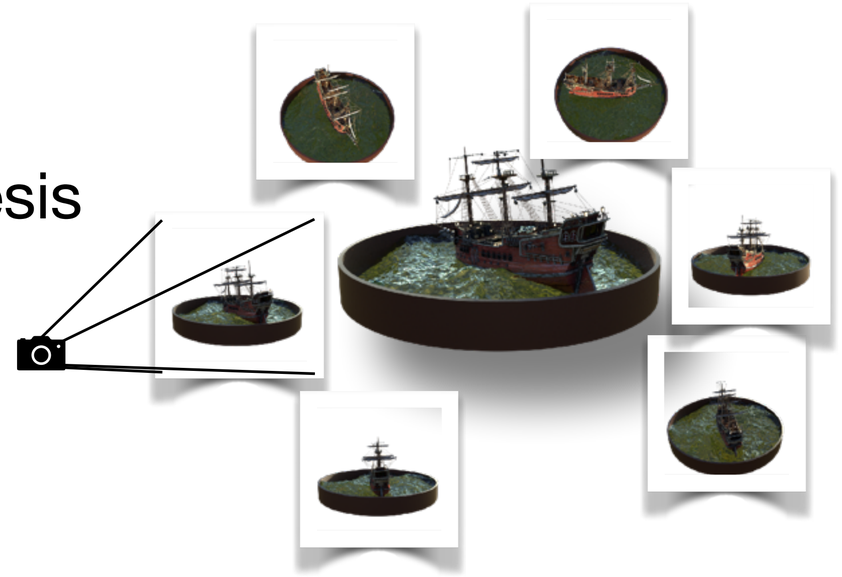
VII: Gaussian Splatting

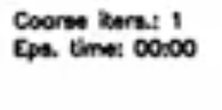
3D CV

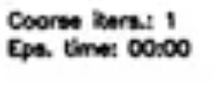
Kirill Struminsky

In the Previous Episode

- Neural Radiance Fields (NeRF) for novel view synthesis
  - Volumetric representations for 3D data
  - Volumetric rendering
- Improvements & applications of NeRFs







Neural Radiance Fields

Representing a Scene with Neural Fields

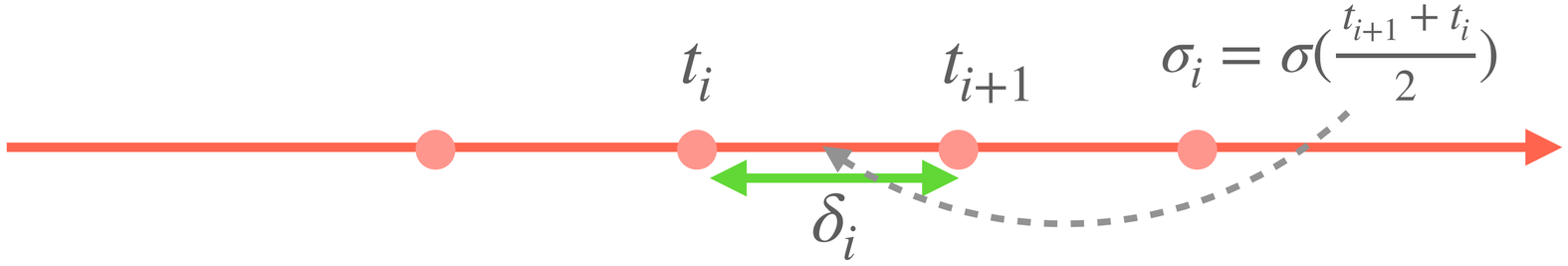
- Represent a scene with two fields
  - Density: $\sigma(x) : \mathbb{R}^3 \rightarrow \mathbb{R}^+$
  - Radiance: $C(x, d) : \mathbb{R}^3 \times S^2 \rightarrow \mathbb{R}^3$
- Density represents is related to opacity
- Radiance represents color of a point
- Architecture: MLP + positional embeddings

Mildenhall B. et al. Nerf: Representing scenes as neural radiance fields for view synthesis //Communications of the ACM. – 2021. – Т. 65. – №. 1. – С. 99-106.

Computing Pixel Color

- Density $\sigma(t) \in [0, +\infty)$ is related to opacity at t
- Divide the ray with points $t_1, \ldots, t_n$ and define

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



- Expected color along a ray is given by

$$C = \sum_i c(t_i) \cdot \alpha_i \prod_{j<i} (1 - \alpha_j)$$

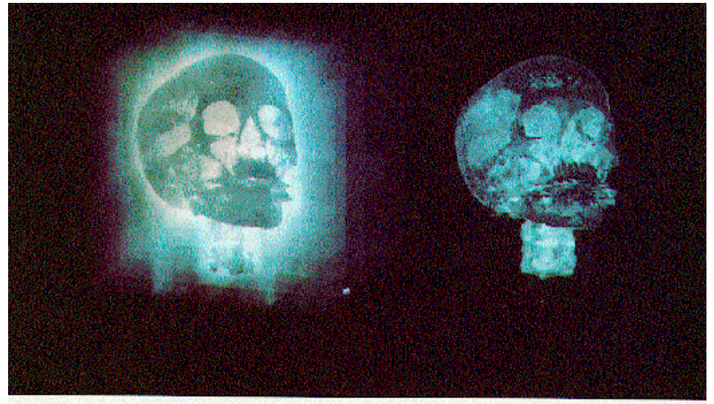


Fig. 11. Costs of rendering Figure 8 using hierarchical enumeration and adaptive termination.

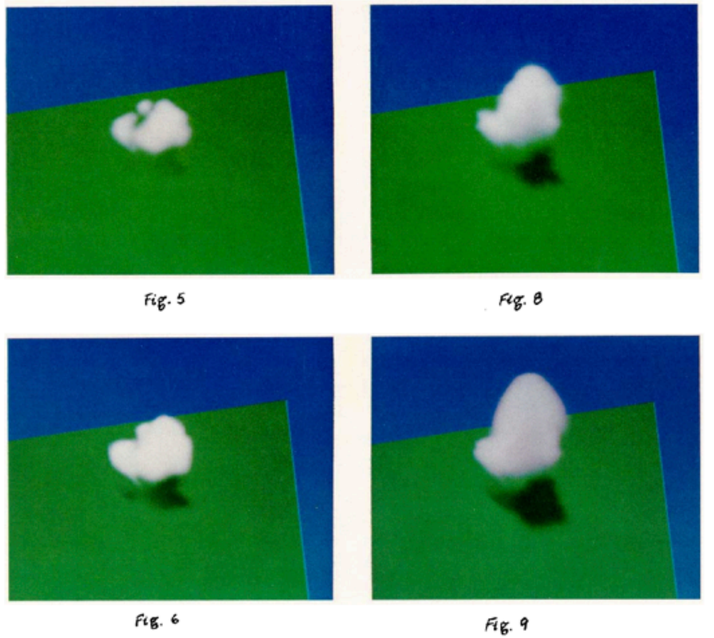


Fig. 5 Fig. 8

Fig. 6 Fig. 9

Computational Overhead

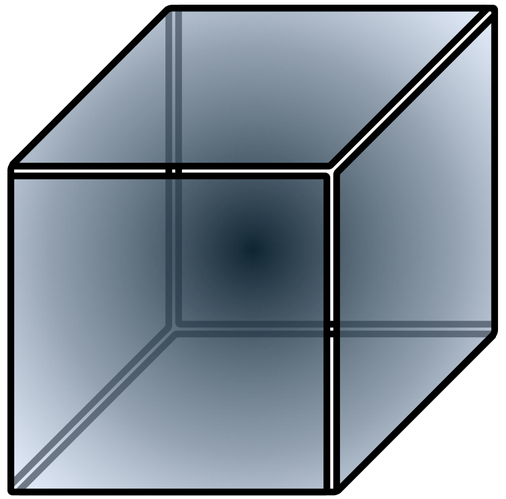
- Notorious rendering time even with high-end GPUs
  - MipNeRF: 0.07 FPS
  - Instan-NGP: 9.0 FPS
- Today: gaussian splatting
  - Alternative solution allowing real-time rendering
  - Gaussian Splatting: 100+ FPS

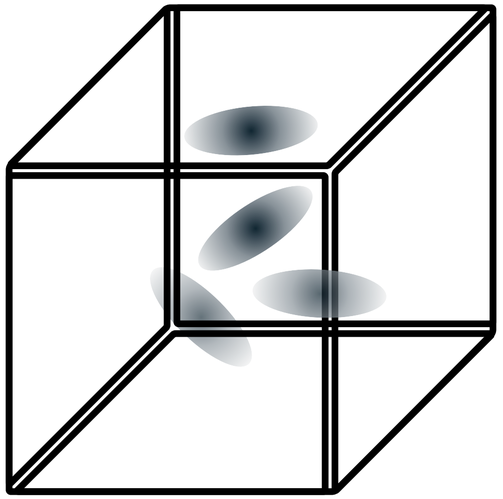
Kerbl B. et al. 3d gaussian splatting for real-time radiance field rendering //ACM Transactions on Graphics. – 2023. – Т. 42. – №. 4. – С. 1-14.

Gaussian Splatting

Idea: Volumetric Rendering + Rasterisation

- Neural Radiance Fields employ **a dense** field parameterisation
- Rendering algorithm runs independently for each pixel
- Gaussian splatting employs **a sparse** parameterisation
  - Similar to meshes, scene consists of primitive shapes
  - Unlike meshes, these primitives have volume
  - Volumetric rendering algorithm based on **rasterization**

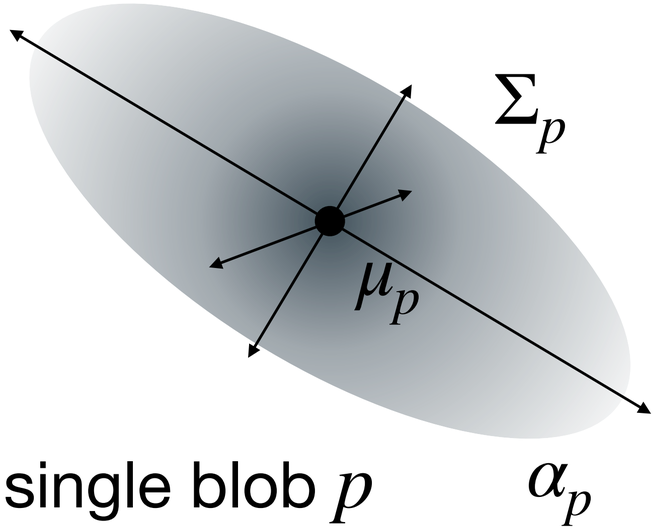




Final Rendering

3D Gaussian Visualization

Representing Scene with Gaussians



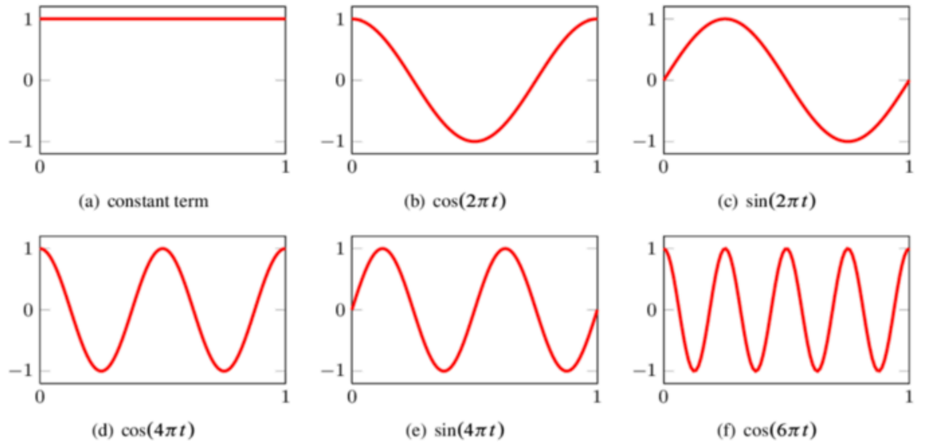
- We will fill the volume with multiple tiny blobs
- In NeRFs we relied on a density field $\sigma(x)$
- Here we will define opacity $\alpha(x) = 1 - \exp(\sigma(x)\Delta x)$ of a single blob p

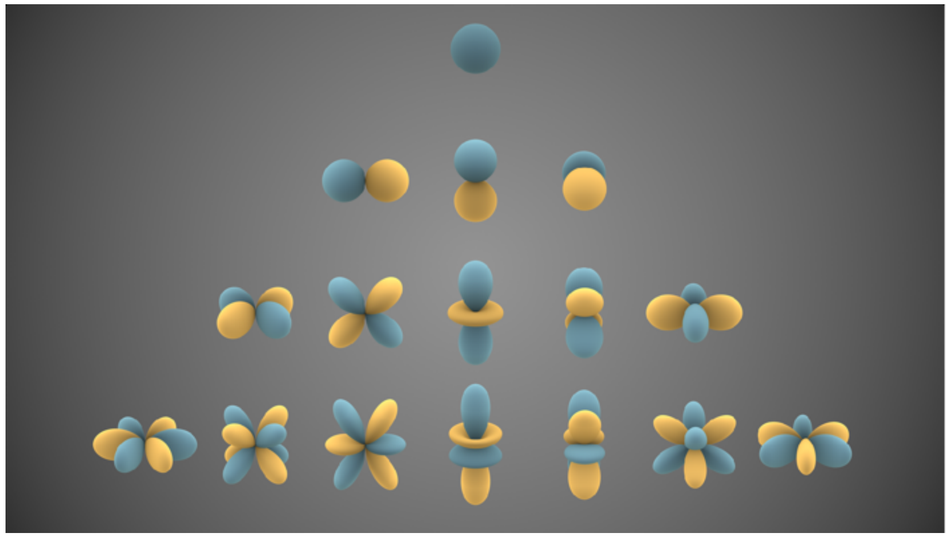
$$\alpha(x) = \alpha_p \exp\left(-\frac{1}{2}(x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p)\right)$$

- Parameters $\alpha_p, \mu_p, \Sigma_p$ define opacity, position, and shape of the blob respectively

Parameterizing Radiance with SH

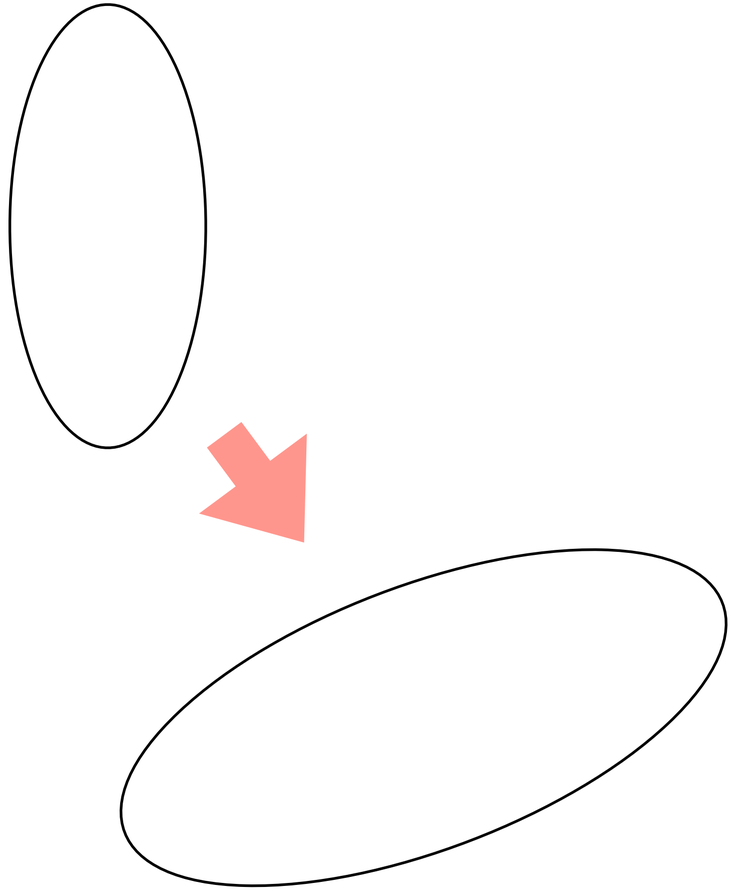
- Each blob radiates light
- Radiance function depends on ray direction
- Omit dependence on position within the blob
- Similar to $\{\sin(k\pi x), \cos(k\pi x)\}_{k\in\mathbb{N}}$ on $[0,1]$, **spherical harmonics** define an orthogonal functional basis for S^2 functions
  - *Physicists: harmonics are solutions to wave equation*
  - *Mathematicians: harmonics are eigenfunctions of Laplace operator*





Triangles vs Gaussians

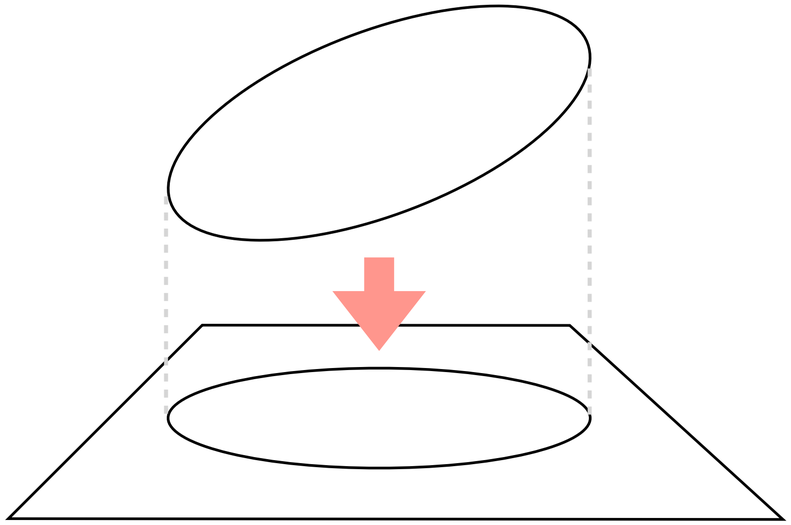
Affine Transformations



- Triangles have multiple convenient properties
- A projective transformation of a triangle is still a triangle
- An affine transformation of a Gaussian is still a Gaussian
  - Let $\xi \sim \mathcal{N}(\mu, \Sigma)$ and $\phi = A\xi + b$
  - Then $\phi \sim \mathcal{N}(A\mu + b, A\Sigma A^T)$
- We will use this property for *world-to-camera* transformation

From 3D to 2D

Marginalization



- Marginal distribution of a Gaussian is a Gaussian
- For a r.v. $\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} \sim \mathcal{N}(\mu, \Sigma)$ distribution of $\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}$ is $\mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right)$
- From graphics standpoint, vector $\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}$ is an orthographic projection of $\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix}$

Perspective Projection with Gaussians

Projection

- Perspective projections are non-linear in Euclidian space

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{pmatrix}$$

- We can use the Jacobian at $x = \mu_p$ of projection as a substitute

$$J = \begin{pmatrix} \frac{f}{z} & 0 & -\frac{fx}{z^2} \\ 0 & \frac{f}{z} & -\frac{fx}{z^2} \end{pmatrix}$$

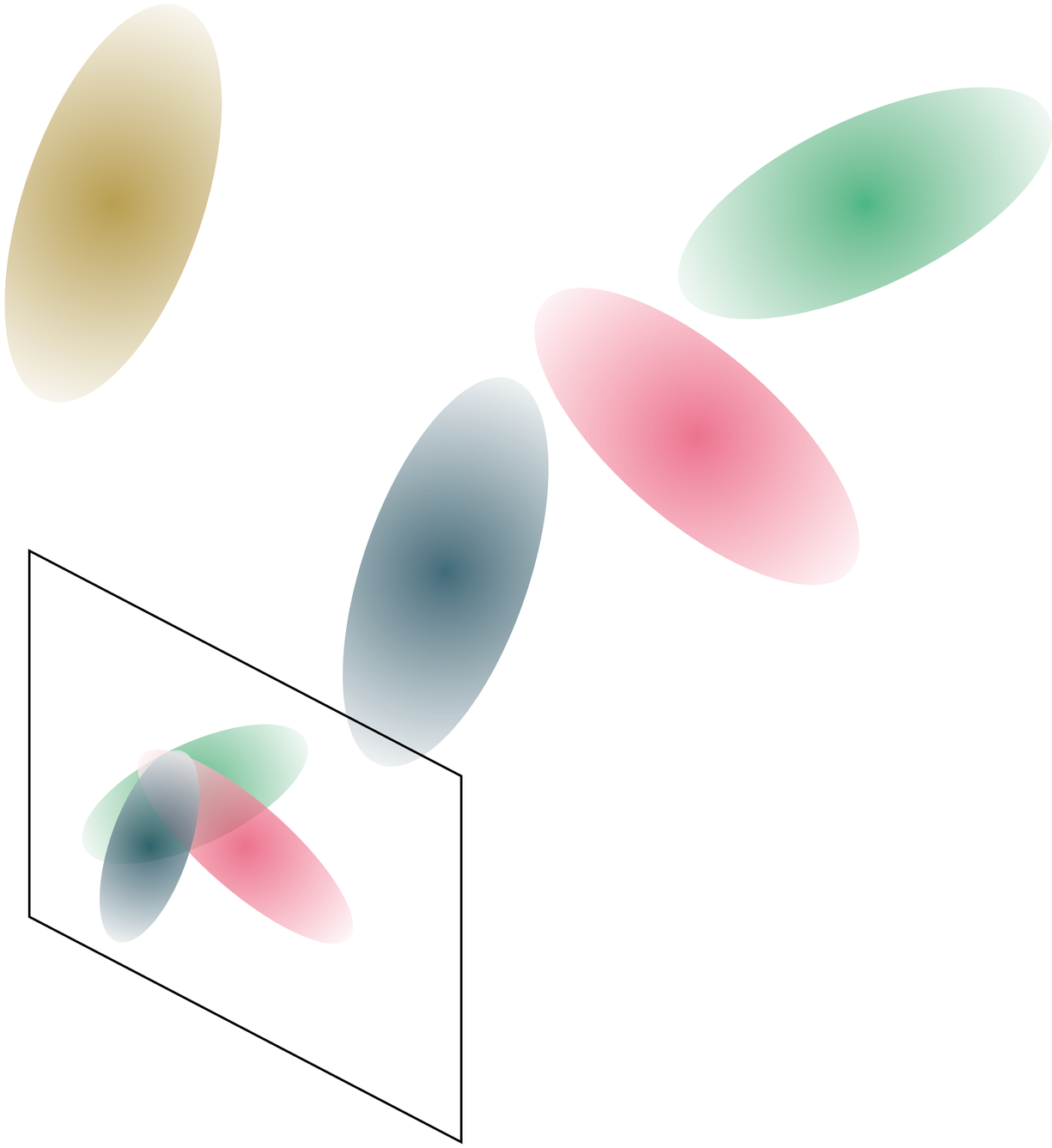
- In practice, we end up with 2D Gaussian with $\Sigma' = JW\Sigma W^T J^T$, where W is world-to-cam matrix

Rendering Algorithm

Rendering multiple spats

- Culling
  - Exclude splats outside of the frame
- Render each splat from closest to farthest
  - Blend frame with alpha-compositing

$$C = \sum_i C_i \alpha_i \prod_{j<i} (1 - \alpha_j)$$



Pseudo-Code

Why Rasterization is so Efficient?

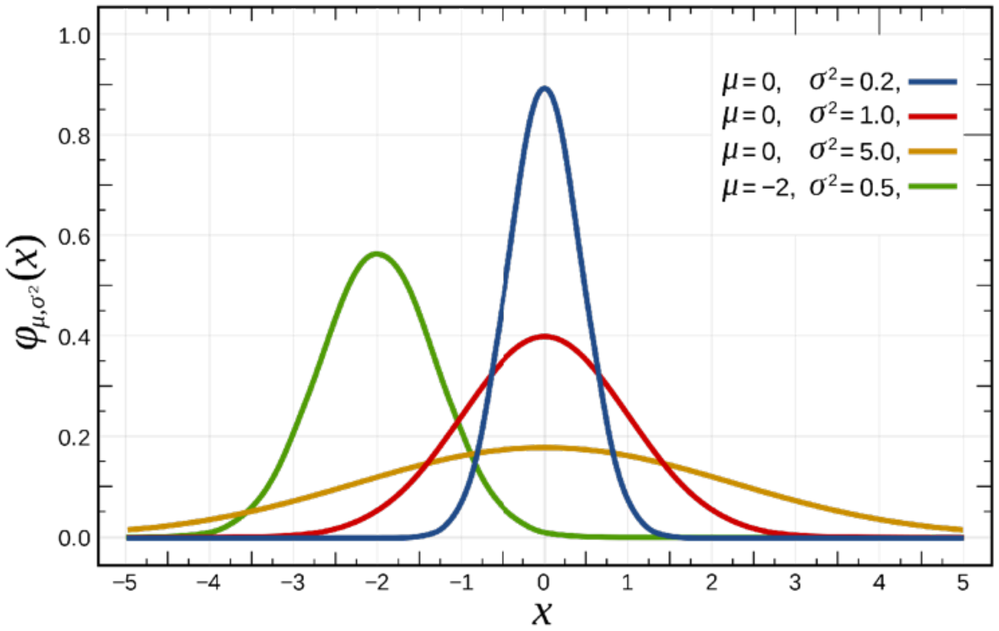
```
for each pixel  # NeRF, 10e6 pixels
  for each segment on a ray  # 10e2 ray segments
    compute & accumulate radiance


for each splat  # Gaussian splatting, 10e4 splats
  for each pixel on a splat  # 10e2 pixels in a splat
    compute & accumulate radiance
```

Fitting Gaussian Splats

A Bag of Tricks

- Unlike triangles in meshes, Gaussian splats have *soft* edges
- As a result, splats amenable to optimization



- However, gradient vanishes far from the mean
- Photometric loss has multiple local minima

$$\|I_{gt} - I_{rendered}\|^2$$

Fitting Gaussian Splats

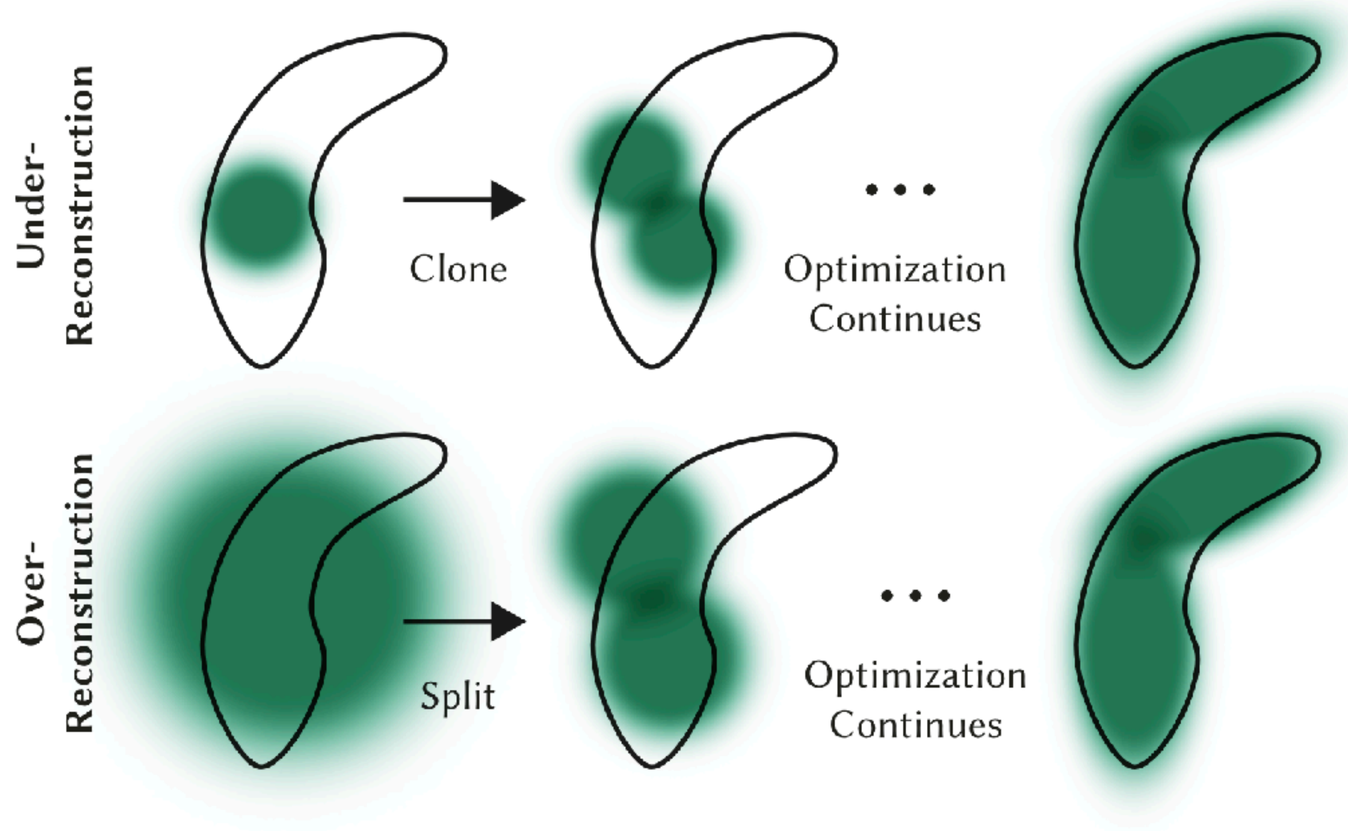
Initialisation with SfM

PSNR 20.90

PSNR 25.82

Fitting Gaussian Splats

Adaptive Control of Gaussians



Overview

- Pros: efficiency
  - Blazingly fast inference
  - Training is also relatively fast
- Cons: overparameterized representation
  - Local optima, PSNR does not match NeRFs
  - Naive representation takes up to 1GB of memory

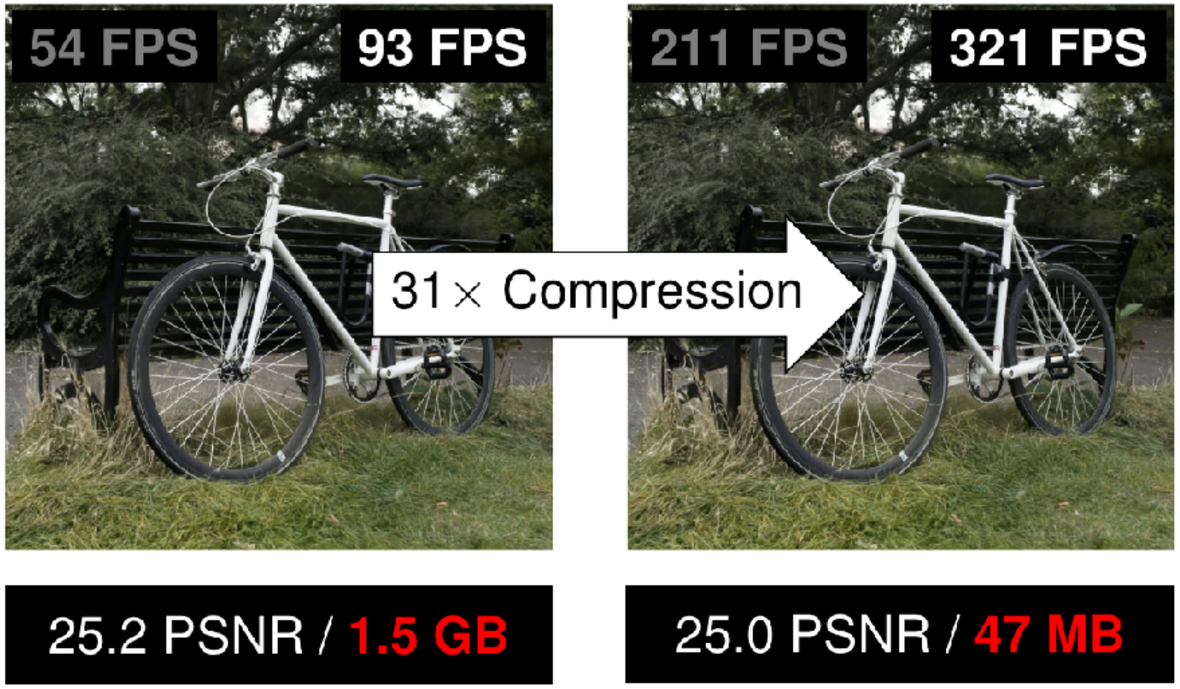
Compression

Memory Footprint

- High-quality scene contains 30k splats
  - Shape is parameterised with mean (3 floats) and covariance (6 floats)
  - Radiance is parameterized with α (1 float) and $16 \times 3 = 48$ SH coefficients
- Approximately $7 \cdot 10^6$ bytes
  - **Color** constitutes **82%** of weights
  - Gaussian **covariance** constitutes **10%** of weights
- Differs from NeRFs by two orders of magnitude

Room for Optimization

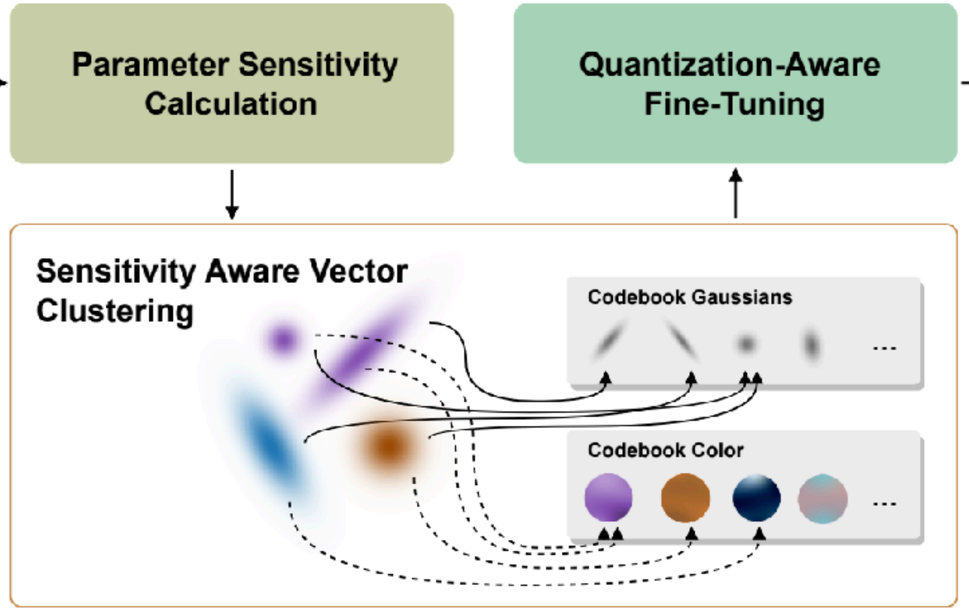
- Memory footprint can be reduced with standard solutions
  - Quantization
  - Low-precision parameterisation
  - Compression



Niedermayr S., Stumpfegger J., Westermann R. Compressed 3d gaussian splatting for accelerated novel view synthesis //arXiv preprint arXiv:2401.02436. – 2023.

Compression Pipeline

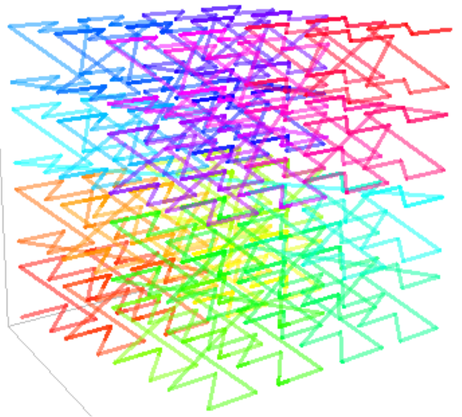
Reconstructed Scene



Compressed Scene

		PSNR ↑	SSIM ↑	LPIPS ↓	SIZE ↓
Color	1024	26.95(-0.67)	0.80(-0.02)	0.24(+0.03)	28.47
	2048	26.95(-0.62)	0.80(-0.02)	0.24(+0.03)	28.65
	4096	26.98(-0.63)	0.80(-0.02)	0.24(+0.03)	28.80
	8192	27.00(-0.58)	0.80(-0.02)	0.24(-0.03)	28.92
Gaussian	1024	26.95(-0.79)	0.80(-0.02)	0.24(+0.03)	28.14
	2048	26.97(-0.80)	0.80(-0.02)	0.24(+0.03)	28.45
	4096	26.98(-0.63)	0.80(-0.02)	0.24(+0.03)	28.80
	8192	26.97(-0.60)	0.80(-0.02)	0.24(+0.03)	29.06

	PSNR ↑	SSIM ↑	LPIPS ↓	SIZE ↓
baseline	27.179	0.861	0.115	1379.99
+ Pruning	27.083	0.856	0.118	1217.25
+ Color Clustering	25.941	0.818	0.178	278.41
+ Gaussian Clustering	25.781	0.811	0.186	164.15
+ QA Finetune	26.746	0.844	0.144	86.69
+ Encode	26.746	0.844	0.144	58.40
+ Morton Order	26.746	0.844	0.144	46.57



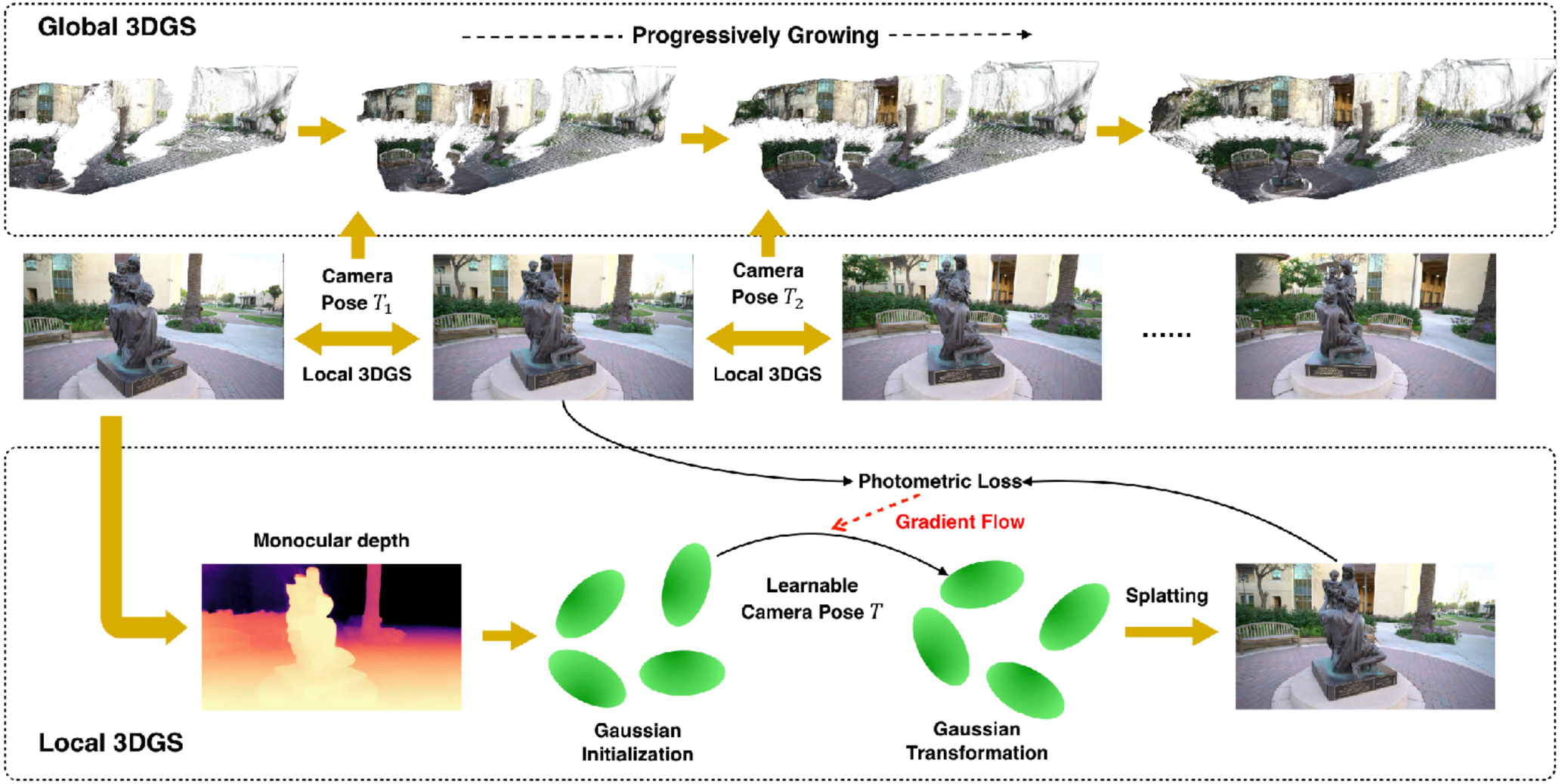
Niedermayr S., Stumpfegger J., Westermann R. Compressed 3d gaussian splatting for accelerated novel view synthesis //arXiv preprint arXiv:2401.02436. – 2023.

Gaussian Splats without Colmap

Do We Need Camera Positions?

- Novel view synthesis still relies on SfM (COLMAP)
- NeRF allow optimizing camera positions
  - BARF, Nope-NeRF, etc.
- Learning positions from scratch is still a challenge

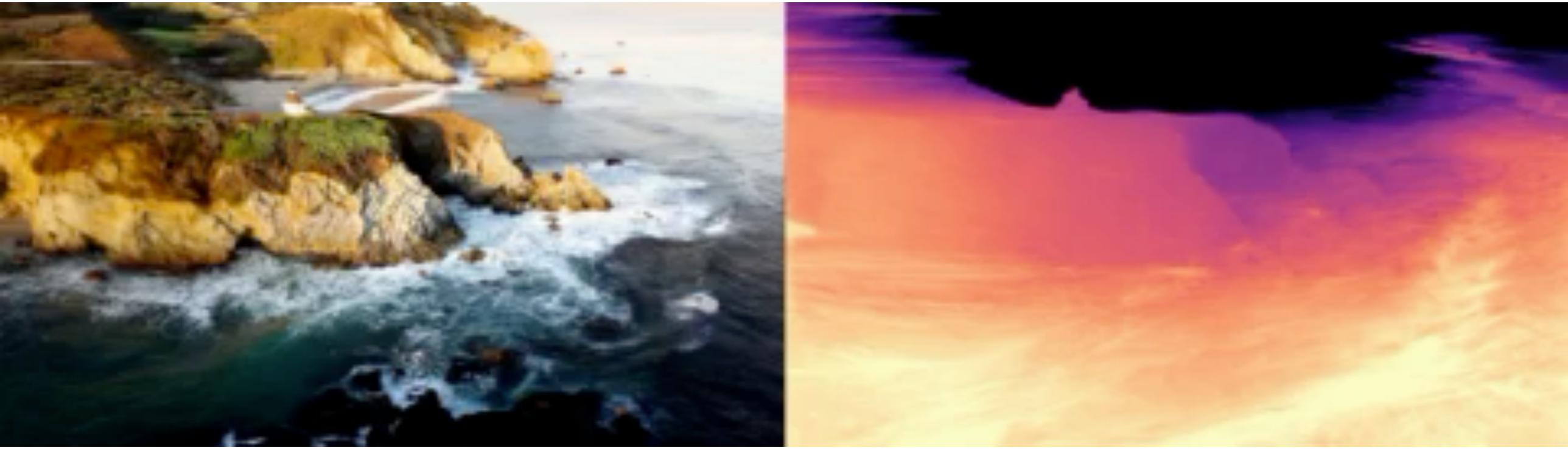
COLMAP-Free 3D Gaussian Splatting



Fu Y. et al. Colmap-free 3d gaussian splatting //arXiv preprint arXiv:2312.07504. – 2023.

Demo on a Real Scene

Demo on a Generated Scene



Key Takeaways

- Gaussian Splatting
  - Volumetric model
  - Efficient rendering based on rasterisation
- Capable or real-time rendering
- Almost reaches visual fidelity of NeRFs

- Next time: generating 3D content with generative models