Mark4D: Temporally-Consistent Watermarking for 4D Gaussian Splatting

Supplementary Material

We provide supplementary materials to support the main manuscript. Specifically, we include: **1)** detailed method descriptions, including the X-CLIP tokenization process and the distortion module; **2)** implementation details and hyperparameter settings of our training pipeline and baseline settings; **3)** extended experimental results such as full quantitative results for each scene of the datasets, ablations of our method, complexity analyses, and sensitivity studies; **4)** additional qualitative visualizations which were not included in the main manuscript due to the page limit.

A. Method Details

A.1. X-CLIP Tokenization

Algorithm 1 provides an overview of the X-CLIP [13] tokenization process.

In our implementation, we employ the ViT-B-32 X-CLIP model, which inherits the Byte-Pair Encoding (BPE) tokenizer in CLIP [15]. This tokenizer defines a fixed vocabulary of 49,408 token IDs (0 to 49,407), where token ID 0 is treated as the padding token, and the two highest IDs, 49406 and 49407, are reserved to indicate sequence boundaries w_{start} and w_{end}, respectively. All remaining IDs (1 to 49405) represent standard subword units and constitute the token space used for mapping individual watermark bits.

To convert an L-bit watermark message into a text token sequence, we construct a deterministic bit-to-token mapping table $\Phi(m_i, i)$, where $\Phi(0, i)$ and $\Phi(1, i)$ denote two distinct token IDs assigned to the i-th bit position. At initialization, for each $i \in \{1, \ldots, L\}$, we randomly sample two different token IDs from the valid token range (1-49405), and we fix this mapping for the entire training and evaluation process. To satisfy the fixed input length $W_{\max}$ required by the X-CLIP text encoder, we append padding tokens (ID $= 0$) after w_{end}. These padding positions are masked out by the encoder's attention mask, to ensure that they do not participate in the self-attention computation.

Algorithm 1 X-CLIP Tokenization

Require: Binary message $M = \{m_i\}_{i=1}^{L}$, $m_i \in \{0, 1\}$; vocab size $|V|$; max token length $W_{\max}$; start token w_{start}; end token w_{end}.

 # Step 1: Initialize bit-to-token mapping Φ
Random sample $2L$ distinct token IDs $\{\omega_1, \ldots, \omega_{2L}\}$ from $[1, |V|-2]$
initialize Φ
for $i = 1$ to L **do**
  $\Phi(0, i) \leftarrow \omega_{2i-1}$
  $\Phi(1, i) \leftarrow \omega_{2i}$
end for
 # Step 2: Construct token sequence
$W \leftarrow [\,]$
append w_{start} to W
for $i = 1$ to L **do**
  $w_i \leftarrow \Phi(m_i, i)$
  append w_i to W
end for
append w_{end} to W
 # Step 3: Zero-padding
if $|W| < W_{\max}$ **then**
  append zeros until $|W| = W_{\max}$
end if
return W

A.2. Distortion Module

The differentiable distortion module $\mathcal{A}$ is used in the training stage of the watermark offsets in Eq. (6) of the main paper, where it comprises a set of *frame-level* and *video-level* perturbations designed to simulate realistic degradations:

- **Frame-level:** Gaussian noise (σ=0.1), rotation (random rotation of the frames within a range $[-\pi/6, +\pi/6]$), scaling (random scaling of the frames within a range of ratio $[0.75, 1.25]$), Gaussian blur (Gaussian kernel of size= 3 with $\sigma = 0.1$), cropping (40%), brightness jittering (random within a range of ratio $[0.5, 1.5]$), and JPEG compression (50% quality).
- **Video-level:** H.264 compression (CRF = 30) and random frame swaps, where 25% of frames in each video clip are randomly exchanged.

Note that some of the distortion operations such as JPEG, H.264, and random frame swaps are not differentiable. Therefore, we follow prior works [5, 20, 21] and adopt a straight-through estimator (STE), where the backward pass uses $\hat{I}_{\mathcal{S}_t} + \text{stopgrad}(\mathcal{A}(\hat{I}_{\mathcal{S}_t}) - \hat{I}_{\mathcal{S}_t})$ in place of $\mathcal{A}(\hat{I}_{\mathcal{S}_t})$. This allows the approximated gradients from the loss to reach the

watermark offset parameters $\varepsilon_i(t)$ and $\delta_i(t)$.

After training, we evaluate the watermarked model by applying the original non-differentiable distortions, to assess its robustness against realistic degradation.

B. Implementation Details

All models, including ours and the baseline methods, were implemented on top of the official 4DGS framework [19]. For each scene in the DyNeRF and D-NeRF datasets, we train a separate 4DGS model (the canonical 3D Gaussians and the deformation MLP) for 14,000 and 20,000 iterations, respectively, following the hyperparameter settings of the original implementation. During the watermark embedding stage, a grid search was conducted over the following common hyperparameter ranges: learning rates for $\varepsilon_i(t)$ in $\{1.6e^{-5}, 1.6e^{-4}, 1.6e^{-3}, 1.6e^{-2}\}$, learning rates for $\delta_i(t)$ in $\{1.0e^{-5}, 1.0e^{-4}, 1.0e^{-3}, 1.0e^{-2}\}$, and λ_{recon} in $\{0, 0.5, 1, 1.5, 2.0\}$.

Our Settings. In our X-CLIP guided decoder, the message decoder $\mathcal{D}_M$ is implemented as a lightweight MLP designed to operate directly on the video–text latent embeddings produced by X-CLIP. Since these embeddings are already highly structured and discriminative, we found that a compact architecture is sufficient and provides more stable optimization than deeper alternatives. Specifically, $\mathcal{D}_M$ is implemented as a 3-layer MLP. The input is first processed by a fully connected (FC) layer with 512 hidden units followed by a GELU [7] activation. The resulting representation is then passed through an FC layer with 256 hidden units and another GELU activation. Finally, an FC layer with a Sigmoid activation maps the representation into an L-dimensional predicted message logits.

For training the decoder, we randomly generate 5,000 binary messages and split them into an 80:20 train–test ratio. This ensures that $\mathcal{D}_M$ is sufficiently exposed to diverse bit patterns during training and generalizes reliably to unseen messages. The decoder is optimized using the Adam optimizer with a learning rate of $1e^{-3}$, which we found to provide stable convergence and consistent message accuracy.

During the watermark embedding stage, a grid search was conducted over the following hyperparameter ranges: K in $\{1, 2, 3, 4, 5\}$, $\lambda_{\min}$ in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, $\lambda_{\max}$ in $\{0.6, 0.8, 1.0, 1.2, 1.4\}$, and λ_{align} in $\{0, 0.5, 1.0, 1.5, 2.0\}$.

Baseline-specific Settings. For fair comparisons, all *2D-decoder baselines* [5, 12, 18, 22, 23] use decoders that we tuned within a reasonable range of their respective hyperparameters, to obtain the most reliable message recovery. Then, the frame-level and video-level baselines in the main paper are constructed by fine-tuning the 4DGS model with our trained 2D image or video decoders. While fine-tuning the 4DGS model, the message loss $\mathcal{L}_{\text{msg}}$ and reconstruction loss $\mathcal{L}_{\text{recon}}$ are computed per rendered frame for frame-level baselines, and per video clip of $T = 8$ consecutive frames for video-level baselines. These losses directly supervise the optimization of the offset parameters $\varepsilon_i(t)$ and $\delta_i(t)$ in Eq. (6). For *3DGS watermarking strategies* [1, 8, 9], We use each method's original decoder, tuned in a similar way to ensure stable watermark recovery, and apply their strategy-specific losses with hyperparameters adjusted within a reasonable range when fine-tuning the strategy-dependent subset of Gaussian parameters in the 4DGS model.

- **CIN** [12]: Since the original paper reports results only for 30-bit and 64-bit message lengths, we retrained the decoder for 50 epochs on MS-COCO [11] dataset, with the same architecture for 32-bit and 48-bit messages to align with our evaluation settings. We tuned the watermarked image reconstruction loss weight hyperparameter $\lambda_{\text{WIm}} \in \{0.1, 0.5, 1\}$, message loss weight $\lambda_{\text{RWm}} \in \{0.001, 0.01, 0.1\}$, and $\lambda_{\text{RIm}} \in \{0, 1\}$.
- **HiDDeN** [23]: We pre-trained the HiDDeN decoder using MS-COCO [11] dataset for 300 epochs. We tuned the image reconstruction loss weight hyperparameter $\lambda_I \in \{0.1, 0.5, 1\}$ and message loss weight $\lambda_M \in \{0.1, 0.5, 1\}$. Following [4] and [9], PCA whitening is applied to the linear layer of the decoder to remove the bias of the message prediction.
- **StegaStamp** [18]: Since the original paper reports results only for a 100-bit message length, we retrained the decoder for 140,000 steps using ImageNet-1K [17] dataset, with the same architecture for 32, 48, and 64-bit messages to align with our evaluation settings. We tuned the watermarked image residual loss weight $\lambda_R \in \{0.1, 0.5, 1, 1.5, 2\}$, LPIPS perceptual image loss weight $\lambda_P \in \{1, 1.5, 2\}$, critic adversarial loss weight $\lambda_C \in \{0.1, 0.5, 1\}$, and message loss weight $\lambda_M \in \{0.5, 1, 1.5\}$.
- **SSL** [6]: Because the decoding rule of SSL is deterministic, we apply it during 4DGS fine-tuning without pre-training any additional decoder. We tuned the multi-bit message loss weight $\lambda_w \in \{1 \times 10^4, 3 \times 10^4, 5 \times 10^4\}$, contrastive loss margin $\mu \in \{1, 3, 5\}$, and false positive rate FPR $\in \{10^{-4}, 10^{-5}, 10^{-6}\}$.
- **RivaGAN** [22]: Since the official implementation of the RivaGAN decoder has the maximum capacity of 32 bits, we pre-train the decoder for 300 epochs using the Hollywood2 [14] dataset, with the same architecture for only 32-bit message. We tuned the decoded data loss weights $\lambda_d, \lambda_a \in \{0.1, 0.5, 1.0\}$ and GAN-related loss weights $\lambda_c, \lambda_r, \lambda_w \in \{0.01, 0.05, 0.1\}$.
- **VideoSeal** [5]: Following the official training protocol of VideoSeal, which consists of image pre-training followed by video fine-tuning, we additionally retrained message decoders for 32, 48, and 64-bit messages (the original paper reports only 96 and 256-bit settings), using up to 601 epochs under the same architecture. For the image pre-training stage, we trained the decoders on the SA-

Table 1. **Quantitative results.** Per-scene metrics of our method for each scene in the DyNeRF and D-NeRF datasets.

Dataset	32 bits					48 bits					64 bits				
	Video-level Bit Acc(%)	*Frame-level* Bit Acc(%)	PSNR	SSIM	LPIPS	*Video-level* Bit Acc(%)	*Frame-level* Bit Acc(%)	PSNR	SSIM	LPIPS	*Video-level* Bit Acc(%)	*Frame-level* Bit Acc(%)	PSNR	SSIM	LPIPS
DyNeRF															
cook_spinach	96.62	96.29	42.28	0.9956	0.0030	96.28	96.23	41.65	0.9946	0.0034	93.16	93.11	42.48	0.9960	0.0027
cut_roasted_beef	99.24	98.78	41.49	0.9946	0.0029	97.69	97.58	42.32	0.9956	0.0025	94.34	94.18	42.15	0.9961	0.0033
coffee_martini	97.21	96.24	41.66	0.9952	0.0032	96.34	96.24	40.60	0.9954	0.0029	94.55	94.10	40.06	0.9944	0.0031
flame_salmon	97.21	97.13	40.85	0.9945	0.0026	96.90	96.44	40.30	0.9945	0.0030	95.41	95.25	40.24	0.9941	0.0029
flame_steak	97.13	96.78	42.77	0.9959	0.0022	96.45	95.85	42.60	0.9955	0.0027	93.67	93.31	41.08	0.9960	0.0028
sear_steak	96.54	96.02	43.05	0.9960	0.0025	95.61	95.56	42.93	0.9960	0.0026	94.55	93.78	41.83	0.9956	0.0036
D-NeRF															
bouncing balls	96.88	96.88	43.03	0.9987	0.0006	91.67	90.54	40.12	0.9975	0.0020	90.40	90.78	44.65	0.9989	0.0008
hellwarrior	97.32	97.31	46.87	0.9982	0.0007	94.35	94.33	42.93	0.9957	0.0019	93.60	93.25	42.48	0.9950	0.0012
hook	98.55	98.44	42.37	0.9972	0.0012	94.35	94.33	40.94	0.9961	0.0017	92.19	92.44	41.40	0.9954	0.0016
jumpingjacks	94.20	93.63	41.87	0.9965	0.0009	89.51	88.41	41.44	0.9967	0.0008	89.29	89.25	40.19	0.9957	0.0011
lego	97.32	95.56	40.23	0.9928	0.0019	93.45	92.96	41.79	0.9940	0.0017	91.96	93.41	42.39	0.9967	0.0011
mutant	96.88	95.31	39.34	0.9949	0.0018	96.65	96.88	42.43	0.9953	0.0020	93.75	92.75	36.71	0.9908	0.0035
standup	95.09	94.38	40.08	0.9950	0.0014	93.75	92.79	36.71	0.9908	0.0035	92.41	93.31	39.90	0.9931	0.0014
trex	97.32	96.00	46.71	0.9988	0.0005	97.32	96.21	43.61	0.9983	0.0010	89.29	89.00	42.22	0.9978	0.0012

1B dataset [10], and for the subsequent video fine-tuning stage, we fine-tuned them on the SA-V dataset [16]. Across both stages, we tuned the watermark extraction loss weight $\lambda_w \in \{0.5, 1.0, 2.0\}$, the image loss weight $\lambda_i \in \{0.25, 0.5, 1.0\}$, and the quality discriminator loss weight $\lambda_{disc} \in \{0.05, 0.1, 0.15\}$.

- **GaussianMarker** [8]: We tuned the uncertainty threshold factor $f \in \{0.002, 0.13, 0.24, 1.0, 3.7\}$, 2D message loss weight $\lambda_{msg} \in \{0.1, 0.5, 1\}$, and 2D rendered image reconstruction loss weight $\lambda_{rec} \in \{0.1, 0.5, 1\}$, 3D message loss weight $\lambda'_{msg} \in \{0.1, 0.5, 1\}$, and 3D adversarial loss weight $\lambda_{adv} \in \{0.1, 0.5, 1\}$.
- **GuardSplat** [1]: For the CLIP-guided decoder, we trained the decoder for 100 epochs, with the same architecture and hyperparameters as the official implementation. For the 4DGS model fine-tuning, we tuned the message loss weight $\lambda_{msg} \in \{0.1, 0.5, 1\}$, image reconstruction loss weight $\lambda_{recon} \in \{0.1, 0.5, 1\}$, and SH coefficients offset regularization loss weight $\lambda_{off} \in \{1, 5, 10\}$.
- **3D-GSW** [9]: For the 2D image decoder, we used the pre-trained HiDDeN decoder shared in their official implementation. For the 4DGS model fine-tuning, we tuned the message loss weight $\lambda_m \in \{0.1, 0.5, 1\}$, image residual loss weight $\lambda_{rec} \in \{0.1, 0.5, 1\}$, image perceptual loss weight $\lambda_{lpips} \in \{0.2, 0.6, 1\}$, wavelet-subband loss weight $\lambda_w \in \{0, 0.3, 0.7, 1\}$, pruning threshold $\in \{10^{-9}, 10^{-8}, 10^{-7}\}$, patch size $|P| \in \{4, 8, 16\}$, patch selection ratio $K \in \{0.01, 0.03, 0.05\}$, and mask gradient manipulation strength $\beta \in \{2, 4, 6\}$.

C. Additional Experimental Results

C.1. Full Quantitative Results for Each Dataset

To complement the averaged metrics in Tab. 5 of the main paper, we provide the full per-scene results of our method for both DyNeRF and D-NeRF scenes in Tab. 1. Across every scene and setting, the method consistently delivers high bit accuracy and strong visual fidelity, demonstrating its robustness under diverse motions and appearance variations.

C.2. Ablation Study on Offset Parameter Choices

In Section 4.1 of the main paper, we defined the watermarked 4D Gaussian representation by introducing learnable time-varying offsets only on the Gaussian positions $x_i(t)$ and SH coefficients $h_i(t)$, while keeping the opacity $\alpha_i(t)$ and covariance $\Sigma_i(t)$ unchanged. To more thoroughly examine this design choice, we consider an extended setting in which offsets may also be applied to $\alpha_i(t)$ and $\Sigma_i(t)$. For this extended formulation, we denote the learnable opacity offset by $\gamma_i(t)$ and the covariance offset by $\Lambda_i(t)$, allowing us to analyze how perturbing these additional parameters influences the behavior and stability of the resulting 4D representation.

Across the different offset configurations presented in Tab. 2, the setting that enables only the positional and SH offsets $\varepsilon_i(t)$ and $\delta_i(t)$ yields the best overall performance in both bit accuracy and PSNR. Introducing opacity offsets $\gamma_i(t)$ noticeably reduces stability and degrades reconstruction quality, while activating covariance offsets $\Lambda_i(t)$ leads to even more pronounced degradation due to their strong influence on Gaussian shape and scale. The full extended configuration, in which all offsets are enabled simultaneously, exhibits the largest drop in fidelity, confirming that perturbations to $\alpha_i(t)$ and $\Sigma_i(t)$ are unsuitable for reliable watermark embedding. These results validate our design choice in the main paper, where offsets are restricted to $\mathbf{x}_i(t)$ and $h_i(t)$ to avoid geometry distortion and ensure imperceptible watermark embedding.

Table 2. Ablation on different offset types in the extended formulation at $L = 48$ bits. Results are averaged over D-NeRF and DyNeRF datasets.

Offset Types				Bit Acc (%) ↑	PSNR ↑
$\varepsilon_i(t)$	$\delta_i(t)$	$\gamma_i(t)$	$\Lambda_i(t)$		
✓	✓	×	×	**95.02**	**41.45**
✓	✓	✓	×	94.90	36.88
✓	✓	×	✓	94.00	26.04
✓	✓	✓	✓	94.83	24.99

C.3. Ablation Study on Temporal Offset Parameterization Strategy

To assess how different temporal formulations of the $\varepsilon_i(t)$ influence the resulting watermarking performance, we compare three parameterization strategies, including the Fourier series based representation introduced in Eq. (15) in the main paper. As baselines, we first consider a non-parameterized variant, where the offset at each timestamp is learned independently. We additionally evaluate a polynomial parameterization that enforces smooth variation over time by expressing the offset as a low-order function of t,

$$\varepsilon_i(t) = \sum_{k=0}^{K-1} p_{i,k} t^k, \tag{1}$$

where $p_{i,k} \in \mathbb{R}^3$ is learnable coefficient of a polynomial. For a fair comparison with the Fourier series parameterization in Eq. (15), which uses six learnable coefficient vectors in total, we set $K = 6$ for this polynomial variant.

As shown in Tab. 3, the Fourier series parameterization achieves the strongest overall performance, outperforming both the polynomial and non-parameterized settings in bit accuracy and visual fidelity. This reflects that Fourier series parameterization can effectively represent the watermark offsets needed to follow the Gaussians' more complex motion trajectories using only a small number of frequency components, while preserving smooth temporal behavior.

Table 3. Ablation on different temporal offset parameterization strategies at $L = 48$ bits. Results are averaged over D-NeRF and DyNeRF datasets.

Parameterization Type	Bit Acc (%) ↑	PSNR ↑
None	92.79	37.05
Polynomial	94.44	40.20
Fourier	**95.02**	**41.45**

C.4. Ablation Study on Motion-Adaptive Loss Weighting Strategy

To investigate alternative ways of assigning adaptive message loss weights across temporal windows, we compare it against several alternative strategies that assign different supervision strengths to each temporal window $\mathcal{S}_t$. As a naive baseline, we first consider uniform weighting, where a constant weight λ_{max} is applied across all timestamps. We then evaluate a threshold-based strategy that assigns either the lower or upper bound of the weighting range depending on whether the normalized motion magnitude β_{S_t} exceeds a predefined threshold,

$$\lambda_{S_t}^{\text{th}} = \begin{cases} \lambda_{\max}, & \beta_{S_t} > 0.5, \\ \lambda_{\min}, & \text{otherwise.} \end{cases}$$

Finally, we include an exponential motion weighting scheme that generalizes our formulation by raising the motion magnitude to an exponent $n \in \mathbb{R}^+$,

$$\lambda_{S_t}^{(n)} = \left(1 - \beta_{S_t}^n\right) \lambda_{\min} + \beta_{S_t}^n \lambda_{\max},$$

which allows the supervision to scale nonlinearly with respect to the motion intensity.

The results for all loss weighting strategies specified above are provided in Tab. 4. Overall, linear motion-adaptive weighting yields the best trade-off between watermark recoverability and visual fidelity.

Table 4. Ablation on different motion-adaptive loss weighting strategies at $L = 48$ bits. Results are averaged over D-NeRF and DyNeRF datasets.

Strategy	Bit Acc (%) ↑	PSNR ↑
Uniform	92.94	39.88
Thresholding	93.47	40.15
Exponential ($n = 0.5$)	94.63	40.13
Exponential ($n = 2$)	94.35	40.81
Ours	**95.02**	**41.45**

C.5. Discussion on Model Complexity

To compare the computational cost of our method and the baselines, Tab. 5 reports the overall training time and storage memory for baselines and our method on the *flame_salmon* scene in the DyNeRF dataset with a 64-bit message, as this scene contains the largest number of Gaussians and thus represents the most computationally demanding case. With the compact Fourier series offset parameterization, we achieve competitive trained model storage compared to the base pretrained 4DGS model. Also, we have competitive training time compared to GuardSplat, which has the fastest training time among frame-level baselines processing one frame at a single iteration.

Table 5. Comparison of total training time and storage (MB) across methods. Values in parentheses indicate the storage difference compared to the base pretrained 4DGS model.

Model	Training Time	Storage (MB)
4DGS+SSL	23m 57s	543.21 (+470.62)
GuardSplat	8m 25s	96.42 (+23.83)
3D-GSW	24m 51s	68.38 (-4.21)
4DGS+VideoSeal	19m 30s	543.21 (+470.62)
Ours	12m 48s	108.34 (+35.75)

C.6. Sensitivity Analysis

We present the results of the sensitivity analysis with respect to bit accuracy and PSNR across different values of hyperparameters in Fig. 1.

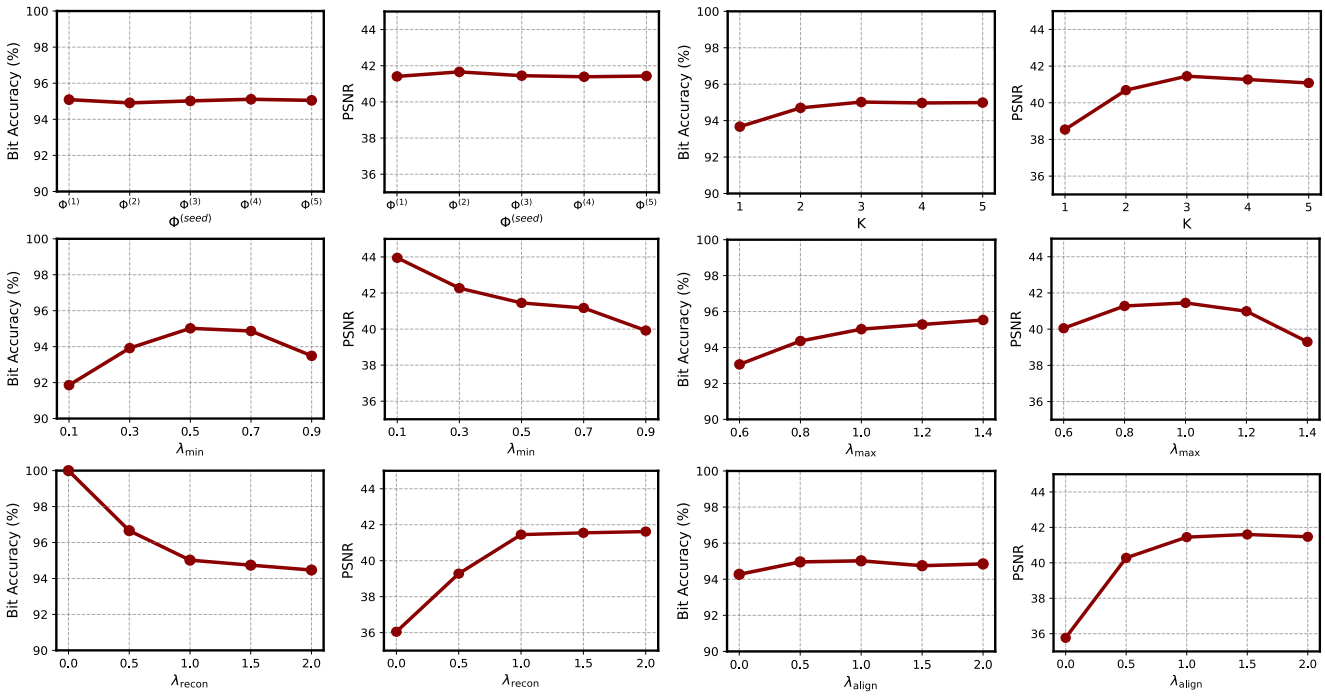

Figure 1. **Hyperparameter sensitivity analysis at $L = 48$ bits.** Plots reporting bit accuracy and PSNR as each hyperparameter is varied across its range. Results are averaged over D-NeRF and DyNeRF datasets.

Sensitivity to the random initialization of Φ. As discussed in Section A.1, Φ is randomly initialized once at the beginning. Changing this random seed does not noticeably affect the performance of our method, as we use the X-CLIP's well-structured and discriminative latent space primarily for efficient mapping during decoder training, rather than relying on any specific semantic meaning of individual latent tokens. Therefore, randomness in the tokenization rule has little influence on the overall behavior of the decoder and the performance of our method.

Sensitivity to the number of Fourier series bases K. We observe that bit accuracy and PSNR are highest when $K = 3$. If K is small, the offset trajectory becomes overly simple, reducing the degrees of freedom available for watermark embedding and lowering bit accuracy, while also making it harder to align with complex Gaussian motion, which harms the visual quality. Thus, $K = 3$ provides a good balance between expressiveness and smoothness.

Sensitivity to λ_{min}. In this experiment, we fix $\lambda_{\text{max}} = 1.0$ and vary λ_{min}. Among the tested values, $\lambda_{\text{min}} = 0.5$ offers the best overall balance. When λ_{min} is small, the average watermark supervision becomes weak, reducing bit accuracy while increasing PSNR because the embedding strength is insufficient. When λ_{min} approaches λ_{max}, the adaptive weighting scheme collapses into a uniform loss weighting, which degrades PSNR.

Sensitivity to λ_{max}. Here, we fix $\lambda_{\text{min}} = 0.5$ and vary λ_{max}. The optimal balance is obtained at $\lambda_{\text{max}} = 1.0$. If λ_{max} is small, the maximum watermark supervision is insufficient, reducing bit accuracy. If λ_{max} is large, bit accuracy increases but PSNR decreases due to the trade-off between embedding strength and visual fidelity.

Sensitivity to λ_{recon}. When $\lambda_{\text{recon}} = 0$, PSNR drops significantly while bit accuracy reaches nearly 100%. As λ_{recon} increases, bit accuracy gradually decreases and PSNR improves, which reflects the direct trade-off between reconstruction fidelity and watermark embedding strength.

Sensitivity to λ_{align}. We find that $\lambda_{\text{align}} = 1.0$ yields the best overall balance. Bit accuracy remains relatively stable across values, but PSNR drops substantially when $\lambda_{\text{align}} = 0$. This indicates that the trajectory alignment loss is crucial for preserving visual fidelity.

D. Additional Qualitative Results

We provide additional qualitative results for our method on the DyNeRF and D-NeRF datasets in Fig. 3 and Fig. 4. We also include sequence visualizations and temporal consistency measurements in Fig. 2 to highlight the improved temporal stability of our approach. For temporal consistency evaluation, we adopt the **tLP** and **tOF** metrics from [2], where **tLP** captures temporal perceptual variation and **tOF** measures differences in motion estimated from con-

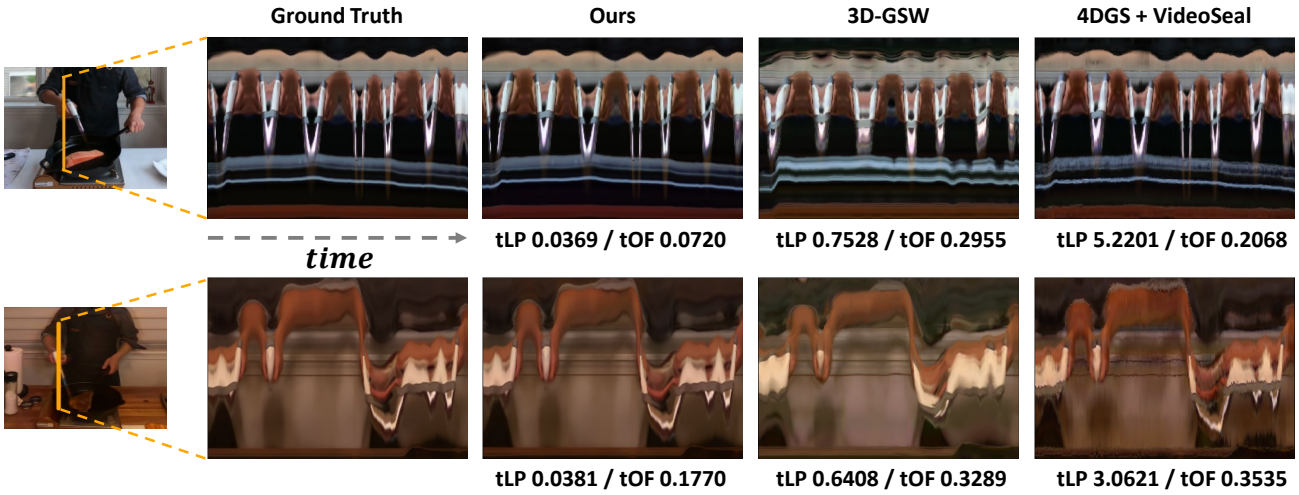

Figure 2. **Comparisons of temporal consistency.** Visualizations of the results at 48 bits on the DyNeRF dataset (Top: *flame_salmon*, Bottom: *flame_steak*) to show temporal consistency of each method. For these visualizations, we extract a vertical column of 310 pixels from each rendered frame at test camera and concatenate these columns across 300 consecutive frames, producing a height–time image. Also, the corresponding temporal consistency metrics (tLP, tOF) are marked under each of the visualization. Our method shows temporal consistency with significantly fewer temporal artifacts and geometric inconsistencies than the baselines.

secutive frames. Each metric is computed as:

$$\begin{aligned} \mathbf{tLP} &= \left\| LP(\hat{I}_{t-1}, \hat{I}_t) - LP(I_{t-1}, I_t) \right\|_1, \\ \mathbf{tOF} &= \left\| OF(\hat{I}_{t-1}, \hat{I}_t) - OF(I_{t-1}, I_t) \right\|_1, \end{aligned} \tag{2}$$

where I_t and $\hat{I}_t$ denote the ground-truth frame and the rendered frame from the watermarked 4DGS model, respectively, and LP and OF denote the LPIPS metric and the Farneback [3] optical flow estimator. As shown in the figure, our method exhibits the strongest temporal consistency compared with the frame-level baseline 3D-GSW and the video-level baseline VideoSeal.

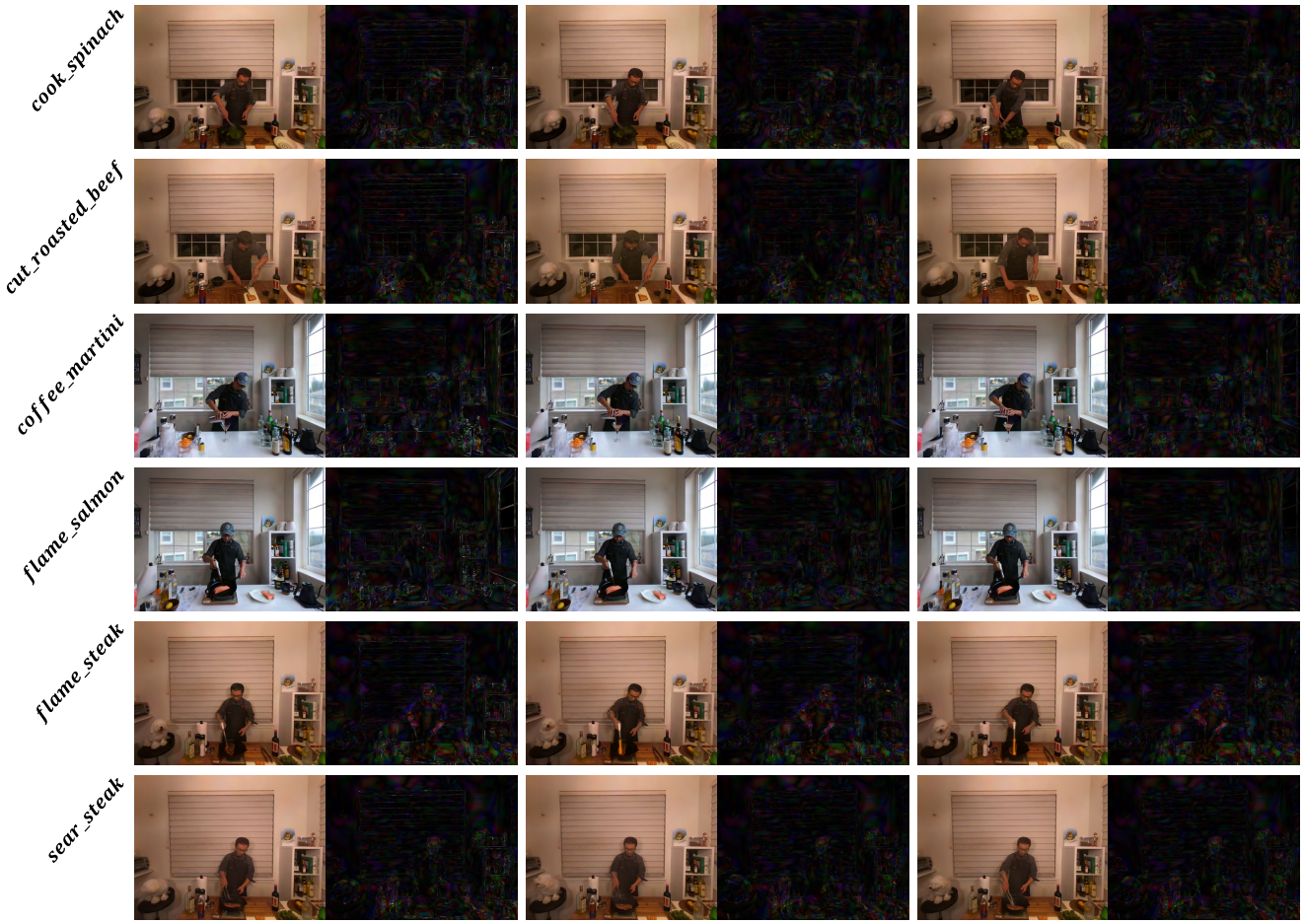

Figure 3. **Additional qualitative results on DyNeRF dataset.** Visualizations of rendered sequences at 48 bits on the DyNeRF dataset. For each frame of the sequence, the image on the right shows the ground-truth residual, magnified by a factor of $\times 10$.

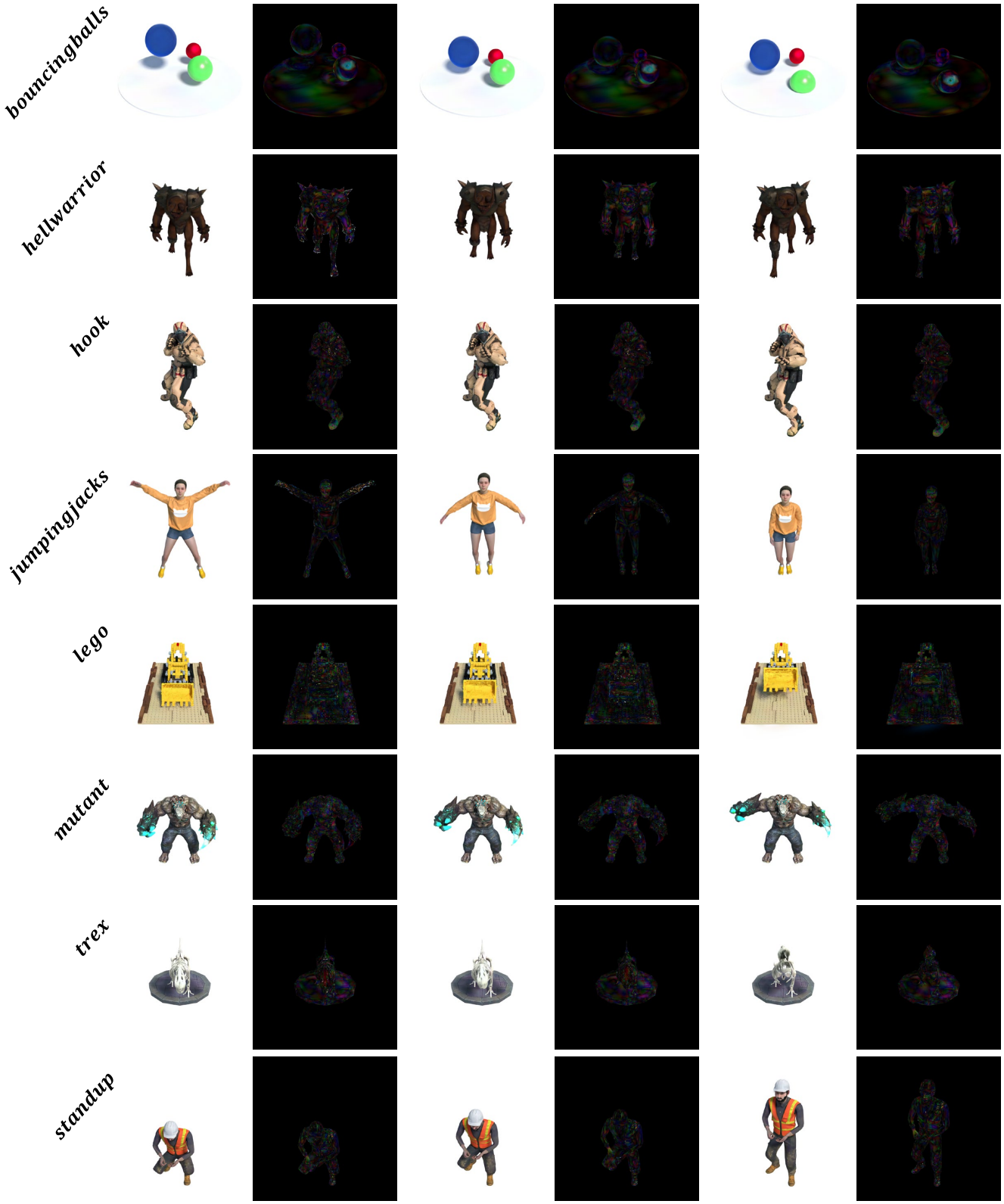

Figure 4. **Additional qualitative results on D-NeRF dataset.** Visualizations of rendered sequences at 48 bits on D-NeRF dataset. For each frame of the sequence, the image on the right shows the ground-truth residual, magnified by a factor of $\times 10$.

References

[1] Chen, Z., Wang, G., et al.: Guardsplat: Efficient and robust watermarking for 3d gaussian splatting. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2025) 2, 3

[2] Chu, M., Xie, Y., et al.: Learning temporal coherence via self-supervision for gan-based video generation. ACM Transactions on Graphics (TOG) (2020) 5

[3] Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Scandinavian conference on Image analysis (2003) 6

[4] Fernandez, P., Couairon, G., et al.: The stable signature: Rooting watermarks in latent diffusion models. In: IEEE/CVF International Conference on Computer Vision (2023) 2

[5] Fernandez, P., Elsahar, H., et al.: Video seal: Open and efficient video watermarking. arXiv preprint arXiv:2412.09492 (2024) 1, 2

[6] Fernandez, P., Sablayrolles, A., et al.: Watermarking images in self-supervised latent spaces. In: IEEE International Conference on Acoustics, Speech and Signal Processing (2022) 2

[7] Hendrycks, D.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016) 2

[8] Huang, X., Li, R., et al.: Gaussianmarker: Uncertainty-aware copyright protection of 3d gaussian splatting. Advances in Neural Information Processing Systems (2024) 2, 3

[9] Jang, Y., Park, H., et al.: 3d-gsw: 3d gaussian splatting for robust watermarking. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5938–5948 (2025) 2, 3

[10] Kirillov, A., Mintun, E., et al.: Segment anything. In: Proceedings of the IEEE/CVF international conference on computer vision (2023) 3

[11] Lin, T.Y., Maire, M., et al.: Microsoft coco: Common objects in context. In: European conference on computer vision (2014) 2

[12] Ma, R., Guo, M., et al.: Towards blind watermarking: Combining invertible and non-invertible mechanisms. In: ACM International Conference on Multimedia (2022) 2

[13] Ma, Y., Xu, G., et al.: X-clip: End-to-end multi-grained contrastive learning for video-text retrieval. In: ACM International Conference on Multimedia (2022) 1

[14] Marszalek, M., Laptev, I., et al.: Actions in context. In: 2009 IEEE conference on computer vision and pattern recognition (2009) 2

[15] Radford, A., Kim, J.W., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning (2021) 1

[16] Ravi, N., Gabeur, V., et al.: Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714 (2024) 3

[17] Russakovsky, O., Deng, J., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision (2015) 2

[18] Tancik, M., Mildenhall, B., Ng, R.: Stegastamp: Invisible hyperlinks in physical photographs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020) 2

[19] Wu, G., Yi, T., et al.: 4d gaussian splatting for real-time dynamic scene rendering. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024) 2

[20] Ye, G., Gao, J., et al.: Itov: efficiently adapting deep learning-based image watermarking to video watermarking. In: International Conference on Culture-Oriented Science and Technology (2023) 1

[21] Zhang, C., Karjauv, A., et al.: Towards robust deep hiding under non-differentiable distortions for practical blind watermarking. In: ACM International Conference on Multimedia (2021) 1

[22] Zhang, K.A., Xu, L., et al.: Robust invisible video watermarking with attention. arXiv preprint arXiv:1909.01285 (2019) 2

[23] Zhu, J., Kaplan, R., et al.: Hidden: Hiding data with deep networks. In: European Conference on Computer Vision (2018) 2