

# Space-Time Forecasting of Dynamic Scenes with Motion-aware Gaussian Grouping

## Supplementary Material

### Contents

#### A. Details of Gaussian Grouping

- A.1. Preliminary: Static Gaussian Grouping
- A.2. Naive Extension of Static Gaussian Grouping
- A.3. Motion-aware Gaussian Grouping

#### B. Implementation Details

- B.1. 4DGS Initialization via SoM
- B.2. Motion-aware Gaussian Grouping
- B.3. Group-wise Optimization
- B.4. Group-wise Motion Forecasting

#### C. Additional Evaluation Results

- C.1. Future View Synthesis
- C.2. Novel View Synthesis

#### D. Component Analysis of MoGaF

- D.1. Analysis for Group-wise Design
- D.2. Effect of Motion-aware Gaussian Grouping

#### E. Limitations and Future Work

### A. Details of Gaussian Grouping

In this section, we provide the detailed algorithms used in our Gaussian grouping pipeline, complementing the description in Section 4.1 of the main paper. We first summarize the static Gaussian grouping mechanism used in Gaga [26] (Section A.1), then describe its naive 4D extension (Section A.2), and finally introduce our motion-aware grouping algorithm that incorporates spatiotemporal cues (Section A.3).

#### A.1. Preliminary: Static Gaussian Grouping

Gaga [26] formulates static 3D Gaussian grouping as the problem of assigning each 3D Gaussian to group label using multi-view 2D instance masks. Given a set of static 3D Gaussians  $\mathcal{G}$  reconstructed from a scene and multi-view images with segmentation masks, each 2D mask  $M$  is first associated with Gaussians whose projections fall inside the mask region:

$$\mathcal{G}(M_i^{(k)}) = \{g \in \mathcal{G} \mid \text{Proj}(g) \in M_i^{(k)}\}, \quad (13)$$

where  $M_i^{(k)}$  is  $k$ -th mask detected in  $i$ -th image.

**3D-aware memory bank.** To enforce cross-view consistency, Gaga employs memory banks storing Gaussian groups  $\{\mathcal{G}^{(k)}\}$ , where each group represents an object-level region in 3D space. At initialization, the masks from the first view are each inserted into the memory bank as separate groups by setting  $\mathcal{G}^{(k)} \leftarrow \mathcal{G}(M_0^{(k)})$  for every mask  $M_0^{(k)}$  in the initial view. For subsequent views, new masks are either merged into one of these

existing groups or used to create additional groups, depending on their overlap with the current memory contents.

**Group ID assignment.** For a new mask  $m = M_i^{(j)}$  extracted from  $i$ -th input image, the similarity between its associated Gaussians  $\mathcal{G}(m)$  and an existing group  $\mathcal{G}^{(k)}$  is computed using the shared-Gaussian overlap:

$$\text{Overlap}(m, k) = \frac{\#(\mathcal{G}^{(k)} \cap \mathcal{G}(m))}{\#(\mathcal{G}(m))}. \quad (14)$$

This formulation depends only on the newly observed set  $\mathcal{G}(m)$ , so the threshold is independent of how many Gaussians have already been accumulated in  $\mathcal{G}^{(k)}$ , avoiding the need to continually retune it as the memory bank grows.

Let  $k^* = \arg \max_k \text{Overlap}(m, k)$  be the group with the highest overlap score. If  $\text{Overlap}(m, k^*)$  exceeds a fixed threshold  $\theta_{\text{ov}}$ , Gaussians in  $\mathcal{G}(m)$  are merged into group  $\mathcal{G}^{(k^*)}$ :

$$\mathcal{G}^{(k^*)} \leftarrow \mathcal{G}^{(k^*)} \cup \mathcal{G}(m). \quad (15)$$

Otherwise, a new group is created in the memory bank with  $\mathcal{G}(m)$  as its initial Gaussian set. In practice, each Gaussian is assigned to at most one group by tracking canonical indices, so once a Gaussian has been inserted into a group, it is not duplicated elsewhere.

**Static setting.** Because Gaga operates on static scenes, each Gaussian maintains a fixed 3D position, making 2D projection overlap a reliable cue for multi-view object grouping. The method produces coherent object-part segmentation and provides the foundation for extending grouping to dynamic scenes in our work.

#### A.2. Naive Extension of Static Gaussian Grouping

**4D-aware Memory Bank.** Given a 2D mask  $M_t^{(k)}$  at timestep  $t$ , we extract deformed Gaussians contributing to the mask:

$$\mathcal{G}_t^{(k)} = \{g \in \mathcal{G} \mid \text{Proj}(g_t) \in M_t^{(k)}\}, \quad (16)$$

where  $g_t$  denotes the deformed state of  $g$ . We extend the static memory bank to maintain motion groups over time. At the first frame, the  $k$ -th memory bank is initialized as

$$\mathcal{M}^{(k)} = \left( \mathcal{G}_t^{(k)}, \tau_{\text{mask}}^{(k)} \right), \quad (17)$$

where  $\tau_{\text{mask}}^{(k)}$  denotes the rigidity type of the segmented region.

**Group ID Assignment.** Different from Gaga [26], we have access to consistent mask identities provided by video segmentation models [32, 33], which serve as a temporal cue for associating  $\mathcal{G}_t^{(k)}$  with the same object across frames. Therefore, we do not apply the max-IoU-based merging strategy used for label-agnostic

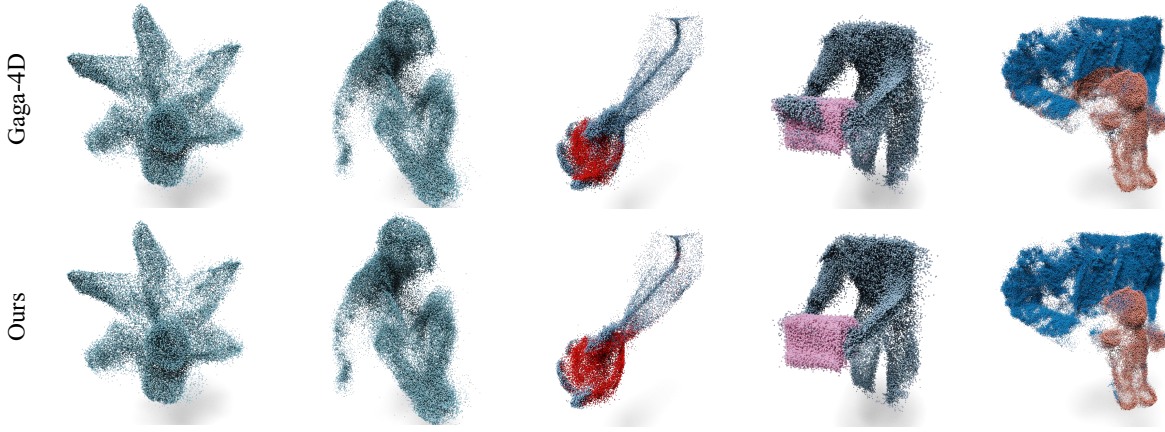


Figure 8. **Gaussian grouping results.** We present Gaussian grouping results of our method and Gaga-4D on iPhone dataset [10], trained on the full sequence.

---

**Algorithm 2: Naive 4D Gaussian Grouping**

---

**Input:** Frames  $\{I_t\}$ , segmentation model  $\mathcal{S}$ , Gaussians  $\mathcal{G}$ , deformed states  $\{g_t\}$ , overlap threshold  $\theta_{ov}$

**Output:** Groups  $\{\mathcal{M}^{(k)} = (G^{(k)}, \tau^{(k)})\}$

**Init (first keyframe):** Obtain

$\{M_{t_0}^{(k)}, \tau_{mask}^{(k)}\} = \mathcal{S}(I_{t_0})$ ; **for**  $k = 1..K$  **do**  
 $G^{(k)} \leftarrow \{g \in \mathcal{G} \mid \text{Proj}(g_{t_0}) \in M_{t_0}^{(k)}\};$   
 $\tau^{(k)} \leftarrow \tau_{mask}^{(k)};$

Define canonical ID  $\text{id}(g)$  for all  $g$ .

**Iterative grouping across frames:** **for**  $t = 1..T$  **do**

Obtain  $\{M_t^{(k)}, \tau_{mask}^{(k)}\};$   
**for**  $k = 1..K$  **do**  
 $G_t^{(k)} \leftarrow \{g \in \mathcal{G} \mid \text{Proj}(g_t) \in M_t^{(k)}\};$   
 Compute  $\text{Overlap}(t, k)$  for all  $k$  using canonical IDs;  
**if**  $\text{Overlap}(t, k) \geq \theta_{ov}$  **then**  
 $G^{(k)} \leftarrow G^{(k)} \cup G_t^{(k)};$

---

matching in static Gaussian segmentation. However, dynamic deformation and occlusions may produce unreliable matches, so we verify consistency using the shared-Gaussian overlap:

$$\text{Overlap}(t, k) = \frac{\#(G^{(k)} \cap G_t^{(k)})}{\#(G_t^{(k)})}. \quad (18)$$

The intersection is defined through canonical identity:

$$G^{(k)} \cap G_t^{(k)} = \{g \in G_t^{(k)} \mid \exists g' \in G^{(k)}, \text{id}(g) = \text{id}(g')\}. \quad (19)$$

If  $\text{Overlap}(t, k) \geq \theta_{ov}$ , we merge:

$$G^{(k)} \leftarrow G^{(k)} \cup G_t^{(k)}. \quad (20)$$

**Limitation of naive 4D extension.** Because this naive extension relies solely on instantaneous 2D projection overlap, Gaussian deformation from motion optimization (e.g., occlusions, drift, partial visibility) can cause inconsistent overlap patterns. As a result, Gaussians belonging to the same object may fail to merge, fragmenting a single object into multiple groups, while unrelated objects may be mistakenly merged due to projected overlap. This makes the naive extension fragile under ambiguous projections (Figure 3a of the main paper), highlighting the need for motion-aware grouping that leverages temporal and motion cues beyond simple projection overlap.

### A.3. Motion-aware Gaussian Grouping

To address the limitations of the naive 4D extension, we propose a spatiotemporal Gaussian grouping algorithm tailored for pre-optimized 4DGS representations. Our method follows an alternating strategy composed of (i) keyframe-based seeding and (ii) feature-space region growing. We first sample  $T_{\text{key}}$  keyframes from the input video, forming the keyframe set  $\{I_t\}_{t=1}^{T_{\text{key}}}$ , and extract Gaussians that contribute to the mask rendering at these frames.

**Iterative Region Growing Strategy.** We initialize motion groups  $\mathcal{M}^{(k)} = \{(G^{(k)}, \tau^{(k)})\}$  by extracting the front Gaussians at a set of keyframes following Equation (16). After initialization, we propagate group assignments to neighboring Gaussians via a spatiotemporal region-growing process.

Each Gaussian  $g \in \mathcal{G}$  is represented by a spatiotemporal feature vector

$$\mathbf{f}_g = [\boldsymbol{\mu}_{c,g}, \mathbf{w}'_g], \quad (21)$$

where  $\boldsymbol{\mu}_{c,g}$  is the canonical-space mean of  $g$ , and  $\mathbf{w}'_g$  denotes its PCA-reduced motion coefficient.

Given these features, each group  $G^{(k)}$  is expanded by aggregating Gaussians in its feature-space neighborhood. For every  $g \in G^{(k)}$ , we consider its nearest neighbors and include any candidate  $g'$  if

$$\|\mathbf{f}_g - \mathbf{f}_{g'}\| < \epsilon_r, \quad g' \notin G^{(k)}. \quad (22)$$

---

**Algorithm 3: Complete Procedure of Motion-aware Gaussian Grouping**


---

**Input:** Video frames  $\{I_t\}_{t=1}^T$ , segmentation model  $\mathcal{S}$ , dynamic Gaussian set  $\mathcal{G}$  with deformed states  $\{g_t\}_{t=1}^T$

**Output:** Motion groups  $\{\mathcal{M}^{(k)} = (G^{(k)}, \tau^{(k)})\}_{k=1}^K$

**Stage 1: Init at the First Keyframe**

Obtain  $\{M_t^{(k)}, \tau_{\text{mask}}^{(k)}\} \leftarrow \mathcal{S}(\{I_t\})$

Select first keyframe  $t_k$

**foreach**  $k = 1, \dots, K$  **do**

$G^{(k)} \leftarrow \{g \in \mathcal{G} \mid \text{Proj}(g_{t_k}) \in M_{t_k}^{(k)}\}$   
 $\tau^{(k)} \leftarrow \tau_{\text{mask}}^{(k)}$

Define features for all  $g$ :  $\mathbf{f}_g = [\mu_{c,g}, \mathbf{w}'_g]$

**Stage 2: Iterative Region Growing**

**foreach** keyframe  $t$  in stride order **do**

// (A) Feature-space region growing

**repeat**

$U \leftarrow \mathcal{G} \setminus \bigcup_k G^{(k)}$

**foreach**  $k = 1, \dots, K$  **do**

$\epsilon_k \leftarrow \alpha \cdot \text{mean}_{g \in G^{(k)}} [\text{KNN}_K(\mathbf{f}_g)]$

$N_k \leftarrow \{g' \in U \mid$

$\min_{g \in G^{(k)}} \|\mathbf{f}_g - \mathbf{f}_{g'}\| < \epsilon_k\}$

$G^{(k)} \leftarrow G^{(k)} \cup N_k$

$U \leftarrow U \setminus N_k$

**until** no group changes;

// (B) Seeding & merge at keyframe  $t$

**foreach**  $k = 1, \dots, K$  **do**

$G_t^{(k)} \leftarrow \{g \mid \text{Proj}(g_t) \in M_t^{(k)}\}$

$G^{(k)} \leftarrow G^{(k)} \cup G_t^{(k)}$

**Stage 3: Reassign Unassigned Gaussians (KNN Voting)**

$G^{\text{labeled}} \leftarrow \bigcup_k G^{(k)}, U \leftarrow \mathcal{G} \setminus G^{\text{labeled}}$

**foreach**  $g_u \in U$  **do**

$\mathcal{N}(g_u) \leftarrow \text{KNN}_K(\mathbf{f}_{g_u}, G^{\text{labeled}})$

**foreach**  $k = 1, \dots, K$  **do**

$v_k(g_u) \leftarrow \sum_{g' \in \mathcal{N}(g_u)} \mathbb{I}[g' \in G^{(k)}]$

$k^*(g_u) = \arg \max_k v_k(g_u)$

$G^{(k^*(g_u))} \leftarrow G^{(k^*(g_u))} \cup \{g_u\}$

---

This update is repeated until convergence. We then perform a new seeding step at the next keyframe and apply region growing again. By alternating between these two stages, the grouping progressively stabilizes and captures coherent spatiotemporal structure across the sequence.

**Handling Unassigned Gaussians.** After the alternating seeding–region-growing stages, a small number of Gaussians may remain unassigned because they were not included in any feature-space neighborhood expansion. To relabel these remaining Gaussians, we perform a feature-space voting procedure using only group-labeled Gaussians.

For each unassigned Gaussian  $g_u$ , we retrieve its  $K$  nearest labeled neighbors in feature space:

$$\mathcal{N}(g_u) = \text{KNN}_K(\mathbf{f}_{g_u}, \mathcal{G}^{\text{labeled}}), \quad (23)$$

where  $\mathbf{f}_{g_u}$  is the spatiotemporal feature of  $g_u$  and  $\mathcal{G}^{\text{labeled}} = \bigcup_k G^{(k)}$  denotes the set of Gaussians with assigned group labels. Among these labeled neighbors, we count how many belong to each motion group. Let  $\mathbb{I}[g' \in G^{(k)}]$  denote an indicator function that evaluates to 1 if  $g'$  belongs to group  $k$ , and 0 otherwise. The voting score for group  $k$  is computed as

$$v_k(g_u) = \sum_{g' \in \mathcal{N}(g_u)} \mathbb{I}[g' \in G^{(k)}]. \quad (24)$$

The final group assignment for  $g_u$  is then obtained by majority voting:

$$k^*(g_u) = \arg \max_k v_k(g_u). \quad (25)$$

We then update the motion group as

$$G^{(k^*)} \leftarrow G^{(k^*)} \cup \{g_u\}. \quad (26)$$

This voting-based reassignment ensures that isolated or boundary Gaussians are consistently grouped according to their local spatiotemporal context.

As shown in Figure 8, our grouping algorithm works robustly across multiple objects. It produces clear and stable group assignments even in scenes where distinct objects exhibit significant spatial overlap in the input video.

## B. Implementation Details

### B.1. 4DGS Initialization via SoM

Our method builds on pre-optimized 4DGS representation following Shape-of-Motion (SoM) [37], trained for 300 epochs. We use the same optimization hyperparameters for all benchmark datasets, following the official SoM implementation. For the future view synthesis experiments, we use only the first 60% or 80% of the input sequence to evaluate extrapolation performance.

### B.2. Motion-aware Gaussian Grouping

**Iterative Region Growing.** We implement our grouping strategy by modifying the official Gaga [26] implementation. During initialization and keyframe-based seeding, we extract the closest 40% front Gaussians with respect to the camera view. For the region-growing stage, each Gaussian is represented as a spatiotemporal feature vector  $[\mu_{c,g}, \mathbf{w}'_g]$ , composed of its canonical-space mean and a PCA-reduced motion coefficient. We apply PCA to the motion coefficients of all dynamic Gaussians in 4DGS representation, scale the resulting PCA components by 0.3, and concatenate them with the Gaussian means. Before performing region growing for each group, we determine the growing radius  $\epsilon_r$  by computing the average neighbor distance among Gaussians belonging to the same group, ensuring that the expansion scale reflects the underlying local density.

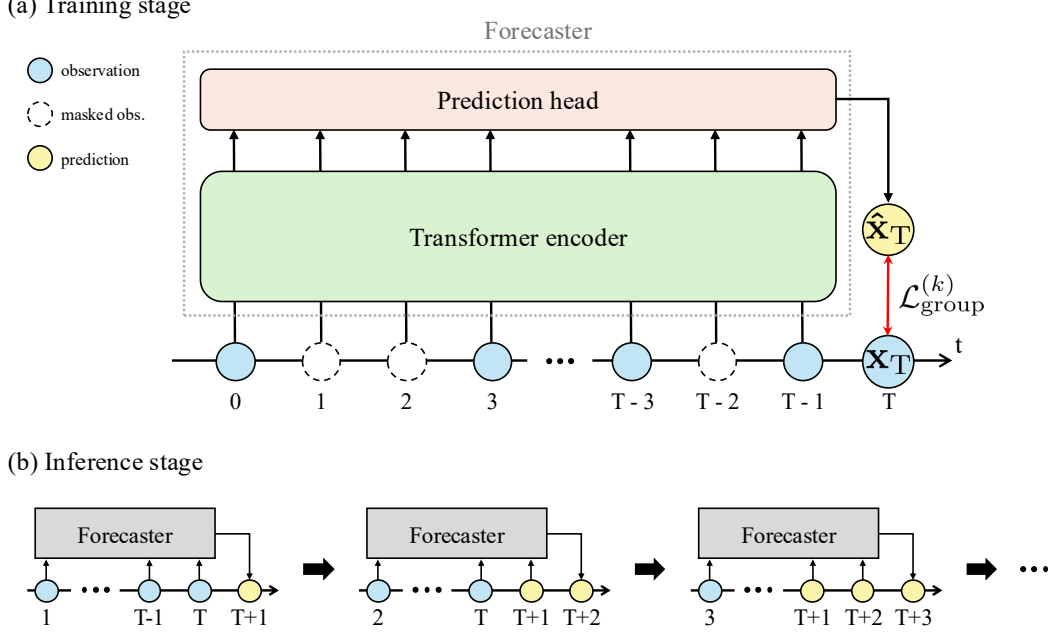


Figure 9. **Overview of the forecaster.** (a) *Training stage*: The forecaster is trained for each Gaussian group  $G^{(k)}$  by minimizing the loss  $\mathcal{L}_{\text{group}}^{(k)}$  between the predicted  $\hat{\mathbf{x}}_T$  and the observed  $\mathbf{x}_T$  at time  $T$ . We randomly apply contiguous masking to the inputs and gradually reduce the masking ratio later in training. (b) *Inference stage*: Future motion is generated via autoregressive rollout for each  $G^{(k)}$ , where the input sequence is iteratively updated with each newly predicted frame. Note that the input length is always  $T-1$ , the same as in training.

**Reassignment Strategy.** In addition, during the Gaussian re-labeling step for unassigned Gaussians, each Gaussian updates its group label by voting among the 8 nearest group-assigned neighbors. To further enforce reliable and spatially consistent grouping, we then reassign only the closest 50% of unassigned Gaussians, determined by their average distance to the neighboring group-assigned Gaussians.

### B.3. Group-wise Optimization

Our group-wise optimization consists of two main stages: (i) rigid motion initialization, and (ii) group-wise refinement. In the first stage, we initialize rigid group motion parameters to enable our motion-aware objective. In the second stage, using the initialized motion information, we refine pre-optimized 4DGS representation to enforce stronger spatiotemporal consistency.

**Rigid Motion Initialization.** Before refining 4DGS representation, we first estimate the rigid body transformation parameters for rigid groups (i.e., groups with  $\tau^{(k)} = 1$ ) to implement our rigid motion anchoring loss. Specifically, for each rigid motion group, we compute a sequence of rigid transformations  $\{\Phi_t^{(k)} = [\mathbf{R}_{c \rightarrow t}^{(k)} \mid \mathbf{t}_{c \rightarrow t}^{(k)}]\}_{t=1}^T$  by solving a Procrustes alignment problem. The alignment at timestep  $t$  is given by

$$\Phi_t^{(k)} = \arg \min_{\mathbf{R}_{c \rightarrow t}^{(k)}, \mathbf{t}_{c \rightarrow t}^{(k)}} \sum_{g \in G^{(k)}} \left\| \mathbf{R}_{c \rightarrow t}^{(k)} \boldsymbol{\mu}_{c,g} + \mathbf{t}_{c \rightarrow t}^{(k)} - \boldsymbol{\mu}_{t,g} \right\|_2^2, \quad (27)$$

where  $\mathbf{R}_{c \rightarrow t}^{(k)} \in \text{SO}(3)$  and  $\mathbf{t}_{c \rightarrow t}^{(k)} \in \mathbb{R}^3$ . Here,  $\boldsymbol{\mu}_{c,g}$  and  $\boldsymbol{\mu}_{t,g}$  denote the canonical and deformed Gaussian centers.

**Refinement of 4DGS Representation.** Similar to the pre-optimization stage (Section B.1), our refinement is also implemented on top of SoM. We incorporate our group-wise motion optimization objective  $\mathcal{L}_{\text{motion}}$  into the original training loss. For optimization stability, we apply scaling weights and reformulate the objective as:

$$\mathcal{L}_{\text{motion}} = \sum_{k=1}^K \left[ \tau^{(k)} \cdot \lambda_{\text{rigid}} \mathcal{L}_{\text{rigid}}^{(k)} + (1 - \tau^{(k)}) \cdot \lambda_{\text{nr}} \mathcal{L}_{\text{nr}}^{(k)} \right], \quad (28)$$

where we set  $\lambda_{\text{rigid}} = 0.1$  and  $\lambda_{\text{nr}} = 0.02$  in all experiments.

We apply our constraint optimization objective together with the original reconstruction loss of SoM [37], defined as

$$\mathcal{L}_{\text{recon}} = \|\hat{\mathbf{I}} - \mathbf{I}\|_1 + \lambda_{\text{depth}} \|\hat{\mathbf{D}} - \mathbf{D}\|_1 + \lambda_{\text{mask}} \|\hat{\mathbf{M}} - \mathbf{M}\|_1, \quad (29)$$

where  $\hat{\mathbf{I}}, \hat{\mathbf{D}}, \hat{\mathbf{M}}$  denote the rendered RGB, depth, and mask predictions from current 4DGS representation. We follow the loss-weight configuration provided in the official SoM implementation. Our final optimization objective is then formulated as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{motion}}. \quad (30)$$

This combined objective ensures that the 4DGS representation remains photometrically faithful while also enforcing coherent group-wise rigid and non-rigid motion behavior.

### B.4. Group-wise Motion Forecasting

**Architecture Details.** We illustrate the architecture of our forecaster in Figure 9. The model uses a single-layer Transformer encoder with a 32-dimensional hidden state, eight attention heads,



Table 4. **Forecasting results on the iPhone dataset.** We forecast frames beyond the observed time window and evaluate the predicted frames rendered from held-out test viewpoints. The leftmost column (*Obs. ratio*) indicates the fraction of input training frames used by each model.

<i>Obs. ratio</i>	Scene	GSPred			GSPred-SoM			MoGaF (Ours)		
		mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓
70%	apple	13.21	0.5562	0.6942	<b>16.58</b>	<b>0.7339</b>	<b>0.5007</b>	16.45	0.7220	0.5329
	block	13.11	0.5333	0.5240	13.47	<b>0.6192</b>	<b>0.4900</b>	<b>14.10</b>	0.6110	0.5030
	paper-windmill	14.67	0.2168	0.4410	18.73	<b>0.5221</b>	0.2191	<b>18.98</b>	0.5197	<b>0.2108</b>
	spin	15.67	0.4738	0.3975	<b>16.00</b>	<b>0.6797</b>	<b>0.3049</b>	15.87	0.6618	0.3438
	teddy	11.46	<b>0.5886</b>	0.5708	10.84	0.5666	0.6555	<b>11.87</b>	0.5562	<b>0.5578</b>
	Average	13.62	0.4737	0.5255	15.12	<b>0.6243</b>	0.4341	<b>15.45</b>	0.6141	<b>0.4296</b>
90%	apple	15.21	0.5550	0.4574	16.88	<b>0.8541</b>	<b>0.4453</b>	<b>16.98</b>	0.8526	0.4530
	block	<b>14.57</b>	0.5445	<b>0.4909</b>	12.49	0.5896	0.5363	14.19	<b>0.6018</b>	0.4912
	paper-windmill	14.78	0.2201	0.4280	19.00	<b>0.5559</b>	0.2111	<b>19.22</b>	0.5538	<b>0.2036</b>
	spin	14.75	0.4631	0.4232	16.14	0.6931	0.3146	<b>16.99</b>	<b>0.6975</b>	<b>0.2933</b>
	teddy	11.37	<b>0.5816</b>	0.5929	11.99	0.5411	0.5894	<b>12.39</b>	0.5433	<b>0.5436</b>
	Average	14.13	0.4728	0.4785	15.30	0.6467	0.4193	<b>15.95</b>	<b>0.6498</b>	<b>0.3970</b>

and a 64-dimensional feedforward layer, followed by a flatten-head decoder. We apply a dropout rate of 0.2 and use instance normalization for each sequence. For both training and inference, we represent each Gaussian motion  $\mathbf{x}_t$  at time  $t$  as

$$\mathbf{x}_t = [\boldsymbol{\mu}_t, \mathbf{q}_t] \in \mathbb{R}^7, \quad (31)$$

where  $\boldsymbol{\mu}_t \in \mathbb{R}^3$  denotes the 3D translation vector and  $\mathbf{q}_t \in \mathbb{R}^4$  denotes the unit quaternion that parameterizes the rotation.

**Training Procedure and Inference Scheme.** During training, we employ variable observation masking, where a contiguous span covering 40–0% of the sequence is masked, and the masking ratio is gradually annealed over the course of training. To encourage physically plausible and smooth motion, we apply an acceleration regularization term with  $\lambda_{\text{acc}} = 1.0$  for all experiments.

We further adopt a group-wise forecasting scheme in which one forecaster is trained per motion group, and each forecaster is responsible for training and predicting only the trajectories associated with its assigned group.

## C. Additional Evaluation Results

### C.1. Future View Synthesis

**Evaluation on the iPhone Dataset.** For the iPhone dataset [10], we provide quantitative evaluation results for additional observation ratios (70% and 90%) in Table 4. We further present additional qualitative results for long-term forecasting (Figure 11) and qualitative comparisons across diverse scenes (Figure 12), extending Figure 4.

**Evaluation on D-NeRF dataset.** We further report evaluation results on D-NeRF dataset [31]. Using the same setup as GSPred [50], we use an 80% observation ratio and a 20% test split. In this setting, Gaussian motions are forecasted, and all trajectories are extrapolated using a weighted sum of the predicted motion. The quantitative results are summarized in Table 5. Additionally, qualitative comparisons against GSPred are presented in Figure 13.

Table 5. **Forecasting results on D-NeRF dataset.** We use 80% of the frames as input and predict the remaining 20%.

Scene	GSPred [50]			Ours		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Trex	<b>21.09</b>	<b>0.9406</b>	<b>0.0461</b>	20.60	0.9405	0.0499
Mutant	28.16	<b>0.9560</b>	0.0256	<b>28.72</b>	0.9552	<b>0.0218</b>
Jumpingjacks	<b>20.51</b>	0.9184	0.0760	20.12	<b>0.9220</b>	<b>0.0734</b>
Standup	25.96	0.9403	0.0481	<b>30.38</b>	<b>0.9544</b>	<b>0.0335</b>
Bouncingballs	26.63	0.9714	<b>0.0361</b>	<b>26.82</b>	<b>0.9728</b>	0.0367
Hook	23.42	<b>0.9089</b>	0.0573	<b>24.13</b>	0.9081	<b>0.0562</b>
Hellwarrior	30.75	0.9281	0.0729	<b>31.29</b>	<b>0.9323</b>	<b>0.0680</b>
Lego	11.99	0.7562	0.2338	<b>24.88</b>	<b>0.9005</b>	<b>0.0532</b>
<b>Average</b>	23.56	0.9150	0.0745	<b>25.87</b>	<b>0.9357</b>	<b>0.0491</b>

### C.2. Novel View Synthesis

To analyze the effect of our group-wise optimization on scene interpolation, we evaluate novel view synthesis (NVS) results and point-tracking performance on iPhone dataset. The experiment is conducted using the full training set (100% observation ratio) so that the evaluation reflects pure interpolation performance.

As reported in Table 6, our constrained optimization yields slight improvements in both point-tracking accuracy and photometric metrics. Qualitatively, Figure 10 shows that our method enhances the geometric consistency of resulting 4DGS representation. These results confirm that our Gaussian grouping and group-wise constrained optimization help obtain more physically plausible dynamic Gaussian representation.

## D. Component Analysis of MoGaF

For a deeper analysis of our pipeline, we conduct a component-wise analysis of MoGaF and evaluate the effectiveness of our motion-aware Gaussian grouping strategy. The component analysis focuses on three key design axes: (1) group-wise optimization, (2) the group-wise forecasting scheme, and (3) the capacity of the motion forecaster. For the grouping strategy, we compare our method with a simple extension of static Gaussian grouping [26]. All experiments are conducted with a 60% observation ratio, forecasting the remaining 40% of each sequence on iPhone dataset [10].

Table 6. **Quantitative evaluation results on scene interpolation.** We compare tracking and photometric performance against SoM [37] on iPhone dataset [10].

Methods	3D tracking			2D tracking			Photometric		
	EPE↓	$\delta_{3D}^{10}\uparrow$	$\delta_{3D}^{05}\uparrow$	AJ↑	$\delta_{avg}\uparrow$	OA↑	MPSNR↑	mSSIM↑	mLPIPS↓
SoM [37]	0.1016	70.62	46.59	37.61	49.17	<b>86.99</b>	16.72	<b>0.6462</b>	0.3914
MoGaF (Ours)	<b>0.0989</b>	<b>70.63</b>	<b>47.64</b>	<b>38.54</b>	<b>49.93</b>	86.85	<b>16.75</b>	<b>0.6462</b>	<b>0.3887</b>

Table 7. **Ablation studies on group-wise design and region growing.** We analyze the contribution of each component in our method across 3D/2D motion tracking and novel view synthesis.

(a) Effect of group-wise optimization, forecasting, and encoder capacity on future Gaussian motion forecasting. The gray row denotes MoGaF (Ours).

Encoder	Group-wise		3D tracking			2D tracking		
	Opt.	Forecast	EPE↓	$\delta_{3D}^{10}\uparrow$	$\delta_{3D}^{05}\uparrow$	AJ↑	$\delta_{avg}\uparrow$	OA↑
5-Layer (HC)	—	—	0.253	37.5	12.7	12.8	7.9	75.9
	✓	—	0.264	40.0	15.8	13.0	7.6	77.8
	—	✓	0.246	42.6	16.7	16.0	8.5	74.8
	✓	✓	0.245	43.5	19.5	16.1	<b>9.4</b>	79.2
1-Layer (LW)	—	—	0.296	35.6	17.1	13.3	8.5	64.1
	✓	—	0.269	36.7	15.8	13.4	7.6	70.5
	—	✓	0.237	42.9	16.9	15.4	8.6	72.2
	✓	✓	<b>0.236</b>	<b>44.8</b>	<b>22.5</b>	<b>17.5</b>	8.8	<b>80.1</b>

(b) Forecasting results comparing a naive extension (Gaga-4D) and our motion-aware Gaussian grouping.

Method	mPSNR ↑	mSSIM ↑	mLPIPS ↓	$\delta_{3D}^{10}\uparrow$	$\delta_{3D}^{05}\uparrow$	OA ↑
Gaga-4D	15.31	0.6015	0.4475	43.4	20.9	75.6
Ours	<b>15.51</b>	<b>0.6143</b>	<b>0.4245</b>	<b>44.8</b>	<b>22.5</b>	<b>80.1</b>

## D.1. Analysis for Group-wise Design

To isolate the impact of each component, we disable the group-wise optimization and group-wise forecasting modules individually. We further evaluate a higher-capacity motion forecaster—a 5-layer transformer encoder (HC)—against our default 1-layer transformer (LW) encoder. To accurately evaluate forecasting performance, we compare various configurations using point tracking accuracy.

**Effect of Group-wise Optimization.** Table 7a summarizes the results of our component analysis. For both HC and LW forecasters, enabling group-wise optimization consistently improves 3D tracking accuracy. This suggests that object-level motion constraints help preserve coherent and stable spatiotemporal trajectories within 4DGS representation.

**Contribution of Group-wise Forecasting.** Group-wise forecasting further improves performance across all metrics. By learning motion patterns within each object independently, the forecaster produces more stable long-term predictions and higher 2D tracking accuracy. When combined with group-wise optimization, it yields the best results for both HC and LW settings, highlighting the complementary roles of the two modules.

**Effect of Forecaster Capacity.** Interestingly, the HC forecaster does not provide noticeable improvement over the LW ver-

sion. We hypothesize that larger models may be more sensitive to noisy or frame-specific motion variations in 4DGS, tend to introduce instability in long-term predictions. In contrast, the lightweight forecaster appears to leverage the structured, object-level motion units more effectively, suggesting that strong forecasting performance is achievable without large model capacity.

## D.2. Effect of Motion-aware Gaussian Grouping

We evaluate the effectiveness of our motion-aware Gaussian grouping strategy for scene forecasting. Specifically, we compare our approach with a baseline that applies iterative region growing based on a naive extension of static Gaussian grouping (Gaga-4D), which is described in Section A.2.

As shown in Table 7b, our method consistently outperforms the baseline across all evaluation metrics, including photometric metrics and point tracking accuracy. These results indicate that our motion-aware grouping enables more accurate separation of Gaussians with coherent motion, leading to improved extrapolation under out-of-distribution (OOD) conditions. Overall, these results demonstrate that our region growing strategy effectively enhances motion awareness in Gaussian grouping under complex dynamic scenes.

## E. Limitations and Future Work

Our method exhibits several limitations arising from its dependence on pretrained 4DGS representation. Since the grouping, optimization, and forecasting stages operate on geometry produced by the baseline 4DGS, the overall quality is bounded by the fidelity of this reconstruction. Thus, the method cannot recover unseen geometry, and severe failures in the underlying motion optimization—especially in highly complex scenes—directly propagate to later stages. Incorporating point-cloud completion techniques or generative priors may alleviate this dependency by providing additional geometric constraints.

Another limitation appears when object groups overlap during forecasting. Because Gaussians are not physical primitives, our model cannot reason about collisions or inter-object interactions, offering limited ability to detect or prevent physically implausible overlap between motion groups. Extending the framework with physical constraints or physically grounded Gaussian representations would be a promising direction for future work.





Figure 10. **Qualitative evaluation results on scene interpolation.** We report NVS performance of SoM [37] and MoGaF on iPhone dataset [10], evaluated on test camera viewpoints over the full training sequence.

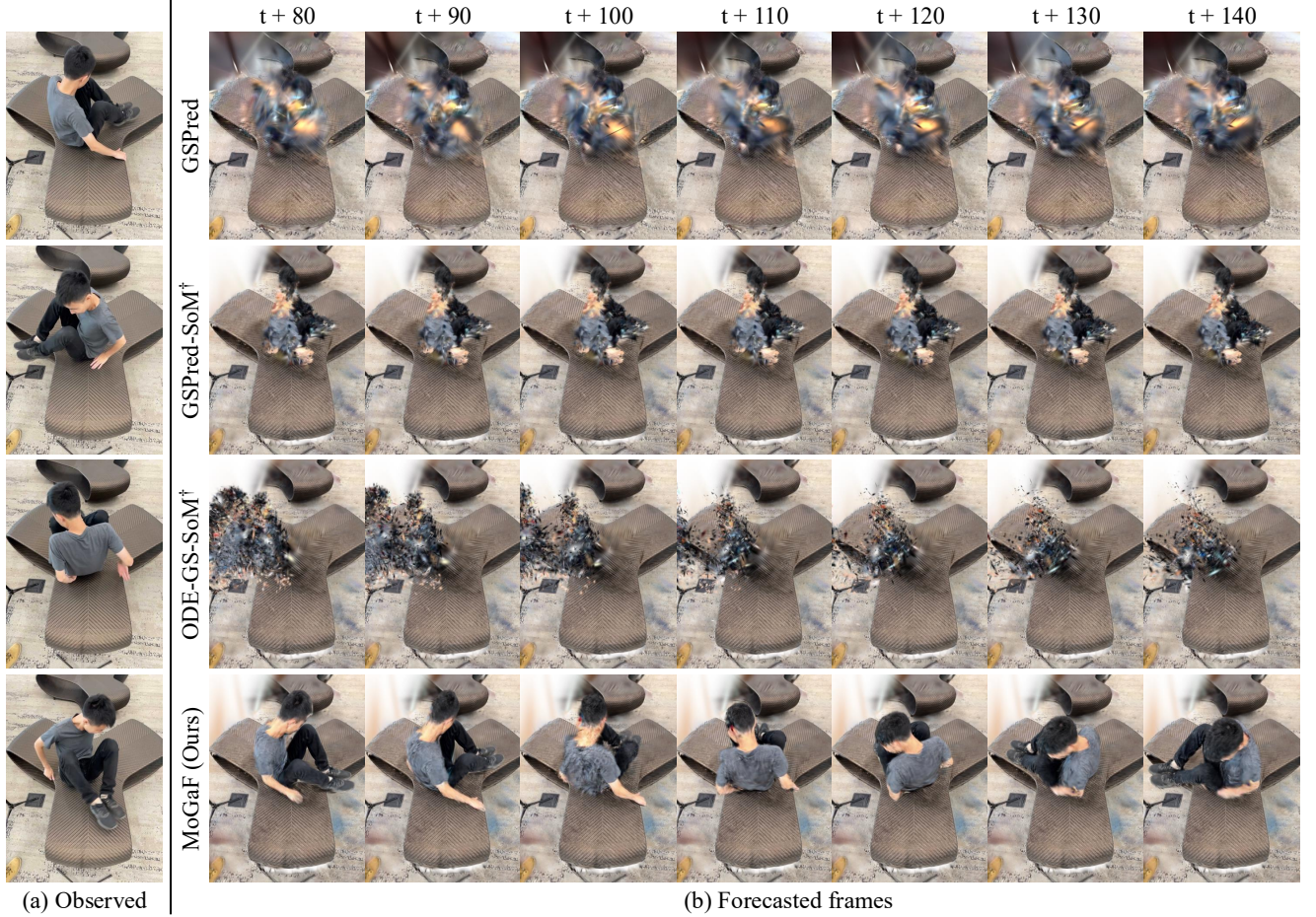


Figure 11. **Long-term forecasting results on iPhone dataset.** (a) shows GT views from test cameras at the observed timesteps, and (b) presents the forecasted renderings for timesteps beyond the observations. Note that  $t$  denotes the timestep of the last observed frame.



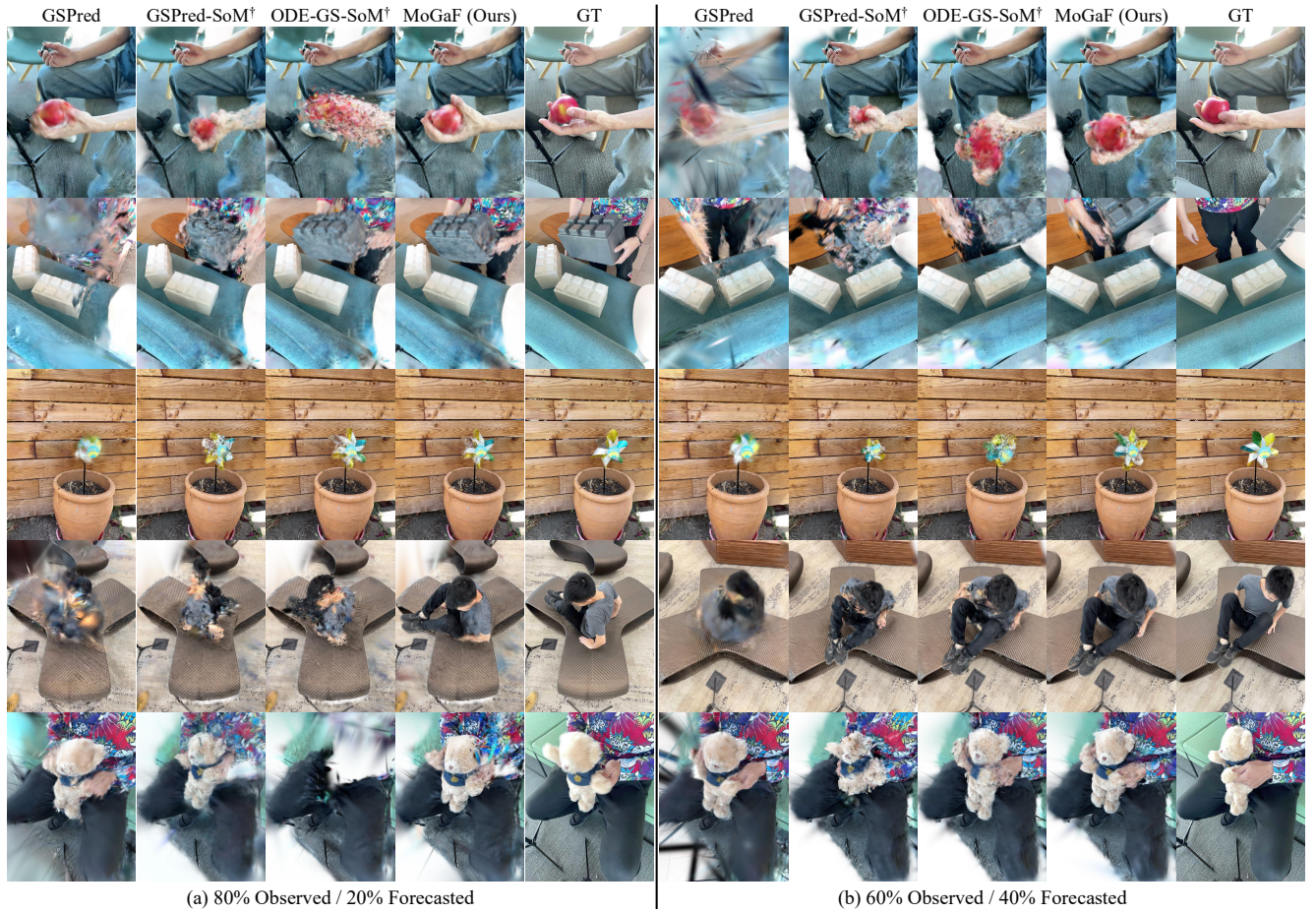


Figure 12. **Qualitative results on iPhone dataset.** We present forecasted frames from test camera views. (a) and (b) correspond to settings where the first 80% and 60% of frames are used for training, and the remaining 20% and 40% are forecasted, respectively.

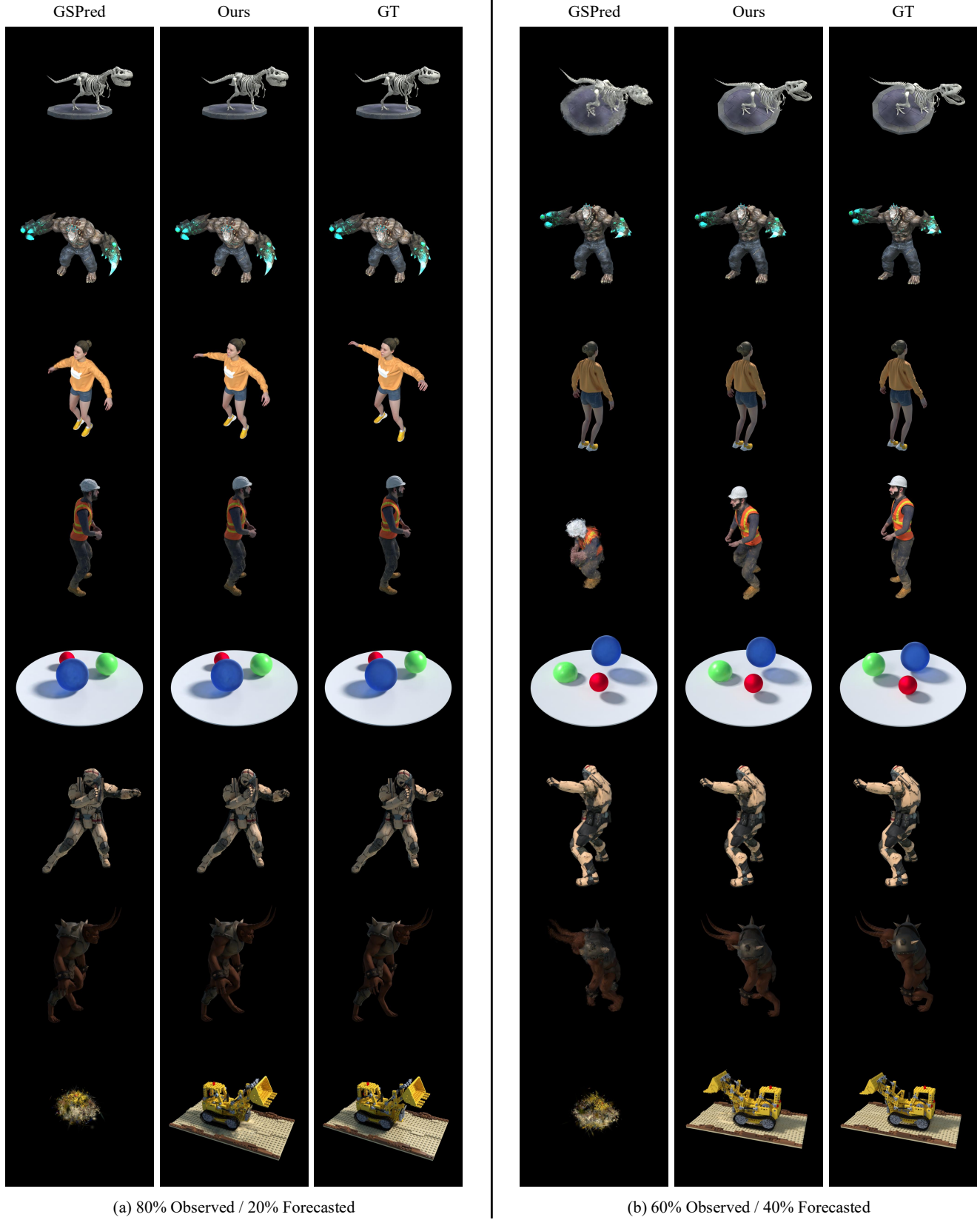


Figure 13. **Qualitative results on D-NeRF dataset.** We present forecasted frames from test camera views. (a) and (b) correspond to settings where the first 80% and 60% of frames are used for training, and the remaining 20% and 40% are forecasted, respectively.



## References

- [1] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zhohus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025. 1
- [2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2
- [3] Jiazhong Cen, Jiemin Fang, Zanwei Zhou, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment anything in 3d with radiance fields. *International Journal of Computer Vision*, pages 1–23, 2025. 2
- [4] Xiaokang Chen, Jiaxiang Tang, Diwen Wan, Jingbo Wang, and Gang Zeng. Interactive segment anything nerf with feature imitation. *arXiv preprint arXiv:2305.16233*, 2023. 2
- [5] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1316–1326, 2023. 2
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. 5
- [7] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023. 2, 6
- [8] Bin Dou, Tianyu Zhang, Zhaohui Wang, Yongjia Ma, Zejian Yuan, and Nanning Zheng. Learning segmented 3d gaussians via efficient feature unprojection for zero-shot neural scene segmentation. In *International Conference on Neural Information Processing*, pages 398–412. Springer, 2024. 2
- [9] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2
- [10] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 35:33768–33780, 2022. 6, 2, 5, 7
- [11] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13505–13515, 2021. 1
- [12] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022. 2
- [13] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 1
- [14] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Co-tracker: It is better to track together. In *European conference on computer vision*, pages 18–35. Springer, 2024. 2, 6
- [15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3
- [16] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21530–21539, 2024. 2
- [17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 2
- [18] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision*, pages 252–269. Springer, 2024. 2
- [19] Yong-Hoon Kwon and Min-Gyu Park. Predicting future frames using retrospective cycle gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1811–1820, 2019. 1
- [20] Jiahui Lei, Yijia Weng, Adam W Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 6165–6177, 2025. 2
- [21] 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. 2
- [22] Wenqian Liu, Abhishek Sharma, Octavia Camps, and Mario Sznajder. Dyan: A dynamical atoms-based network for video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 170–185, 2018. 2
- [23] Chaochao Lu, Michael Hirsch, and Bernhard Scholkopf. Flexible spatio-temporal networks for video prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6523–6531, 2017. 1
- [24] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 2
- [25] Zhanpeng Luo, Haoxi Ran, and Li Lu. Instant4d: 4d gaussian splatting in minutes, 2025. 2
- [26] Weijie Lyu, Xueting Li, Abhijit Kundu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Gaga: Group any gaussians via 3d-aware

- memory bank. *arXiv preprint arXiv:2404.07977*, 2024. 2, 3, 4, 6, 1, 5
- [27] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2
- [28] Ruibo Ming, Zhewei Huang, Zhuoxuan Ju, Jianming Hu, Lihui Peng, and Shuchang Zhou. A survey on future frame synthesis: Bridging deterministic and generative approaches. *arXiv preprint arXiv:2401.14718*, 2024. 1
- [29] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5865–5874, 2021. 2
- [30] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021.
- [31] 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. 2, 7, 5
- [32] 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. 2, 1
- [33] Tianhe Ren, Qing Jiang, Shilong Liu, Zhaoyang Zeng, Wenlong Liu, Han Gao, Hongjie Huang, Zhengyu Ma, Xiaoke Jiang, Yihao Chen, Yuda Xiong, Hao Zhang, Feng Li, Peijun Tang, Kent Yu, and Lei Zhang. Grounding dino 1.5: Advance the “edge” of open-set object detection, 2024. 3, 1
- [34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5
- [35] Angel Villar-Corrales, Ismail Wahdan, and Sven Behnke. Object-centric video prediction via decoupling of object dynamics and interactions. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 570–574. IEEE, 2023. 2
- [36] Daniel Wang, Patrick Rim, Tian Tian, Dong Lao, Alex Wong, and Ganesh Sundaramoorthi. Ode-gs: Latent odes for dynamic scene extrapolation with 3d gaussian splatting. *arXiv preprint arXiv:2506.05480*, 2025. 2, 3, 5, 6
- [37] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9660–9672, 2025. 2, 3, 5, 6, 7, 4
- [38] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d LSTM: A model for video prediction and beyond. In *International Conference on Learning Representations*, 2019. 2
- [39] 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. 2, 6
- [40] Jialong Wu, Shaofeng Yin, Ningya Feng, Xu He, Dong Li, Jianye Hao, and Mingsheng Long. ivideoopt: Interactive videoopts are scalable world models. *Advances in Neural Information Processing Systems*, 37:68082–68119, 2024. 1
- [41] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5539–5548, 2020. 1
- [42] Yue Wu, Qiang Wen, and Qifeng Chen. Optimizing video prediction via video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17814–17823, 2022. 1
- [43] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videoopt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 2
- [44] Jinyu Yang, Mingqi Gao, Zhe Li, Shang Gao, Fangjing Wang, and Feng Zheng. Track anything: Segment anything meets videos. *arXiv preprint arXiv:2304.11968*, 2023. 2
- [45] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10371–10381, 2024. 2, 6
- [46] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 2
- [47] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20331–20341, 2024. 2
- [48] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pages 162–179. Springer, 2024. 2, 3
- [49] Xi Ye and Guillaume-Alexandre Bilodeau. Vptr: Efficient transformers for video prediction. In *2022 26th International conference on pattern recognition (ICPR)*, pages 3492–3499. IEEE, 2022. 2
- [50] Boming Zhao, Yuan Li, Ziyu Sun, Lin Zeng, Yujun Shen, Rui Ma, Yinda Zhang, Hujun Bao, and Zhaopeng Cui. Gaussianprediction: Dynamic 3d gaussian prediction for motion extrapolation and free view synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–12, 2024. 1, 2, 3, 5, 6, 7
- [51] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suyu You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. 2

- [52] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. [3](#)