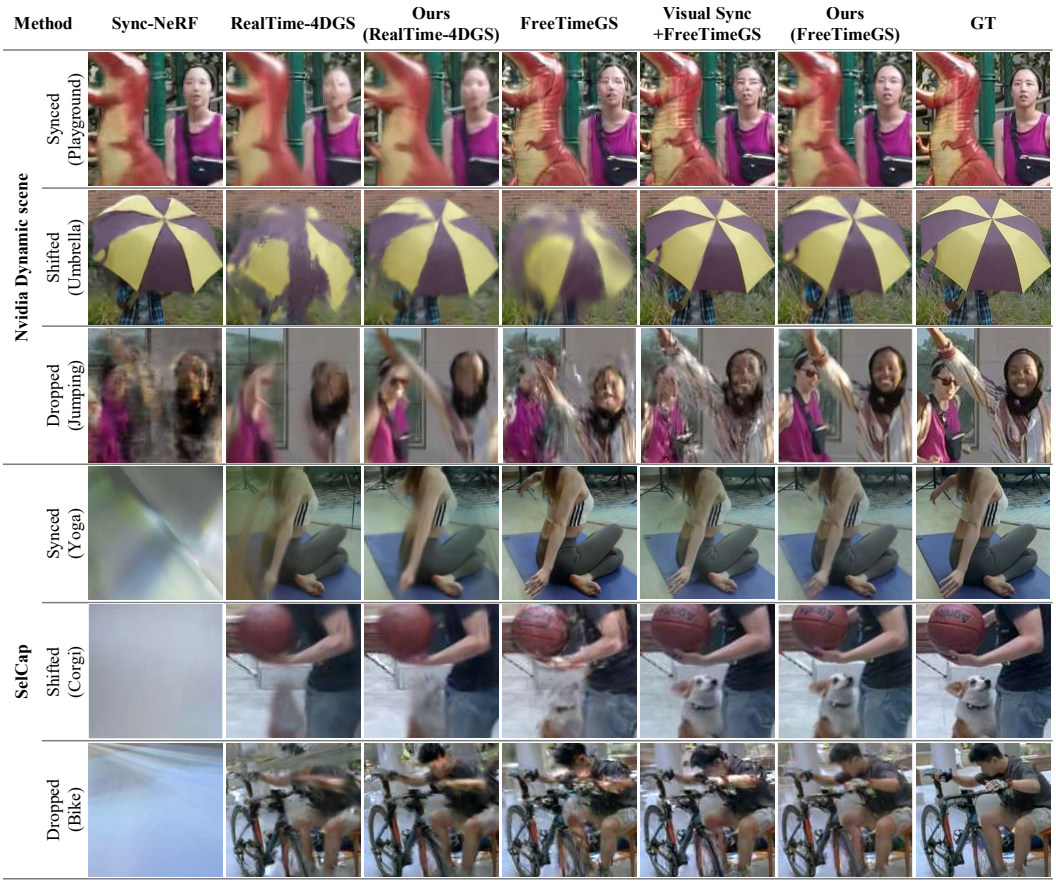


Figure 3: Qualitative comparison under the synchronized condition (Synced) and the unsynchronized condition (Shifted and Dropped) on **Nvidia Dynamic Scene dataset** and **SelfCap datasets**. Phase-DGS consistently produces cleaner renderings while baselines exhibit ghosting and geometric distortion.

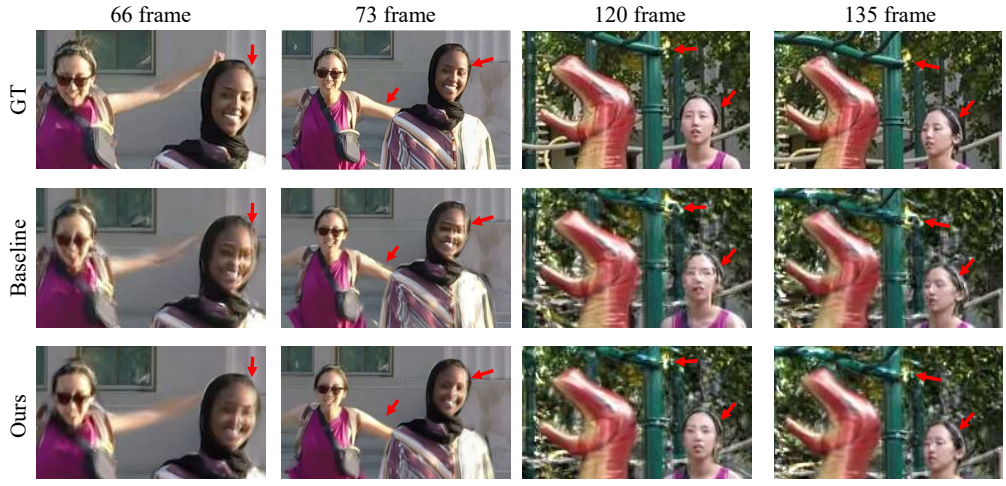


Figure 4: Qualitative comparison with enlarged views under the Synced condition. While baseline models exhibit ghosting and geometric distortion, Phase-DGS consistently reduces artifacts in rendered images. $\rightarrow$: visual artifact.

4.4 Qualitative Results

As shown in Fig. 3, Phase-DGS with the FreeTimeGS [34] backbone achieves the best visual quality among all the methods compared. This qualitative comparison reveals perceptual differences that quantitative metrics alone cannot fully capture. For the RealTime-4DGS [35] backbone, as an additional discussion , the difference in PSNR before and after applying Phase-DGS is small. However, the rendering results show that the Gaussians in dynamic regions are accurately synchronized and converge to a single spatial point, and as a result the shape of dynamic objects appears much sharper than in the baseline. Since the phase estimated by the phase projector is gradually reflected throughout the training process, the overall PSNR of the scene shows only a limited change, while the Gaussian alignment in dynamic regions steadily improves. These results demonstrate that Phase-DGS effectively learns accurate phase information for each frame, precisely defining the three-dimensional position of each Gaussian and enabling cleaner renderings across both backbones.

Fig. 4 presents enlarged views for a more detailed comparison, where all outputs are rendered from models trained under the Synced condition. Comparing Phase-DGS against the FreeTimeGS backbone, we show pairs of frames from the same scene to highlight inter-frame differences. Although Phase-DGS under the Synced condition yields a lower PSNR than the FreeTimeGS baseline in the Jumping scene, the rendered images reveal noticeably fewer visual artifacts, a trend that is consistently observed in the Playground scene as well. We attribute this to the fact that Phase-DGS learns a radiance field with higher spatiotemporal coherence, enabling more geometrically consistent reconstructions even when quantitative metrics do not fully reflect this improvement. A notable observation is that even on datasets provided as synchronized, the backbone models without Phase-DGS still exhibit subtle rendering artifacts, empirically supporting the claim that even sub-frame-level timing disparities can measurably affect reconstruction quality.

4.5 Ablation Study

We conduct ablation studies on the Jumping scene of the Nvidia Dynamic Scene Dataset [41] and the Bike scene of SelfCap [38] under the Dropped condition to analyze the contribution of each component in Phase-DGS with FreeTimeGS [34] backbone.

Monotonic Loss. Without this loss, the phase is estimated in a temporally disordered manner, causing the beginning and end of the sequence to be assigned phases without consistent ordering. (see Section 3.4)

Feature Bank. When the feature bank is initialized but not updated during training, the stored features remain fixed and the accuracy of phase estimation drops substantially (see Section 3.3). This shows that the learnable feature bank plays an essential role by providing a sufficient reference over the entire sequence of each camera, which stabilizes the phase estimation process.

Upsampling Factor k. We vary the upsampling factor k of the feature bank and analyze its effect on performance (see Section 3.3). When $k = 1$, the feature bank operates at the original frame rate and cannot capture sub-frame asynchrony. At $k = 2$, the finer temporal resolution yields the best trade-off between phase accuracy and computational cost. Increasing k to 4 does not improve performance, as the enlarged feature bank requires more training iterations to converge within the fixed training budget, leading to a slight PSNR degradation.

ViT Backbone. We compare DINOv2-S, -B, and -L [23] depending on the number of parameters of the ViT encoder. The self-supervised DINOv2 learns detailed visual representations, and the performance gap between different sizes of DINOv2 models is small. This suggests that phase estimation captures the overall progression of motion, for which the global semantic representation learned by the base model is already sufficient.

ROI Crop. We compare two input configurations. The first crops the input image to the motion-intensive ROI, while the second forcibly resizes the full image to 224×224 for the visual transformer (see Sections 3.2 and 3.6) [23]. The ROI-based configuration achieves slightly better performance, but the difference is less than 0.2 dB in both scenes, which shows that Phase-DGS also operates on full images without the ROI crop. The ROI crop is expected to be more effective when the dynamic objects occupy only a small region of the image, since resizing the full image to 224×224 can remove fine motion cues.

Table 2: Ablation study on the *Jumping* scene **(Nvidia Dynamic Scene dataset)** and *Bike* scene **(SelfCap dataset)** under the Dropped condition. FreeTimeGS is used as the DGS backbone.

	Jumping			Bike		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Ours (default)	29.386	0.942	0.037	31.212	0.977	0.016
w/o Monotonic Loss	24.421	0.887	0.073	25.903	0.944	0.039
w/o Bank Learning	24.794	0.897	0.070	27.642	0.958	0.026
$k = 1$	29.425	0.942	0.038	31.130	0.977	0.015
$k = 4$	29.006	0.938	0.038	31.101	0.976	0.015
DINOv2-S	29.034	0.939	0.039	30.932	0.976	0.016
DINOv2-L	29.648	0.943	0.037	31.411	0.978	0.015
w/o ROI Crop	29.231	0.939	0.039	31.16	0.977	0.017

5 Conclusion

We presented Phase-DGS, a self-supervised framework for robust dynamic scene reconstruction under unsynchronized multi-camera conditions. By replacing absolute timestamps with per-frame phases extracted directly from images, Phase-DGS eliminates the reliance on hardware synchronization and enables robust 4DGS training even under severe frame drops and temporal offsets. Experiments on Nvidia Dynamic Scene Dataset and SelfCap demonstrate that Phase-DGS consistently outperforms existing synchronization baselines under unsynchronized conditions, achieving competitive performance even under the synchronized setting. Furthermore, plug-and-play compatibility with multiple 4DGS backbones highlights the generalizability of our approach. The ability to train without precise timestamps suggests the potential to extend Phase-DGS to archival or previously captured footage, where hardware synchronization cannot be applied retroactively. We hope Phase-DGS serves as a practical and scalable solution for real-world multi-camera capture systems.

Limitations and Future Work. The reconstruction quality of Phase-DGS depends on the underlying 4DGS backbone, and the way timestamps are processed inside the network also affects performance. In addition, Phase-DGS currently struggles with scenes containing periodic motions, such as the *Balloon1* sequence, where visually similar frames make accurate phase assignment difficult. We plan to address these issues through backbone-agnostic training strategies and inter-frame constraints that encourage smooth and consistent phase transitions.

Acknowledgements

This work was supported by the IITP grant funded by the Korea government (MSIT): Development of immersive video spatial computing technology for ultra-realistic metaverse services (No.2022-0-00022, RS-2022-II220022), and by the IITP grant (RS-2021-II211343, AI Graduate School Program (Seoul National University)), funded by the Korean government (MSIT).

Disclosures

The authors declare no competing interests.

References

[1] Sameer Ansari, Neal Wadhwa, Rahul Garg, and Jiawen Chen. Wireless software synchronization of multiple distributed cameras. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9. IEEE, 2019.

[2] Nicholas J Bryan, Paris Smaragdis, and Gautham J Mysore. Clustering and synchronizing multi-camera video via landmark cross-correlation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2389–2392. IEEE, 2012.

[3] Changwoon Choi, Jeongjun Kim, Geonho Cha, Minkwan Kim, Dongyoon Wee, and Young Min Kim. Humans as a calibration pattern: Dynamic 3d scene reconstruction from unsynchro-

nized and uncalibrated videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6598–6608, 2025.

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

[5] Ahmed Elhayek, Carsten Stoll, Kwang In Kim, H-P Seidel, and Christian Theobalt. Feature-based multi-video synchronization with subframe accuracy. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 266–275. Springer, 2012.

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

[7] Huayu Huang, Yang Shang, Banglei Guan, and Qifeng Yu. 3d trajectory reconstruction of moving points based on asynchronous cameras. *Acta Mechanica Sinica*, 41(7):425322, 2025.

[8] Vladimir Iashin, Weidi Xie, Esa Rahtu, and Andrew Zisserman. Synchformer: Efficient synchronization from sparse cues. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5325–5329. IEEE, 2024.

[9] IEEE. IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, 2019.

[10] Rishit Javia, Falak Shah, and Shivam Dave. Posesync: Robust pose based video synchronization. *arXiv preprint arXiv:2308.12600*, 2023.

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

[12] 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.

[13] Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. Wildgaussians: 3d gaussian splatting in the wild. *arXiv preprint arXiv:2407.08447*, 2024.

[14] Yonghan Lee, Tsung-Wei Huang, Shiv Gehlot, Jaehoon Choi, Guan-Ming Su, and Dinesh Manocha. Synctrack4d: Cross-video motion alignment and video synchronization for multi-video 4d gaussian splatting. *arXiv preprint arXiv:2512.04315*, 2025.

[15] 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.

[16] Shaowei Liu, David Yifan Yao, Saurabh Gupta, and Shenlong Wang. Visual sync: Multi-camera synchronization via cross-view object motion. *arXiv preprint arXiv:2512.02017*, 2025.

[17] Cheng Lu and Mrinal Mandal. A robust technique for motion-based video sequences temporal alignment. *IEEE Transactions on Multimedia*, 15(1):70–82, 2012.

[18] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20654–20664, 2024.

[19] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7210–7219, 2021.

[20] Jaro Meyer, Frédéric Giraud, Joschua Wüthrich, Marc Pollefeys, Philipp Fürnstahl, and Lilian Calvet. Rocsync: Millisecond-accurate temporal synchronization for heterogeneous camera systems. *Sensors*, 26(3):1036, 2026.

[21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4): 1–15, 2022.

[22] Avihai Naaman, Ron Shapira Weber, and Oren Freifeld. Synchronization of multiple videos. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12514–12523, 2025.

[23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[24] 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.

[25] Dmitry Pundik and Yael Moses. Video synchronization using temporal signals from epipolar lines. In *European conference on computer vision*, pages 15–28. Springer, 2010.

[26] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[27] RED.COM, LLC. Genlock. `https://docs.red.com/955-0156/EPICSCARLETOperationGuide/en-us/Content/8_TC_Gen_Multi/Genlock.htm`, 2019. EPIC/SCARLET Operation Guide, 955-0156_V6.3, REV-N.

[28] Prarthana Shrestha, Mauro Barbieri, Hans Weda, and Dragan Sekulovski. Synchronization of multiple camera videos using audio-visual features. *IEEE Transactions on Multimedia*, 12(1): 79–92, 2009.

[29] Prarthana Shrstha, Mauro Barbieri, and Hans Weda. Synchronization of multi-camera video recordings based on audio. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 545–548, 2007.

[30] Society of Motion Picture and Television Engineers. SMPTE ST 274:2008 – Television – 1920 × 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates. SMPTE Standard, jan 2008. Revision of SMPTE 274M-2005.

[31] Society of Motion Picture and Television Engineers. ST 296:2011 – SMPTE Standard – 1280 × 720 Progressive Image Sample Structure — Analog and Digital Representation and Analog Interface. SMPTE Standard, oct 2011. Revision of SMPTE 296M-2001.

[32] Minh Vo, Srinivasa G Narasimhan, and Yaser Sheikh. Spatiotemporal bundle adjustment for dynamic 3d reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1710–1718, 2016.

[33] Oliver Wang, Christopher Schroers, Henning Zimmer, Markus Gross, and Alexander Sorkine-Hornung. Videosnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014.

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

[35] 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.

[36] Xinyi Wu, Zhenyao Wu, Yujun Zhang, Lili Ju, and Song Wang. Multi-video temporal synchronization by matching pose features of shared moving subjects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[37] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5438–5448, 2022.

[38] 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.

[39] Zhixin Xu, Hengyu Zhou, Yuan Liu, Wenhan Xue, Hao Pan, Wenping Wang, and Bin Wang. Dynamic gaussian scene reconstruction from unsynchronized videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 11469–11477, 2026.

[40] Liqiang Yin, Ruize Han, Wei Feng, and Song Wang. Self-supervised human pose based multi-camera video synchronization. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1739–1748, 2022.

[41] 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.

[42] 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.



Figure A1: Multi-camera frames at the same frame index from the *Jumping* sequence (Nvidia Dynamic Scene Dataset) under the **Dropped** condition. Despite sharing the same frame index, each camera captures a distinct motion state, illustrating the severity of temporal misalignment introduced by frame drops and freezes.

A Additional Implementation Details

Data Generation. To assess robustness under various real-world capture conditions, we construct three data configurations with increasing levels of temporal asynchrony. The **Synced** condition uses the original dataset without modification, serving as the upper-bound reference under ideal synchronization. The **Shifted** condition introduces inter-camera temporal offsets of 20–30 frames by randomly shifting the starting frame of each camera, simulating clock drift or delayed recording onset. The **Dropped** condition simulates severe asynchrony by applying 10% random frame drops and 5–10 frame freezes that vary independently across cameras, reflecting realistic packet loss and buffer failures in large-scale capture systems. As illustrated in Fig. A1, frames captured at the same frame index across cameras under the **Dropped** condition exhibit pronounced motion discrepancies, highlighting the severity of temporal misalignment introduced by our simulation. Additional visualizations are provided in the supplementary video.

ROI Cropping. Since this work focuses on the synchronization problem, we implement a manual ROI extraction tool throughout the experiments with the help of an LLM. The ROI extraction tool used in our experiments is implemented as a simple interface that supports manual bounding box selection as shown in Fig. A2. For each scene, the ROI is specified to cover the region where motion is most concentrated, based on a single representative frame from the scene. The tool also provides options to use the central square region of the frame or to keep the full frame without cropping. Different cameras in the same scene can have bounding boxes of different sizes and positions, and the location of each bounding box can be specified freely. Note that the ROI crop is an optional preprocessing step and is not required by Phase-DGS (see Section 4.5). The ROI extraction process can be further automated using algorithms such as optical flow, motion detection, or video segmentation models such as SAM2 [26]. For reproducibility, we will release the ROI coordinates and segmentation masks used in our experiments together with the code.

Feature bank Initialization. The feature bank is initialized with features extracted from all frames of all cameras. When the upsampling factor is $k > 1$, the feature entries at upsampled positions that do not correspond to original frames are filled by linear interpolation between neighboring features. Initializing the feature bank in this way provides meaningful feature values from the beginning of training, which shortens the convergence time.

Compute Resources. All experiments in this work were conducted on a single NVIDIA RTX 5090 GPU with Python 3.12, PyTorch 2.8.0, and CUDA 12.8. For the RealTime-4DGS [35] backbone, 100K iterations take approximately one hour. For the FreeTimeGS [34] backbone, 500K iterations take approximately 2.5 hours. The GPU memory usage depends on the scene, with the RealTime-4DGS backbone using approximately 12.4 GB and the FreeTimeGS backbone using approximately 4 GB. The full set of reported experiments covers 6 scenes (3 from the Nvidia Dynamic Scene

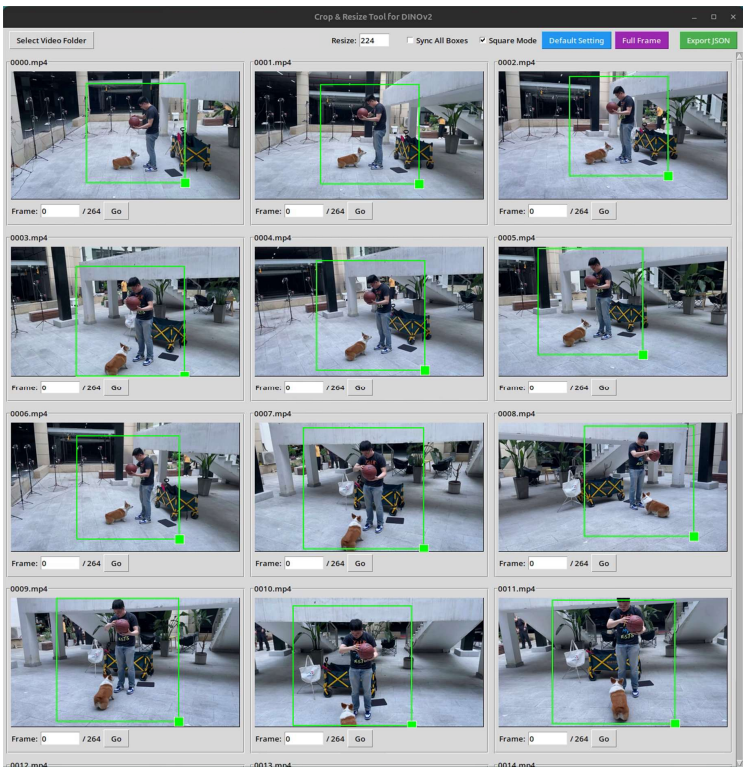


Figure A2: Interface of the ROI extraction tool used in our experiments. The tool supports manual bounding box selection, with options to use the central square region or the full frame. Different cameras in the same scene can have independent bounding boxes.

Dataset and 3 from SelfCap) across 3 synchronization conditions (Synced, Shifted, Dropped) and two 4DGS backbones (RealTime-4DGS and FreeTimeGS). The reported computation does not include preliminary experiments and failed runs.

Training Overhead. To quantify the additional cost of phase learning, we measure the GPU memory usage and training throughput with the FreeTimeGS [34] backbone on a multi-view sequence captured with 23 cameras (50 frames), as shown in Tab. A1. Note that the absolute values differ from those reported above because the dataset is different. Phase-DGS requires an additional 514 MB (+30.5%) of GPU memory, which mainly comes from the feature bank, the phase projector, and the ViT encoder. Since the ViT encoder remains frozen, back-propagation through the encoder is not required, and only the lightweight phase projector and the feature bank are additionally optimized. The training throughput decreases from 148 to 70 iterations per second (2.11× slower). This overhead applies only to training, since the ViT encoder, the phase projector, and the feature bank are not used at inference, where the backbone is rendered directly at a given phase.

Table A1: Training overhead of Phase-DGS with the FreeTimeGS backbone on a 23-camera, 50-frame sequence, measured on a single NVIDIA RTX 5090 GPU.

Configuration	GPU Memory	Training Throughput
FreeTimeGS (backbone only)	1686 MB	148 it/s
Phase-DGS (FreeTimeGS)	2200 MB (+514 MB)	70 it/s (2.11× slower)

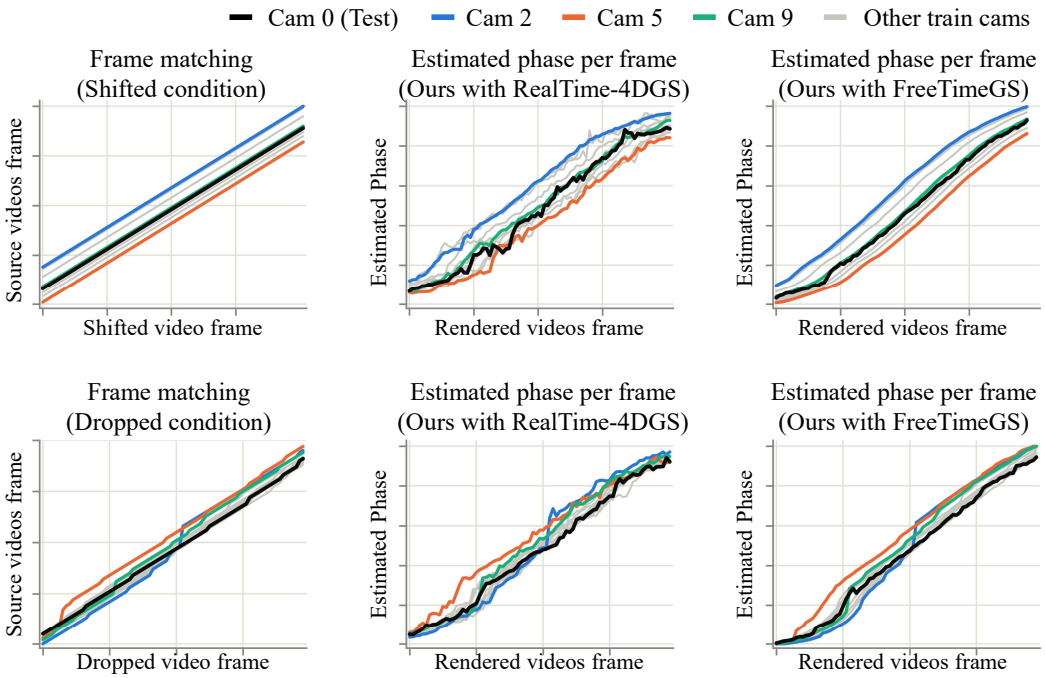


Figure A3: Phase estimation results on the Jumping scene under the Shifted (top) and Dropped (bottom) conditions. (Left) Ground-truth frame correspondences between each camera and the source video. (Center, Right) Per-frame phase estimates produced by Phase-DGS with the RealTime-4DGS and FreeTimeGS backbones, respectively.

B Evaluation Protocol

Phase Reference Computation. Phase-DGS does not estimate an absolute timestamp from an image but instead extracts a relative phase value. As shown in Fig. A3, the phase values produced by Phase-DGS for frames within a single scene follow a non-linear curve. It is noted that the phase curves of the train views are regularized by the monotonic loss so that they remain monotonic, while the test view (Cam 0) is not subject to this loss and thus exhibits a slightly different phase curve, although it follows a similar monotonic trend. As a result, the phase stamp of the Phase-DGS, which refers to the ordering of frames along the phase axis and plays a role similar to a time stamp, does not provide absolute temporal information. To quantitatively evaluate Phase-DGS, we therefore need to define a reference phase stamp for each scene. We should construct the reference phase stamp for the test view. The unsynchronized dataset used in our experiments provides the correspondence between each view's frame index and the index in the original source video. Using this correspondence, we can identify which frame of the source video each test view frame is sampled from, and then find the train view frames that are sampled from the same source index. The phase reference for each test view frame is obtained by averaging the phases of the corresponding train view frames. The overall procedure is summarized in Algorithm 1. Since this protocol does not recover the exact absolute phase stamp but only estimates a reference phase in an indirect manner, the actual time of a test view frame may not perfectly align with the time implied by the estimated reference phase. This misalignment can introduce minor inaccuracies in the reported quantitative metrics. We note that this protocol is used only for evaluation purposes, and the exact procedure for constructing the phase-line may vary depending on the application or the dataset. However, this correspondence is a form of ground-truth temporal alignment that is available only in our simulated datasets, and it is not used for the baselines. The reported metrics therefore reflect the reconstruction quality of Phase-DGS given an accurate temporal anchor, which is stronger than the anchor available in practice, where the global phase stamp is defined without ground truth (see Section C).

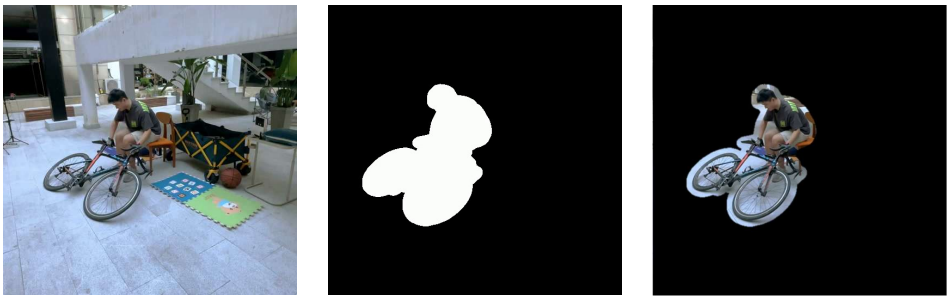
Evaluation. To evaluate the performance of the trained GS model, we first compute the phase reference $\tilde{p}^f_{test}$ for each test view frame using Algorithm 1. Here, the reference phase $\tilde{p}^f_{test}$ from

Algorithm 1 contains estimation noise, which can lower the reported metrics even when the underlying reconstruction is accurate. To account for this, we adopt a strategy similar to the test-time optimization used in NeRF-W [19] and WildGaussians [13]. With the trained GS model fully frozen, we search within $\pm\delta$ of $\tilde{p}^f_{test}$ for the phase $\tilde{p'}^f_{test}$ that yields the highest PSNR against the ground-truth frame. Because the search window is small and phase is a single scalar, this procedure corrects for estimator noise without introducing meaningful bias toward the ground truth. In the experiments, we set $\delta = 0.05$, which corresponds to a window size of approximately 10 to 30 frames relative to the ground truth. The final metrics are computed from images rendered at these refined phases. To further assess the effect of synchronization on rendering quality, we compute metrics exclusively over dynamic regions. As shown in Fig. A4, segmentation masks are extracted from the ground-truth frames, and PSNR, SSIM, and LPIPS are evaluated only within these masked areas. Since dynamic regions are most sensitive to temporal misalignment, restricting the evaluation to these areas allows for a more precise examination of whether the phase reference is accurately synchronized with the ground truth.

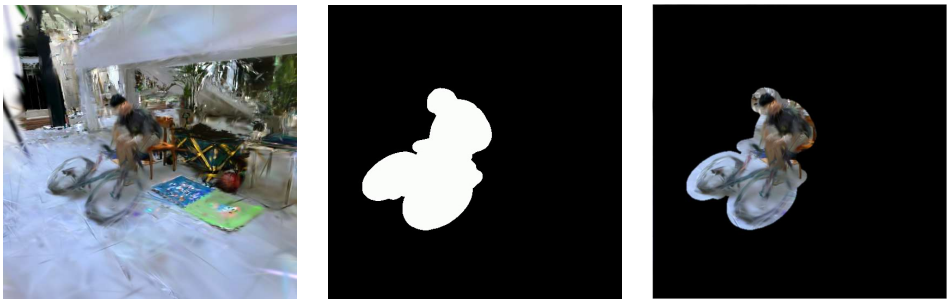
Search Window Sensitivity. To verify that the test-time phase search only compensates for the estimation noise, we vary the search window W and report the PSNR with the FreeTimeGS [34] backbone in Tab. A2. $W{=}0$ denotes rendering directly at the reference phase $\tilde{p}^f_{\text{test}}$ without any search, and $W{=}10$ is the default setting used for all reported results. Increasing the window from 10 to 50 does not change the PSNR in any setting, which indicates that the search converges within a small neighborhood of the reference phase. Without the search ($W{=}0$), the PSNR changes by less than 0.2 dB in five of the six settings, and the largest drop of 0.75 dB is observed in the Shifted condition of the Jumping scene. Even at $W{=}0$, Phase-DGS outperforms the FreeTimeGS baseline by 4.8–7.0 dB under all Shifted and Dropped settings (see Tab. A4 and A5), which shows that the performance gains do not result from the test-time search.

Table A2: Sensitivity of PSNR to the search window W of the test-time phase search with the FreeTimeGS backbone. $W{=}0$ denotes rendering at the reference phase without search, and $W{=}10$ is the default setting.

Condition	Scene	$W{=}0$	$W{=}5$	$W{=}10$ (default)	$W{=}50$
Synced	Bike	31.355	31.376	31.376	31.376
	Jumping	28.851	28.851	28.851	28.851
Shifted	Bike	31.110	31.299	31.299	31.299
	Jumping	27.354	28.090	28.103	28.103
Dropped	Bike	31.131	31.217	31.217	31.217
	Jumping	29.386	29.386	29.386	29.386



GT image | Mask based on GT | Masked GT



Rendered image (RealTime-4DGS) | Mask based on GT | Masked render image

Figure A4: To assess rendering quality in dynamic regions, we generate segmentation masks from ground-truth images and compute metrics exclusively over the masked areas. The mask is generated by object segmentation using SAM2 [26], and a small padding margin is applied to the mask boundaries to assess the reconstruction quality of regions surrounding dynamic objects.

Algorithm 1 Phase Reference Computation for Test Views

Require: Train view images $\{u_i^f\}$ with source indices $\{s_i(f)\}$, test view source indices $\{s_{\text{test}}(f)\}$, Trained Phase projector Φ

Ensure: Reference phase $\tilde{p}_{\text{test}}^f$ for each test view frame

for each test view frame index $f = 1, \ldots, F$ **do**
  $\mathcal{T} \leftarrow \emptyset$ ▷ Collect matching train view frames
  for each train view $i = 1, \ldots, N$ **do**
    for each frame index $f' = 1, \ldots, F$ **do**
      if $s_i(f') = s_{\text{test}}(f)$ **then**
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{(i, f')\}$
      end if
    end for
  end for
  $p_{\text{sum}} \leftarrow 0$
  for each $(i, f') \in \mathcal{T}$ **do**
    $p_{\text{sum}} \leftarrow p_{\text{sum}} + \Phi(u_i^{f'})$ ▷ Extract phase from each matching train frame
  end for
  $\tilde{p}_{\text{test}}^f \leftarrow p_{\text{sum}} / |\mathcal{T}|$
end for
return $\{\tilde{p}_{\text{test}}^f\}_{f=1}^F$

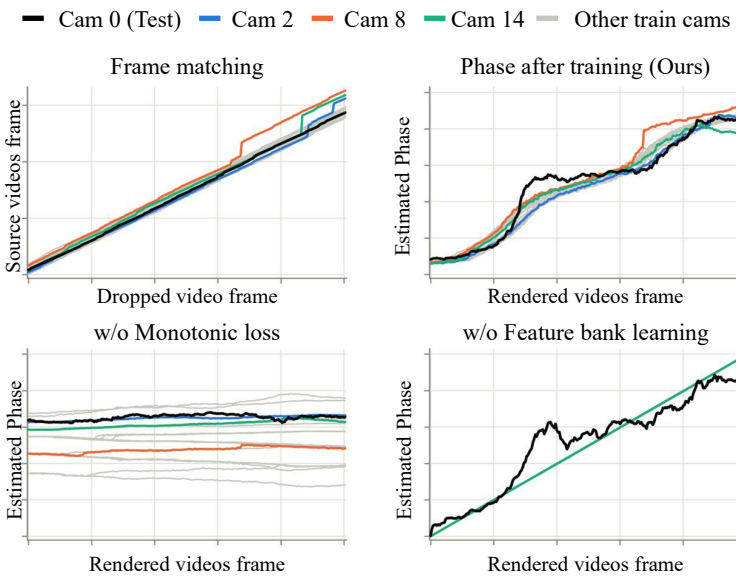


Figure A5: Per-frame phase estimates on the Bike scene (SelfCap) under the Dropped condition. (Top left) Ground-truth frame correspondences. Phase estimation results under (Top right) default setting, (Bottom left) without monotonic loss. (Bottom right) without feature bank learning.

C Additional Discussion

Backbone-Dependent Reconstruction Quality. As briefly noted in Section 5, the performance of Phase-DGS is influenced by the visual representation capacity of the underlying 4DGS backbone. We attribute this to the relatively large weight assigned to the rendering loss during training, which makes the rendering quality of the backbone an important factor for stable phase learning. We provide a more detailed observation here. The RealTime-4DGS [35] backbone began to overfit to the training scene after approximately 60,000 iterations, where the test-view PSNR dropped sharply and, at the same time, the consistency of the estimated phase values also deteriorated. In contrast, the FreeTimeGS [34] backbone maintained stable visual quality up to 500,000 iterations, and the corresponding phase estimates remained close to the ground-truth phase profile throughout training.

Global phase estimation in Practice. The phase projector module of Phase-DGS estimates the phase independently for each frame of each camera view. As shown in our experiments, it is not possible to define an accurate global time stamp under the Dropped condition. Since Phase-DGS can only extract relative phase stamps, the global phase stamp must be defined externally. We propose two approaches for this. The first uses the average of the phase stamps estimated across all camera views as the global phase stamp. The second uses the phase estimated from a single reference camera view as the global phase stamp.

D Ablation Detail

As shown in the ablation study, the learnable feature bank and Monotonic Loss are the two most critical components for training Phase-DGS. Removing either of them prevents accurate phase estimation, making the Gaussian Splatting training infeasible. Figure A5 shows the per-frame predicted phase of each camera view, produced by the fully trained phase projector module. In the default setting, the relative phase values across frames follow the ground truth labels. Without Monotonic Loss, phase differences across views appear, but the phase within a single view barely changes, making training collapse. Without feature bank learning, all train views produce the same linear phase output, which means the phase projector module provides no useful information.

E Numerical Stability

To verify the stability of Phase-DGS across different runs, we train the model four times from scratch, including COLMAP initialization, on the *Corgi* scene under the Dropped condition. As reported in Tab. A3, Phase-DGS achieves consistent performance across all runs, with a PSNR standard deviation of 0.466 dB, demonstrating reliable convergence regardless of initialization.

Table A3: Numerical stability of Phase-DGS across four independent runs on the *Corgi* scene **(SelfCap dataset)** under the Dropped condition.

Run	PSNR↑	SSIM↑	LPIPS↓
1	35.81	0.989	0.008
2	36.86	0.990	0.007
3	36.04	0.990	0.008
4	35.99	0.990	0.008
Mean	36.175	0.990	0.008
Std	0.466489	0.000450	0.000455

F Failure Case

Highly Periodic Content. Since Phase-DGS estimates the phase from visual patterns extracted from image content, it has a limitation when a scene contains highly repetitive patterns along the temporal axis. We observed this behavior in the Balloon scene of the Nvidia Dynamic Scene Dataset [41]. The scene consists of a person repeatedly moving a balloon up and down against a simple background, and the visual patterns of frames at different times remain nearly identical. As a result, the phase projector assigns similar phase values to frames that are temporally distant, which provides ambiguous supervision to the 4DGS training and prevents accurate reconstruction. Fig. A6 shows the estimated phase values and the rendered results on the Balloon scene. This limitation arises from the fact that Phase-DGS uses only the visual pattern of a single frame as input, and extending the phase estimation module to incorporate temporal causality across frames (e.g., frame indices or global frame-to-frame similarity) or cross-camera information is expected to mitigate this ambiguity.

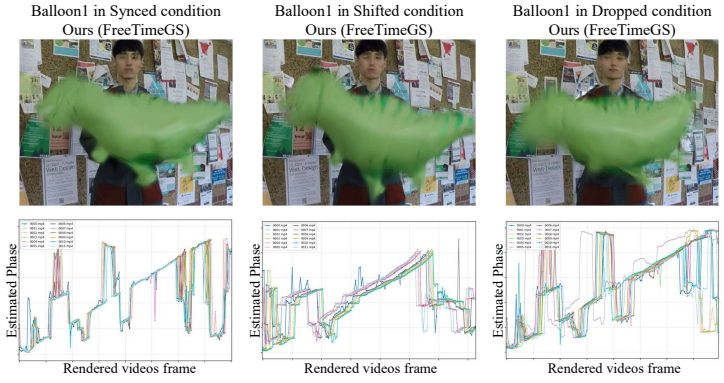


Figure A6: Failure Case of Phase-DGS on Balloon1, a Highly Periodic Scene.

G Per-scene quantitative results

Tables A4 and A5 provide per-scene results corresponding to the averages reported in Tab. 1.

Table A4: Quantitative comparison on the **SelfCap dataset** under synchronized (Synced) and unsynchronized (Shifted, Dropped) conditions. ↑: higher is better, ↓: lower is better. Green : best among methods sharing the same backbone and condition. **Red**: best across all methods under the same condition.

			Bike			Corgi			Yoga		
Condition	**Backbone**	**Method**	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Synced	K-Planes	Sync-NeRF	-	-	-	-	-	-	-	-	-
	RealTime-4DGS	-	26.430	0.950	0.040	35.552	0.985	0.013	30.593	0.974	0.036
		Ours	28.565	0.960	0.032	34.907	0.982	0.019	32.963	0.978	0.028
	FreeTimeGS	-	30.888	0.977	0.014	35.677	0.990	0.006	35.977	0.987	0.011
		VisualSync	30.490	0.974	0.016	36.646	0.990	0.006	33.203	0.984	0.016
		Ours	31.376	0.978	0.015	37.681	0.991	0.007	35.124	0.987	0.013
Shifted	K-Planes	Sync-NeRF	-	-	-	-	-	-	-	-	-
	RealTime-4DGS	-	24.594	0.932	0.049	29.100	0.971	0.025	29.149	0.966	0.043
		Ours	26.753	0.948	0.036	30.812	0.974	0.021	27.298	0.961	0.054
	FreeTimeGS	-	24.139	0.935	0.035	27.678	0.965	0.018	29.693	0.965	0.026
		VisualSync	26.227	0.949	0.024	29.975	0.974	0.012	32.632	0.978	0.018
		Ours	31.299	0.977	0.016	34.767	0.988	0.010	32.775	0.979	0.017
Dropped	K-Planes	Sync-NeRF	-	-	-	-	-	-	-	-	-
	RealTime-4DGS	-	26.339	0.946	0.037	27.149	0.967	0.025	26.548	0.963	0.045
		Ours	28.608	0.961	0.029	33.664	0.981	0.016	26.642	0.956	0.052
	FreeTimeGS	-	26.304	0.947	0.027	26.505	0.964	0.020	30.872	0.976	0.028
		VisualSync	29.746	0.970	0.019	27.491	0.968	0.017	31.981	0.978	0.021
		Ours	31.217	0.977	0.016	36.859	0.990	0.007	35.061	0.985	0.014

Table A5: Quantitative comparison on the **Nvidia Dynamic Scene dataset** under synchronized (Synced) and unsynchronized (Shifted, Dropped) conditions. ↑: higher is better, ↓: lower is better. Green : best among methods sharing the same backbone and condition. **Red**: best across all methods under the same condition.

			Umbrella			Jumping			Playground		
Condition	**Backbone**	**Method**	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Synced	K-Planes	Sync-NeRF	31.467	0.968	0.028	24.228	0.889	0.088	30.972	0.969	0.027
	RealTime-4DGS	-	32.494	0.971	0.026	26.891	0.917	0.073	29.820	0.967	0.035
		Ours	30.328	0.961	0.049	26.724	0.912	0.093	29.763	0.965	0.040
	FreeTimeGS	-	33.839	0.978	0.019	29.759	0.942	0.032	29.904	0.969	0.016
		VisualSync	33.998	0.978	0.019	27.904	0.929	0.039	29.384	0.967	0.018
		Ours	32.549	0.972	0.032	28.851	0.936	0.038	29.849	0.969	0.021
Shifted	K-Planes	Sync-NeRF	30.947	0.967	0.030	25.088	0.891	0.087	29.442	0.963	0.031
	RealTime-4DGS	-	24.432	0.942	0.067	19.365	0.852	0.135	25.452	0.950	0.050
		Ours	29.215	0.958	0.047	25.124	0.901	0.097	25.627	0.951	0.047
	FreeTimeGS	-	24.169	0.945	0.068	20.985	0.860	0.100	26.202	0.950	0.036
		VisualSync	26.042	0.951	0.044	28.071	0.931	0.038	29.761	0.967	0.018
		Ours	32.316	0.971	0.035	28.103	0.931	0.049	29.090	0.965	0.026
Dropped	K-Planes	Sync-NeRF	29.708	0.955	0.053	22.175	0.873	0.104	28.167	0.960	0.035
	RealTime-4DGS	-	23.990	0.942	0.067	23.901	0.888	0.100	28.750	0.964	0.038
		Ours	29.410	0.958	0.047	26.551	0.913	0.081	28.827	0.962	0.039
	FreeTimeGS	-	23.948	0.942	0.065	23.509	0.881	0.076	28.482	0.963	0.023
		VisualSync	30.069	0.965	0.036	25.605	0.903	0.058	28.497	0.962	0.023
		Ours	32.357	0.970	0.033	29.386	0.942	0.037	30.251	0.971	0.021

H License Information

All datasets, pretrained models, and baseline implementations used in this work are used in accordance with their respective licenses and terms of use.

Datasets. We use the Nvidia Dynamic Scene Dataset [41] and SelfCap [38] for evaluation. Both datasets are publicly available for academic research and are used in accordance with the terms provided by their respective authors.

Pretrained Models. We use DINOv2 [23] as the ViT encoder, which is released under the Apache 2.0 license.

Baseline Methods. Our re-implementation of FreeTimeGS [34] is based on the original paper. We also compare with RealTime-4DGS [35], VisualSync [16], and Sync-NeRF [12]. We cite the original papers for all methods and use them in accordance with their stated conditions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and Section 1, namely dynamic scene reconstruction under asynchronous conditions through phase-based supervision and plug-and-play compatibility with multiple DGS backbones (RealTime-4DGS and FreeTimeGS), are supported by the quantitative and qualitative experiments in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of phase-DGS are discussed in Section 5 (Conclusion), where we note that the reconstruction quality of phase-DGS depends on the representational capacity of the underlying DGS backbone, and that the way the timestamp is processed inside the neural network also affects the effectiveness of phase-DGS.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include theoretical results. The equations in the Method section describe the architecture and loss functions of phase-DGS, and they are not formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The architecture and loss functions of phase-DGS are fully described in Section 3, and the training configurations and hyperparameters are provided in the Implementation Details table in Section 4. Additional implementation details, including ROI cropping, Feature Bank initialization, and compute resources, are described in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: To preserve anonymity during the review process, the code and pretrained models are not released at this time. The code will be released upon acceptance. To support reproducibility, we provide detailed descriptions of the architecture, training procedure, and hyperparameters in Section 3 (Method), Section 4 (Experiments), and the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and test details are provided in the Implementation Details section of Section 4 (Experiments), which includes the optimizer, the number of training iterations, learning rates, loss weights, and configurations for each backbone. Additional details are described in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Due to the high computational cost of each training run, we report results from four independent runs including full COLMAP initialization on the Corgi scene under the Dropped condition.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Detailed information on compute resources is provided in the Compute Resources section of the supplementary material, including the GPU type, training time per backbone, memory usage, the total scale of the reported experiments, and a note on preliminary and failed experiments that are not included in the paper. The additional training cost of Phase-DGS over the backbone, including GPU memory usage and training throughput, is also reported in the Training Overhead section of the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms with the guidelines in every respect.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work addresses a technical problem in multi-camera synchronization for dynamic scene reconstruction. The proposed method does not directly enable harmful applications, nor does it involve sensitive data, human subjects, or deployment in high-risk domains. We therefore consider the broader societal impact of this work to be minimal.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release pretrained models, image generators, or scraped datasets with high risk for misuse. phase-DGS is a training method for Dynamic Gaussian Splatting and does not introduce assets that require such safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, pretrained models, and baseline methods used in this work are listed in the License Information section of the supplementary material, with proper citations and licenses specified for each asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release any new datasets or pretrained models. In accordance with the licenses of the datasets used, we do not redistribute their processed versions. The code will be released upon acceptance, with documentation provided at that time.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects, so IRB approval is not required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs were used as a coding assistant during implementation, where the authors designed the core algorithms and architectures of phase-DGS, and LLMs assisted in translating these designs into code. The authors reviewed and refined all generated code. LLMs were also used for writing assistance in the preparation of this manuscript. The core methodology, architectural choices, and experimental design of phase-DGS were developed by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.