

4D Gaussian Splatting with Motion-Aware Frame Selection

Members: 曾紹嶂、林悅揚

Advisor: 陳冠文 教授

[GitHub](#)

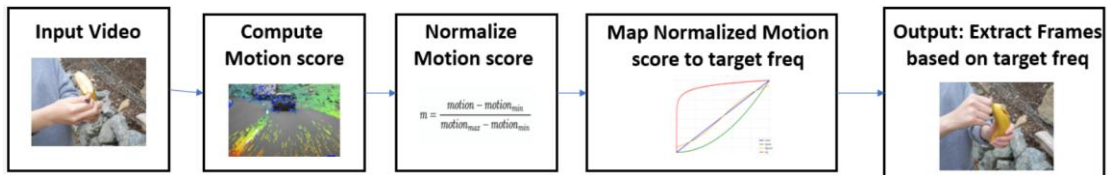
Introduction

3DGS[1] is an efficient rendering method that represents static scenes using transparent 3D Gaussians. Its extension, 4DGS [2], enables real-time rendering of dynamic scenes.

However, 4DGS employs a **fixed sampling frequency**, which can lead to over-sampling in slow-motion videos and result in inefficient training. Moreover, the overall pipeline—including COLMAP reconstruction and training—remains **time-consuming**.

To address this, we propose adjusting the sampling frequency based on **motion scores**, making the process more **efficient** and accessible while preserving the quality of 4DGS results.

Method



1. Compute Motion Scores via Optical Flow :

For each frame pair t and $t+1$, dense optical flow is computed via Farneback's algorithm to obtain per-pixel displacement vectors $(u(x, y), v(x, y))$.

The motion score for frame t is then defined as the average magnitude of all flow vectors:

$$\text{MotionScore}(t) = \frac{1}{N} \sum_{x,y} \sqrt{u(x,y)^2 + v(x,y)^2}$$

For each segment (e.g., 1 second), the segment-level motion score is defined as the average of motion scores across all frames within the segment.

2. Normalize Motion Scores :

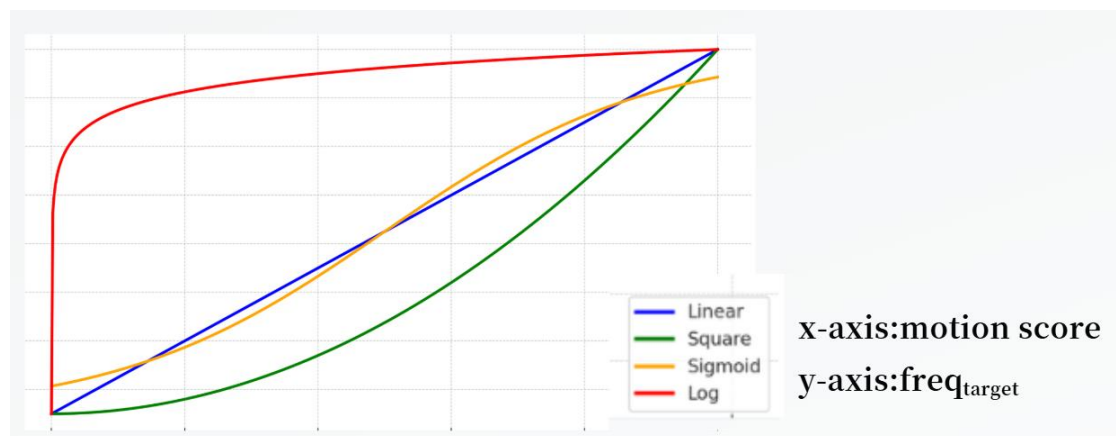
$$m = \frac{motion - motion_{min}}{motion_{max} - motion_{min}}$$

3. Map Normalized Motion to Target Frequency:

We use 4 different mapping method

- Linear : $freq_{target} = m \cdot (freq_{max} - freq_{min}) + freq_{min}$
- Square : $freq_{target} = m^2 \cdot (freq_{max} - freq_{min}) + freq_{min}$
- Sigmoid : $freq_{target} = (1 / (1 + e^{-5(m - 0.5)})) - 1 \cdot (freq_{max} - freq_{min}) + freq_{min}$
- log : $freq_{target} = \text{Take } \log(m + \epsilon), \text{ normalize again, then map to } freq_{target}$

The mapping graph



4. Extract Frames Based on Target Frequency:

- Looks up the $freq_{target}$ assigned to this segment
 $freq_{target} = \text{segment_freq}[\text{segment_idx}]$
- Track time using $\text{next_save_time} += 1 / freq_{target}$
- Save the frame when $\text{curr_time_sec} \geq \text{next_save_time}$

Evaluation Metrics

We evaluate our results using three metrics:

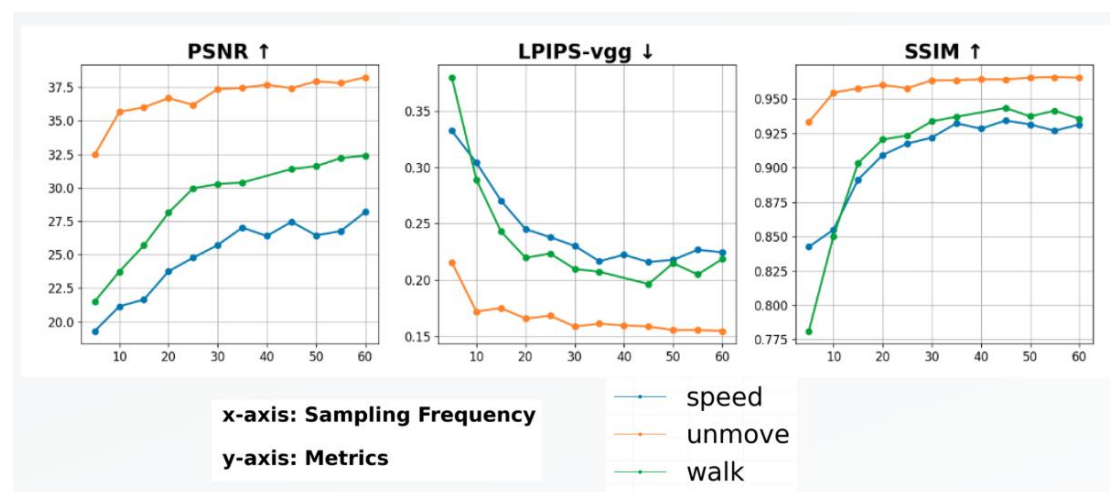
- PSNR↑: Measures **pixel-level** error between two images.
- LPIPS-vgg↓: Uses a deep model (VGG) to extract features and compares

the distance in **feature space**.

- **SSIM↑**: Compares **structural similarity**, including luminance, contrast, and structural components between two images.

Experiment

We assume that when the motion intensity exceeds a certain threshold, increasing the sampling frequency further will yield similar results. To test this, we created conditions with varying motion intensities. [video](#)



Based on our observations, the assumption appears to be correct.

We further apply our method on [HyperNeRF Datasets](#). [video](#)

P.S. Quantitative results on the synthesis dataset. The best and the second best results are denoted by **yellow** and **pink**, LPIPS is LPIPS-vgg.

Method	Banana(17s)			
	PSNR ↑	LPIPS ↓	SSIM ↑	Times ↓
Baseline	29.11	0.16	0.88	167mins
Liner	28.37	0.17	0.87	117mins
Square	27.55	0.18	0.85	127mins
Sigmoid	28.30	0.16	0.87	117mins
Log	28.83	0.16	0.88	152mins

Method	3D Printer(6s)				Broom(6s)			
	PSNR ↑	LPIPS ↓	SSIM ↑	Times ↓	PSNR ↑	LPIPS ↓	SSIM ↑	Times ↓
Baseline	23.66	0.18	0.80	89mins	21.85	0.40	0.51	85mins
Liner	22.51	0.20	0.77	68mins	20.49	0.43	0.41	69mins
Square	22.68	0.20	0.78	66mins	20.60	0.42	0.44	81mins
Sigmoid	23.08	0.19	0.78	68mins	20.44	0.43	0.42	68mins
Log	22.91	0.19	0.78	71mins	20.90	0.41	0.47	76mins

We find that the **sigmoid mapping performs best**, as it significantly accelerates

the training process while maintaining strong evaluation metrics.

By applying the **sigmoid method**, we achieved a **1.428X speed-up** on the *Banana* sequence from the **HyperNeRF dataset**. [compare video](#)

FUTURE WORKS

- Apply learned sampling frequency prediction using **deep learning models**(a model trained to predict optimal sampling frequency dynamically)
- Implement current method for **multi-view input**
- Find a more **robust method** for computing **motion scores** (current optical flow is affected by lighting)
- Although real-time rendering is achieved, the majority of processing time is still spent on training pipeline; thus, **adopting faster strategies** from recent papers is essential.

References

- [1] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 1–14.
- [2] Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., & Wang, X. (2024). 4D Gaussian splatting for real-time dynamic scene rendering (arXiv preprint arXiv:2310.06677).