4D Gaussian Splatting for Real-Time Dynamic Scene Rendering With Motion-Aware Frame Selection

Team 13
資工系大三 曾紹幃 林悅揚
指導教授: 陳冠文教授

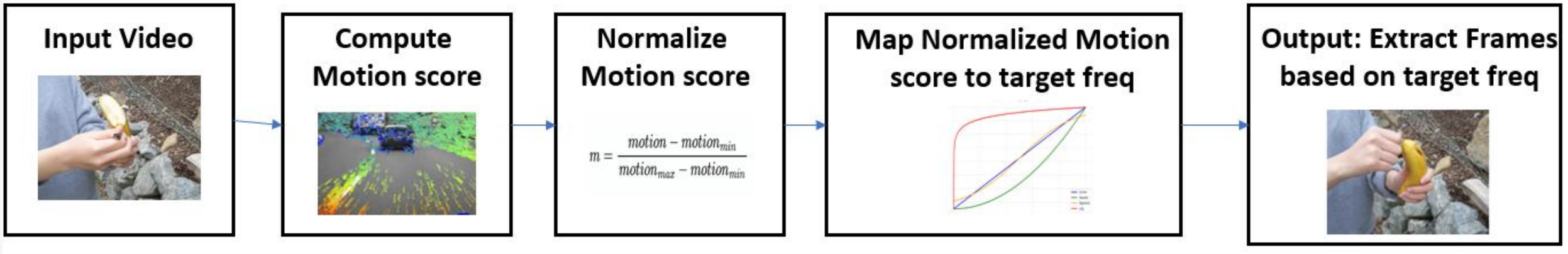
Introduction

- 3DGS[1] is a fast rendering method that represents static scenes using **transparent 3D Gaussians**.
- 4DGS[2] extends 3DGS to render **dynamic scenes** and achieves real-time rendering speed.
- However, it uses a **fixed sampling frequency**, which may over-sample slow-motion videos, leading to inefficient training.
- The pipeline, including COLMAP and training, is **time-consuming.**
- Adjusting the sampling frequency based on **motion scores** makes the process more **efficient** and accessible, with minimal impact on 4DGS results.

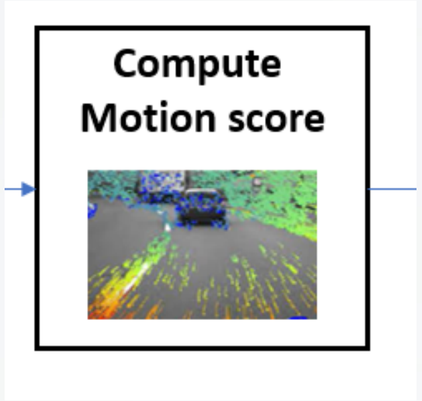
[1]Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), 1–14.
[2]Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., & Wang, X. (2024). 4D Gaussian splatting for real-time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

Method

- Pipeline :



Method

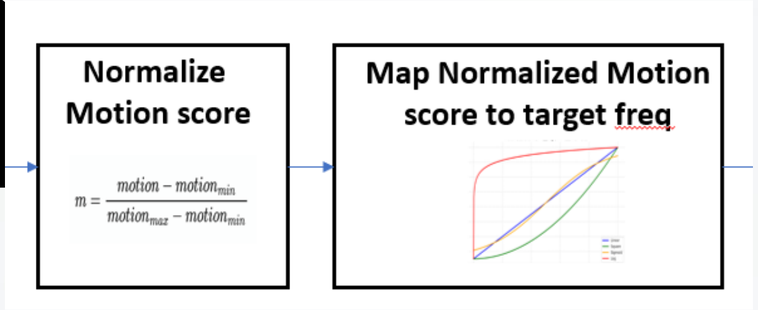


- **Compute Motion Scores via Optical Flow :**
  - For each frame pair t and t+1, dense optical flow is computed via Farneback's algorithm to obtain per-pixel displacement vectors (u(x,y),v(x,y))
  - The motion score for frame t is then defined as the average magnitude of all flow vectors:

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

  - 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.

Method



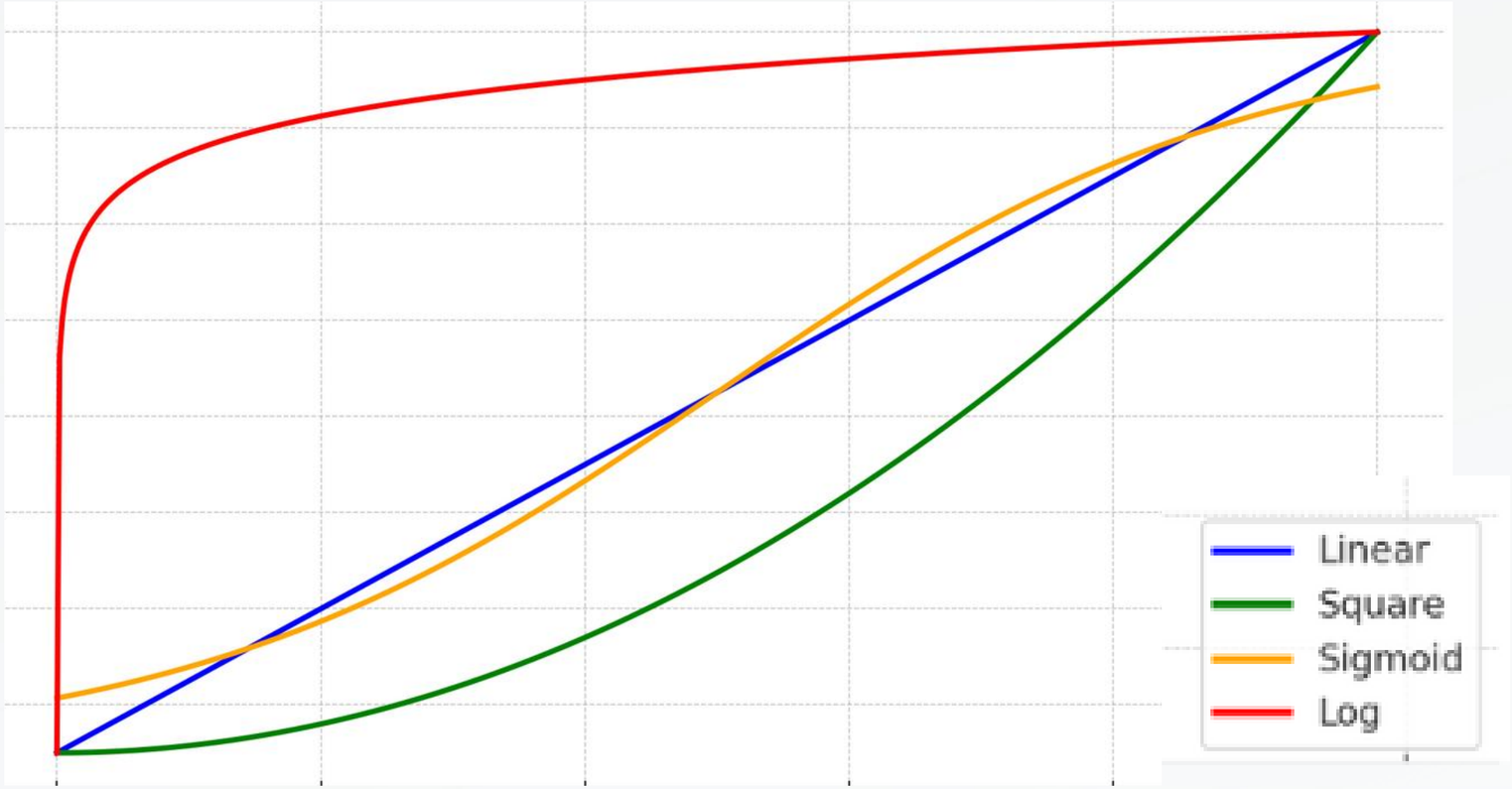
- **Normalize Motion Scores :**

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

- **Map Normalized Motion to $freq_{target}$:**
  - Linear : $freq_{target}$ = m·($freq_{max} - freq_{min}$)+ $freq_{min}$
  - Square : $freq_{target}$ = m2·($freq_{max} - freq_{min}$)+ $freq_{min}$
  - Sigmoid : $freq_{target}$ = (1/(1+e-5(m-0.5)))-1·($freq_{max} - freq_{min}$)+ $freq_{min}$
  - log : $freq_{target}$ = Take log(m + ε), normalize again, then map to $freq_{target}$

Method

- Map Normalized Motion score to $freq_{target}$(graph) :



x-axis:motion score

y-axis: $freq_{target}$

Method

- **Extract Frames Based on $freq_{target}$:**
  - Looks up the $freq_{target}$ assigned to this segment :

$$freq_{target} = \text{segment\_freq[segment\_idx]}$$

  - Track time using **next_save_time += 1 / $freq_{target}$**
  - Save the frame when **curr_time_sec >= next_save_time**

Experiment

Average motion score

unmove:1.79

walk:2.32

speed:3.08

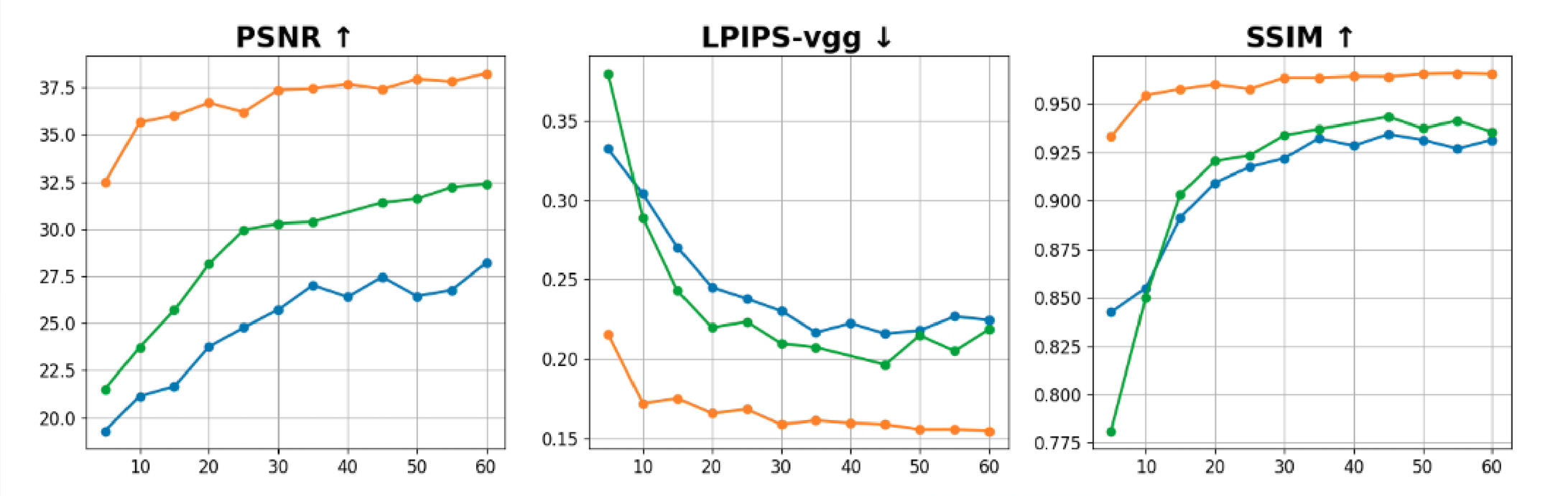
Experiment

Evaluation 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

Evaluation Metrics Across Motion Scores and Sampling Frequency



Experiment

Evaluation Metrics using our method on HyperNeRF Datasets

Broom: 197 frames

Banana : 513 frames

3D printer : 207 frames

Experiment

Evaluation Metrics using our method on Hypernerf Dataset

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

Method	Banana(17s)			
	PSNR ↑	LIPIPS ↓	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 ↑	LIPIPS ↓	SSIM ↑	Times ↓	PSNR ↑	LIPIPS ↓	SSIM ↑	Times ↓
Baseline	23.66	0.18	0.80	89mins	21.85	0.40	0.51	85mis
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

Experiment

Baseline

Sigmoid

Speed up: 1.428X

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.

DEMO

DEMO VIDEO

THANK YOU
FOR LISTENING