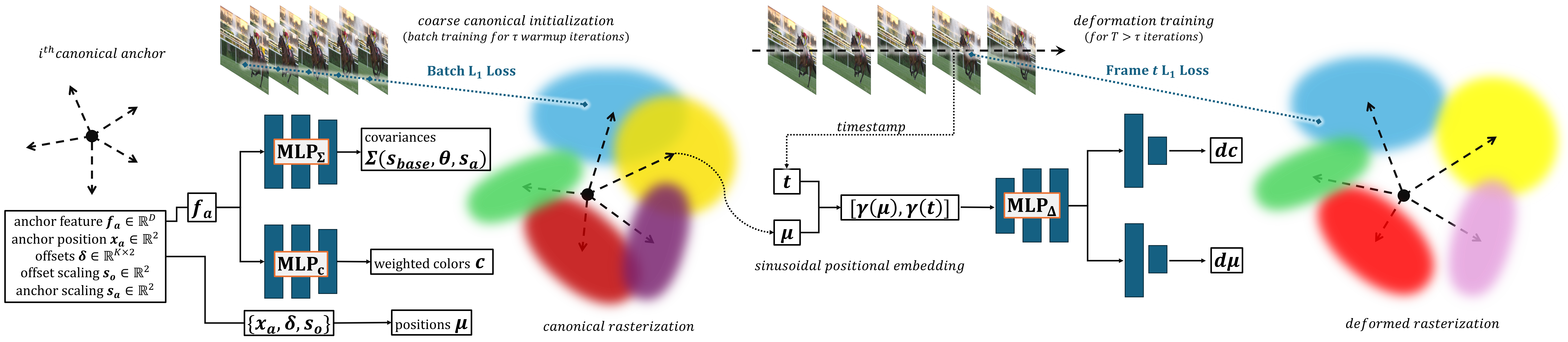
UNIVERSITY OF NOTRE DAME

SGV: Deforming Structured 2D Gaussians for Efficient and Compact Video Representation

Max Ficco, Matias Toro
Computer Science and Engineering Department, University of Notre Dame

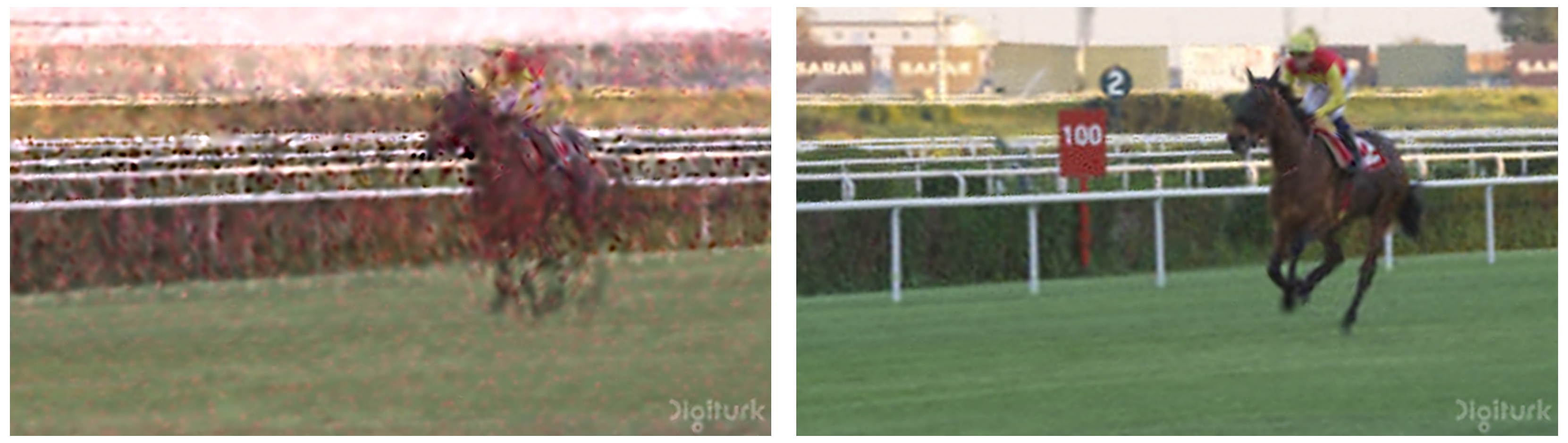
project page *GitHub repo*


i^{th} canonical anchor
anchor feature $f_a \in \mathbb{R}^D$
anchor position $x_a \in \mathbb{R}^2$
offsets $\delta \in \mathbb{R}^{K\times 2}$
offset scaling $s_o \in \mathbb{R}^2$
anchor scaling $s_a \in \mathbb{R}^2$
f_a
MLP_Σ
MLP_c
$\{x_a, \delta, s_o\}$
covariances $\Sigma(s_{base}, \theta, s_a)$
weighted colors c
positions μ
coarse canonical initialization
(batch training for τ warmup iterations)
Batch L1 Loss
canonical rasterization
deformation training
(for T > τ iterations)
Frame t L1 Loss
timestamp
t
μ
$[\gamma(\mu), \gamma(t)]$
sinusoidal positional embedding
MLP_Δ
dc
$d\mu$
deformed rasterization

Introduction

With the increasing use of video data across a wide range of domains including medical imaging, computer vision, and online streaming platforms, efficient and compact video representation is essential for cost-effective storage without sacrificing video fidelity. Recent methods in Deformable 2D Gaussian Splatting (D2GV) [1] represent video using a canonical set of 2D Gaussians that are deformed over time to render individual frames.

Compared to existing techniques in Implicit Neural Representations (INRs), Gaussian splatting achieves faster training and rendering times, often with improved video fidelity. However, storing and deforming Gaussian primitives independently ignores the spatial and temporal similarities among local Gaussians across frames. To exploit these similarities, we incorporate anchor-based neural Gaussians to utilize INR-based parameterization of Gaussian primitives for compact storage.



canonical frame $G_0 + MLP_\Delta(\gamma(\mu), \gamma(t)) \rightarrow$ *frame* G_t

frame t=6 from UVG "jockey"

Method

We partition video sequences into fixed-length segments for parallel training and linear scaling. Each segment uses N grid-positioned anchors with a set of attributes:

$$\boldsymbol{A} = \{\boldsymbol{x_a} \in \mathbb{R}^2, \boldsymbol{f_a} \in \mathbb{R}^D, \boldsymbol{\delta} \in \mathbb{R}^{K\times 2}, \boldsymbol{s_o} \in \mathbb{R}^2, \boldsymbol{s_a} \in \mathbb{R}^2\}$$

The positions of K associated Gaussians are computed as:

$$\{\boldsymbol{\mu}^{(k)}\}_{k=0}^{K-1} = \boldsymbol{x_a} + \{\boldsymbol{\delta}^{(k)}\}_{k=0}^{K-1} \odot \boldsymbol{s_o}$$

$\mathbf{MLP_c}$ predicts weighted colors for K associated Gaussians. $\mathbf{MLP_\Sigma}$ predicts scaling and rotation parameters $\boldsymbol{s_{base}}, \boldsymbol{\theta}$ to ensure covariance $\boldsymbol{\Sigma}$ is positive semi-definite.

$$\boldsymbol{\Sigma} = \boldsymbol{RS}(\boldsymbol{RS})^\top ;$$

$$\boldsymbol{R} = \begin{bmatrix} \cos(\boldsymbol{\theta}) & -\sin(\boldsymbol{\theta}) \\ \sin(\boldsymbol{\theta}) & \cos(\boldsymbol{\theta}) \end{bmatrix}, \qquad \boldsymbol{S} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}, (s_1, s_2) = \boldsymbol{s_{base}} \odot \boldsymbol{s_a}$$

$\gamma(\cdot)$ is the positional encoding function where p represents position $\boldsymbol{\mu}$ or time $\boldsymbol{t}$, normalized to $(0,1]$, and L is the number of encoding frequencies.

$$\gamma(p) = \left(\sin\left(2^k\pi p\right), \cos\left(2^k\pi p\right)\right)_{k=0}^{L-1}$$

Gaussian primitives from the canonical frame are deformed to render individual frames across time. $\mathbf{MLP_\Delta}$ predicts position and color deformations for frame $\boldsymbol{t}$ Gaussians:

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + d\boldsymbol{\mu}, \qquad \boldsymbol{c}' = \boldsymbol{c} + d\boldsymbol{c}$$

Following [2], the final pixel color $\boldsymbol{C}$ is then computed using:

$$\boldsymbol{C} = \sum_{i \in I} \boldsymbol{c}' G_i$$

Where the spatial density of a Gaussian is defined as:

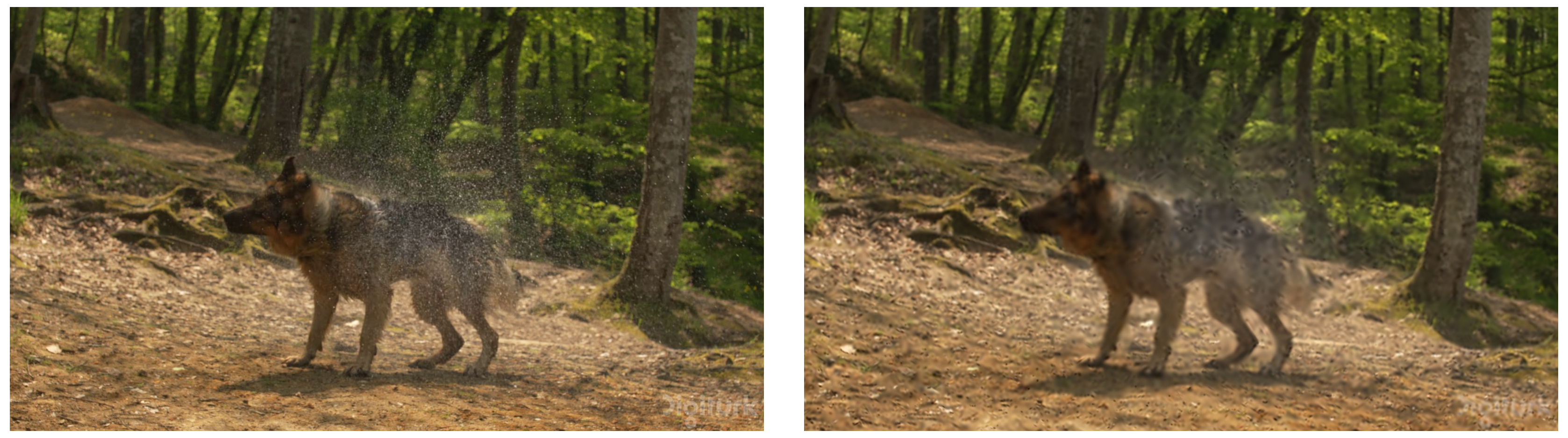
$$G(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}')^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}'))$$

Results

Quantitative results comparison on UVG† [3] dataset (resized to 720p with 4x framerate subsampling). Each uses 20,000 Gaussian primitives. Metrics are averaged across the first 10 frames of each video.

Model	Size (MB)	Iterations	Training Time (s)	Decode FPS	PSNR
SGI*	6.41	15000	639.85	677.85	32.69
D2GV	1.10	70000	485.05	333.06	32.97
Ours	0.95	40000	258.17	287.92	29.90

The above are limited preliminary results. Check the project page for updated results and video demos!



Ground Truth *Ours*

Current limitations in fine detail shown by frame t=2 from UVG "shake"

References

*SGI is an unreleased paper, rendering single images using anchored neural Gaussians

[1] Liu, M., et al. "D2GV: Deformable 2D Gaussian Splatting for Video Representation in 400FPS," *arXiv preprint arXiv:2503.05600*, 2025.

[2] L. Zhu, G. Lin, J. Chen, X. Zhang, Z. Jin, Z. Wang, and L. Yu. Large Images are Gaussians: High-quality large image representation with levels of 2D Gaussian splatting. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 10977–10985, 2025.

[3] A. Mercat, M. Viitanen, J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," in *Proceedings of the 11th ACM multimedia systems conference*, 2020, pp. 297–302.

†videos used: *Bosphorus*, *Beauty*, *SetGo*, *Bee*, *Yacht*, *Jockey*, *Shake*

Acknowledgements

As part of the CSE-CSA, funding for this work was provided by NSF CNS-2211428. We also thank our research mentor Zixuan Pan, faculty advisor Yiyu Shi and program director Arturo Russell.