

RiGS: Rigid-aware 4D Gaussian Splatting from a Single Monocular Video

Supplementary Material

Table 4. Hyper Parameters

Parameter	Value	Parameter	Value
λ_{ssim}	0.1	lr_{μ}	0.00016
λ_{alpha}	0.5	lr_{s}	0.005
λ_{depth}	0.05	lr_{q}	0.001
λ_{normal}	0.05	lr_{o}	0.05
λ_{track}	2.0	lr_{c}	0.01
λ_{flow}	0.01	lr_{β}	0.001
λ_{β}	0.5	lr_{γ}	0.001
λ_{s}	0.5	lr_{w}	0.01
—	—	lr_{T}	0.0001

6. Additional Implementation Details

Object-Wise Dynamic Masks. As shown in Eq. 5, we compute motion scores by combining the flow-based weights w_t with the Sampson error [27]. To obtain w_t , we use the flow uncertainty $u_t \in \mathbb{R}^+$ estimated by SEARAFT [50] together with the occlusion mask m_t^{occ} from a forward-backward consistency check [36]:

$$w_t = \frac{1 - m_t^{\text{occ}}}{(1 + u_t)^2}. \quad (18)$$

The Sampson error is given by

$$e = \frac{|\mathbf{x}_l^T \mathbf{F} \mathbf{x}_r|}{\sqrt{\|\mathbf{F} \mathbf{x}_l\|_2^2 + \|\mathbf{F} \mathbf{x}_r\|_2^2}}, \quad (19)$$

where \mathbf{x}_l and \mathbf{x}_r are sampled from the non-occluded regions $\neg m^{\text{occ}}$. The fundamental matrix \mathbf{F} is estimated using the LMEDS (Least Median of Squares) method based on 10,000 sampled correspondences for efficiency.

For object mask generation, we apply SAM [25, 44] for automatic video object segmentation. To detect objects that do not appear in the first frame, we modify AutoSeg to enable fast discovery of new object instances. For temporal filtering, we set $\epsilon^{\text{temp}} = 1e-4$ as a conservative threshold for static objects. Since different videos exhibit varying FPS and motion magnitudes, the dynamic threshold ϵ^{dyn} is adaptively set to $\max(s_i)/4$.

Hyper-Parameters We present additional hyper-parameters in Table 4. All kinds of Gaussians share the same set of learning rates. For the Nvidia dataset, we use 3K iterations for static Gaussian warm-up and 12K iterations for rigid Gaussian warm-up, while for the DyCheck iPhone dataset, these values are set to 5K and 40K, respectively.

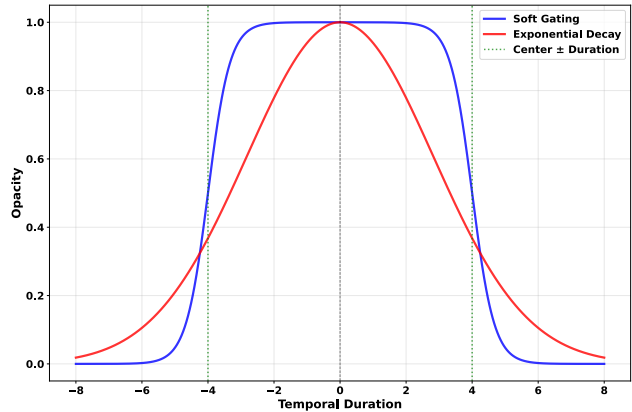


Figure 6. Comparison of Soft Gating and Exponential Decay Functions

Opacity Decay While prior works [35, 51, 58] use an exponential distribution for opacity decay, we employ a soft gating function due to its sharper temporal boundary. We compare the two functions in Figure 6. As shown in the figure, the soft gating function maintains a flat response near its peak, whereas the exponential decay decreases gradually. We also experiment with replacing our soft gating function with the exponential form and observe that it leads to the emergence of more short-term Gaussians. We attribute this behavior to the soft boundary of the exponential decay when the temporal duration becomes large.

7. Two-peak Pattern

To verify that the observed two-peak pattern is not tied to a particular scene, we further sample sequences from both the Nvidia dataset [12] and DAVIS [20, 23]. As shown in Figure 7, transient Gaussians predominantly correspond to fast or complex motions, whereas rigid Gaussians align with more stable, consistent motions.

We attribute this separation to the regularization term in Eq. 17. When certain Gaussians are occluded in the training frames, the regularization encourages them to remain in the scene and follow the motion trajectories represented by the motion bases. In such cases, their temporal duration β^r tends to expand, as these Gaussians continue to exhibit coherent motion over time.

Conversely, in regions with fast or highly nonlinear motion, the limited expressiveness of the motion bases causes rigid Gaussians to drift into unwanted areas in other frames, making them appear as artifacts. The temporal duration penalty suppresses these unstable Gaussians by reducing

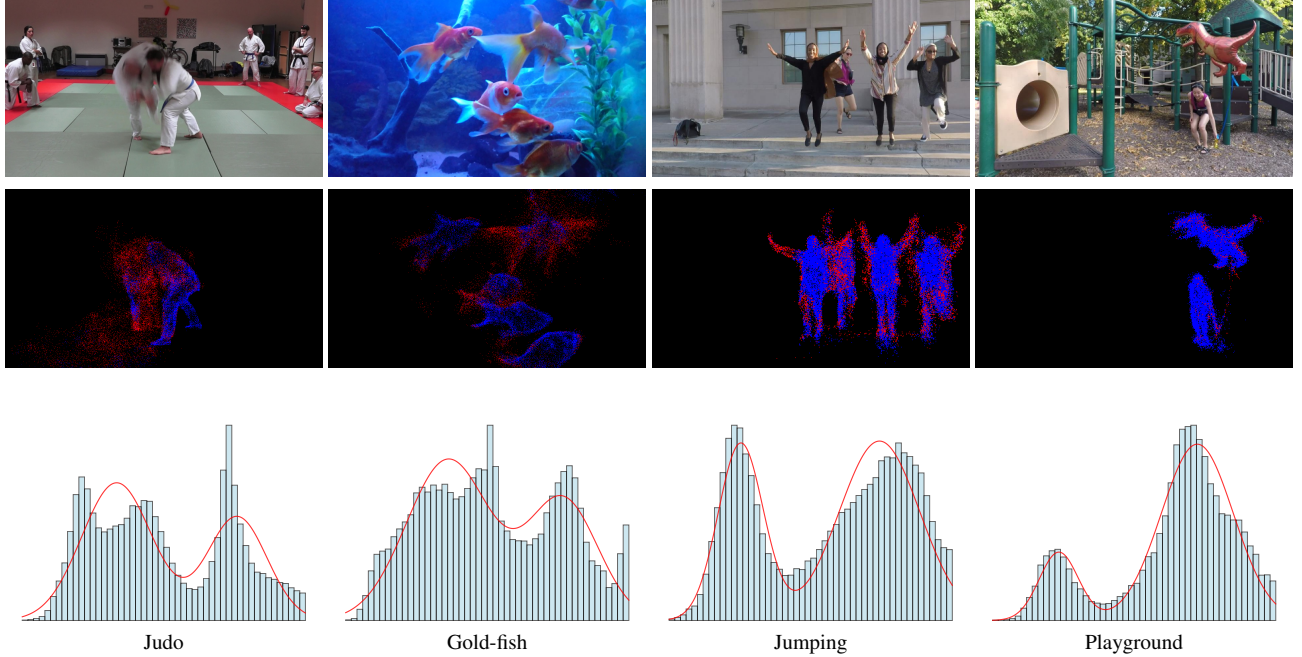


Figure 7. **Distribution of Temporal Duration.** We visualize the distribution of the temporal duration β^r . The first row shows the original image of the scene. The second row visualizes the canonical points of the rigid Gaussians (blue) and transient Gaussians (red). The third row presents the corresponding statistical distribution.

Table 5. Runtime Profile

Step	Time(s)↓
Optical Flow Estimation	1.763
Video Object Segmentation	23.151
Sampson Error Calculation	8.308
Mask Composition	0.911
All	34.133

Table 6. Evaluation on DAVIS 2016 Dataset

Methods	IoU↑	Time(s)↓
RoMo [14]	65.9	552.7
SegAnyMo [21]	85.3	176.3
Ours	81.9	34.1

their β^r , effectively turning them into transient Gaussians. Meanwhile, for rigid Gaussians that correctly model stable or occluded regions, the regularization reinforces long-term consistency and preserves large temporal durations.

Together, these opposing behaviors naturally induce the two-peak temporal-duration distribution, with one peak corresponding to transient Gaussians that model rapid or complex motion, and the other to rigid Gaussians that capture stable and persistent motion patterns.



Figure 8. **Qualitative Comparison on DAVIS.** SegAnyMo yields significantly worse results when its predicted motion points fall into background regions. Our method avoids this issue by aggregating motion information within pre-segmented objects and filtering out those with low motion scores.

8. Dynamic Mask

Dynamic Mask Evaluation. To demonstrate the effectiveness of our dynamic mask segmentation method, we further evaluate it on the DAVIS dataset [20] and compare it with recent approaches [14, 21]. We report both **IoU** and **runtime**, averaged across all scenes. As shown in Table 6, our method achieves higher segmentation accuracy than RoMo, while running substantially faster than both SegAnyMo and RoMo. This highlights the advantage of using object-wise motion cues: our method avoids spurious motion responses in background regions and produces cleaner motion segmentation. Moreover, as illustrated in Figure 8, our approach exhibits stronger robustness than SegAnyMo, particularly in scenes where motion prediction becomes noisy or ambiguous.

We additionally provide a breakdown of the runtime in Table 5. The main computational bottleneck arises from

video object segmentation, especially in scenes where a large number of objects are identified by SAM [25]. Nevertheless, all other components remain lightweight, and the overall runtime is still significantly lower than recent alternatives.

Object-wise Over-marking. Our object-wise masking design mitigates over-marking in two key ways: (1) Motion averaging⁴ ensures that localized motions are diluted by the dominant static parts of an object, preventing spurious dynamic labeling. (2) The segmentation threshold $\max(s_i)/4$ prioritizes major dynamic objects and avoids over-segmentation of minor movements. We ablate the over-marking behavior on the *Balloon2* scene. As shown in Table 7, the over-allocation slightly increases PSNR (28.25→28.45), suggesting that over-marking can be beneficial for reconstructing ambiguous scenes.

Table 7. Quantitative comparison of over-marking on *Balloon2*.

Mask	PSNR	Mem (GB)	#Static GS	#Dynamic GS
No Over-mark	28.25	0.71	290,789	24,959
Over-mark	28.45	0.66	261,218	35,152

Quantitative Impact of Masking Strategy. To isolate the contribution of our masking strategy from the Gaussian representation, we evaluate our method using different mask types. Following RoDynRF [33], we compute Sampson-based masks (denoted *Ours-RoDynRF*). Following MoSca [28], we further densify the masks using 2D tracks (denoted *Ours-MoSca*). Table 8 demonstrates the effectiveness of our Gaussian representation.

Table 8. Comparison of masking strategies and optimization.

	RoDynRF	Ours-RoDynRF	MoSca	Ours-MoSca	Ours
PSNR	25.89	26.00	26.72	27.33	27.43

9. Sensitivity Studies

Sensitivity study on β_r . We conduct a sensitivity study by varying β_r on the Nvidia dynamic scene dataset. As shown in Table 9, we vary β_r from 1 to 10. Performance remains within ± 0.15 dB of the optimum, demonstrating the robustness of our method to this threshold.

Table 9. Sensitivity study on β_r .

	$\beta_r = 1$	$\beta_r = 2$	$\beta_r = 4$	$\beta_r = 7$	$\beta_r = 10$
PSNR	27.34	27.43	27.49	27.46	27.37

Robustness to Depth Prior Models. Since 4D reconstruction from monocular video is an ill-posed problem, it is

common practice to use prior models. We mitigate errors from depth priors using a Laplacian edge filter, from optical flow using occlusion masks, and from scene flow using an intersection mask. To further demonstrate robustness, we test different depth models on the Nvidia dynamic scene dataset. As shown in Table 10, the PSNR variation is small (± 0.21 dB), confirming that our method is not critically sensitive to the specific depth prior.

Table 10. Comparison of different depth prior models.

	Metric3D-v2	MoGe	UniDepth	DepthPro
PSNR	27.43	27.49	27.28	27.31

10. Training and Inference Comparison

We compare training and inference costs on the DyCheck dataset in Table 11. We include results using both our default iteration count and a reduced 45K iteration setting.

11. More Results

We summarize the training statistics in Table 12. We further report per-scene metrics on the DyCheck iPhone dataset in Table 13 for a more detailed evaluation.

Table 11. Training and inference comparison on DyCheck dataset.

Method	PSNR	Train. Time	Infer. FPS	Infer. Mem
SoM	17.32	2hrs	144	1.2GB
MoSca	19.32	0.78hrs	38	1.3GB
Ours	19.50	1.8hrs	132	1.0GB
Ours-45K	19.44	0.85hrs	196	1.0GB

Table 12. Training Statistics

Scene	VRAM (GB)	Time (h)	#Static GS	#Dynamic GS	#Total GS
DyCheck iPhone Dataset					
Apple	1.68	1.69	288,066	23,998	312,064
Block	6.63	1.89	136,123	110,445	246,568
Paper-windmill	0.66	1.65	387,559	24,845	412,404
Space-out	1.90	1.80	135,605	47,986	183,591
Spin	2.09	1.80	283,770	31,010	314,780
Teddy	9.12	2.30	101,689	197,523	299,212
Wheel	3.73	1.54	216,963	86,172	303,135
Average	3.69	1.81	221,396	74,568	295,965
Nvidia Dynamic Scene Dataset					
Balloon1	3.02	0.29	259,834	66,593	326,427
Balloon2	0.71	0.29	290,789	24,959	315,748
Jumping	7.92	0.36	240,412	77,244	317,656
Truck	4.40	0.33	285,384	48,359	333,743
Skating	0.50	0.29	254,010	17,098	271,108
Umbrella	3.34	0.31	414,920	34,320	449,240
Playground	1.05	0.31	370,305	30,399	400,704
Average	2.99	0.31	302,236	42,710	344,947

Table 13. Quantitative comparison on DyCheck dataset.

Method	Apple			Block			Paper-windmill			Space-out		
	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓
Dyn. Gaussians	7.65	–	0.766	7.55	–	0.684	6.24	–	0.729	6.79	–	0.733
4DGS	15.41	–	0.456	11.28	–	0.633	15.60	–	0.297	14.60	–	0.372
Gaussian Marbles	17.70	–	0.492	17.42	–	0.384	17.04	–	0.394	15.94	–	0.435
SoM	18.57	0.771	0.341	17.41	0.644	0.323	18.14	0.415	0.225	16.85	0.601	0.324
MoSca	19.40	0.810	0.340	18.06	0.680	0.330	22.34	0.740	0.150	20.48	0.660	0.260
Ours	19.67	0.807	0.301	18.62	0.676	0.285	22.20	0.728	0.134	20.43	0.680	0.243

Method	Spin			Teddy			Wheel			AVE		
	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓	mPSNR↑	mSSIM↑	mLPIPS↓
Dyn. Gaussians	8.08	–	0.651	7.41	–	0.690	7.28	–	0.593	7.29	–	0.692
4DGS	14.42	–	0.339	12.36	–	0.466	11.79	–	0.436	13.64	–	0.428
Gaussian Marbles	18.88	–	0.428	13.95	–	0.442	16.14	–	0.351	16.72	–	0.418
SoM	19.35	0.582	0.247	13.69	0.542	0.380	17.21	0.628	0.230	17.32	0.598	0.296
MoSca	21.31	0.750	0.190	15.47	0.620	0.350	18.17	0.680	0.230	19.32	0.706	0.264
Ours	20.92	0.710	0.211	16.20	0.639	0.303	18.45	0.692	0.205	19.50	0.705	0.240