Hands-On AI Based 3D Vision
Summer 25

Lecture 08 – 3D Gaussian Splatting (3DGS)

Prof. Dr. Gerard Pons-Moll

University of Tübingen / MPI-Informatics

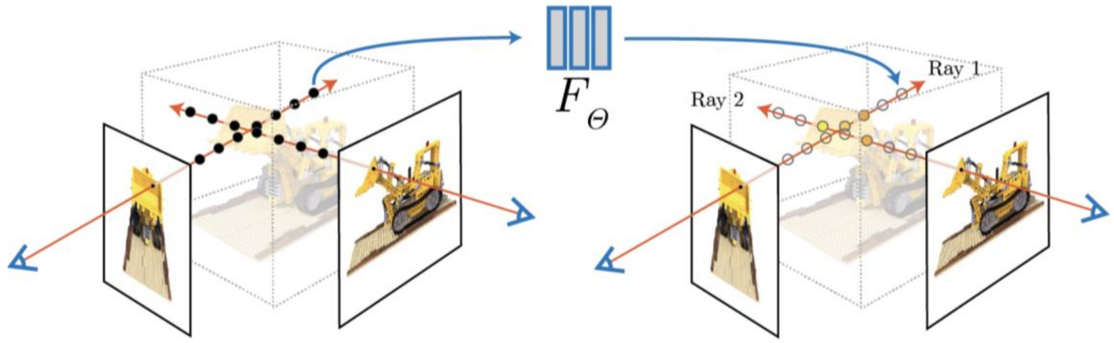
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

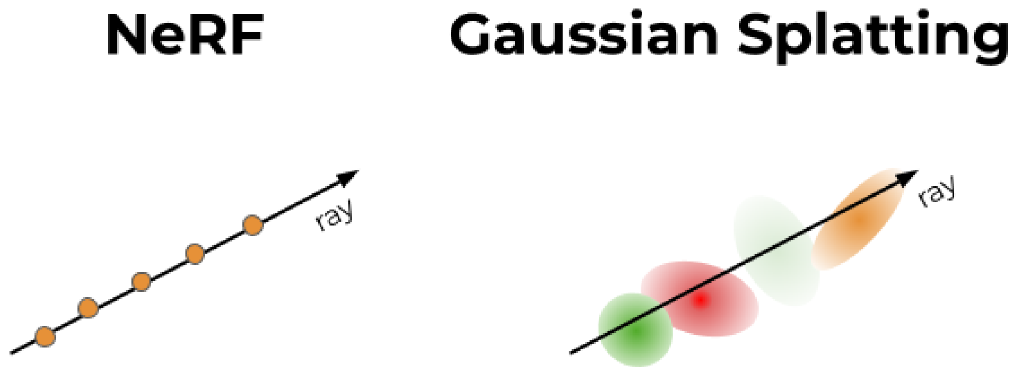
In this lecture, we will learn …

- **Problems of NERF**
- **Point-Based Rendering**
- **3D Gaussian Splatting (3DGS)** [Kerbl & Kopanas ‘23]
- Applications of 3DGS and addressing its limitations (i.e., dynamic scene, compression, surface reconstruction…)

Problems of Nerf

- NeRF suffers from slow training and rendering.
- We have to query a neural network to obtain color and density values for each point.



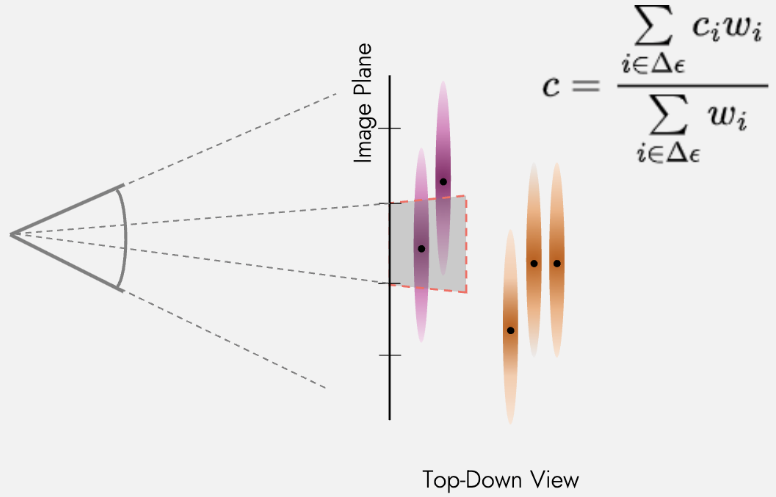


Point-Based Rendering

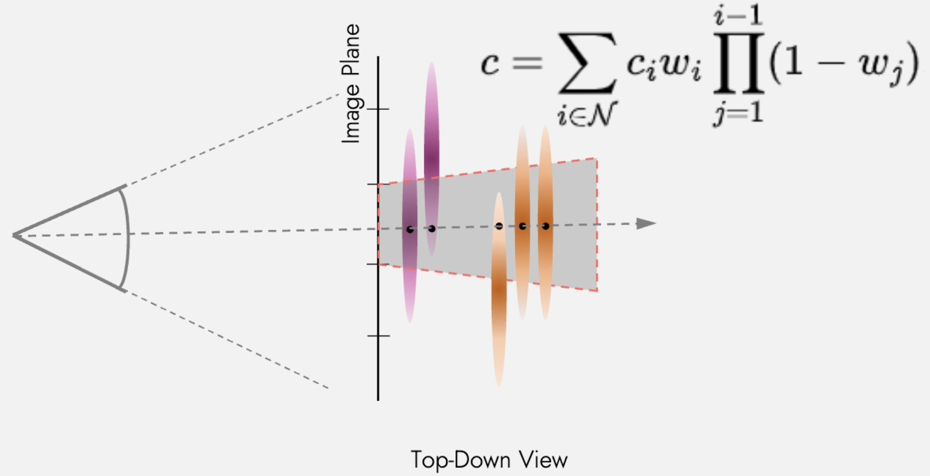
Surface Splatting vs Volume Splatting

1. How do we blend points in screen space?

[Zwicker1 '01] / [Yifan '19]



[Zwicker2 '01] / [Kerbl & Kopanas '23]



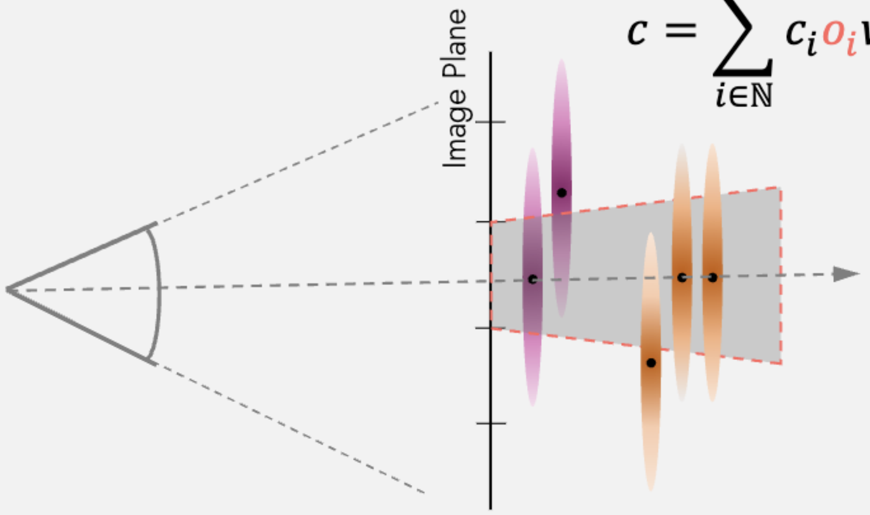
[Zwicker1 '01] Surface Splatting
[Yifan '19] Differentiable Surface Splatting for Point-Based Geometry Proccessing
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Surface Splatting vs Volume Splatting

1. How do we blend points in screen space?
2. Opacity for each point, allows us to make points disappear.

[Zwicker2 '01] / [Kerbl & Kopanas '23]

$$c = \sum_{i \in \mathbb{N}} c_i o_i w_i \prod_{j=1}^{i-1} (1 - o_i w_j)$$



Top-Down View

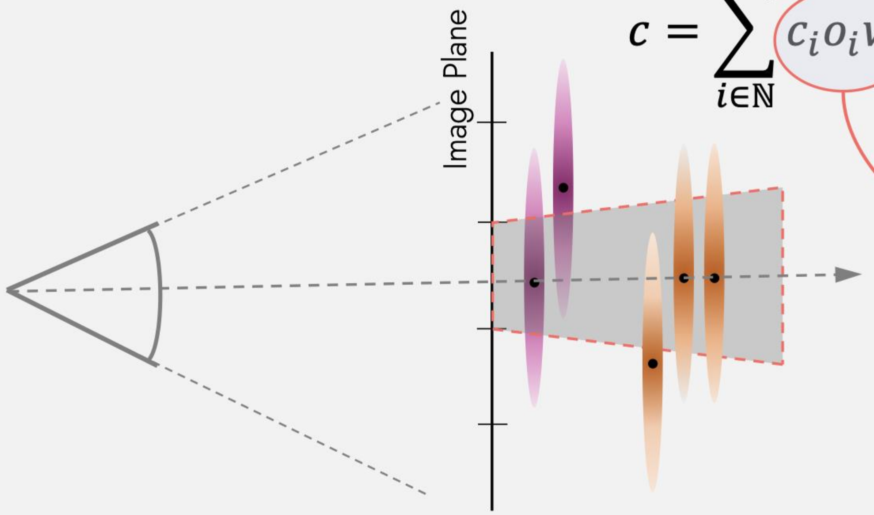
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Surface Splatting vs Volume Splatting

1. How do we blend points in screen space?
2. Opacity for each point, allows us to make points disappear.

Ours

$$c = \sum_{i \in \mathbb{N}} c_i o_i w_i \prod_{j=1}^{i-1} (1 - o_i w_j)$$



Top-Down View

results in the MLP being evaluated at continuous positions over the course of optimization. We use these samples to estimate $C(\mathbf{r})$ with the quadrature rule discussed in the volume rendering review by Max [26]:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad (3)$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. This function for calculating $\hat{C}(\mathbf{r})$ from the set of $(\mathbf{c}_i, \sigma_i)$ values is trivially differentiable and reduces to traditional alpha compositing with alpha values $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$.

Screenshot from NeRF [Mildenhall '20]

Goal: Reconstructing 3D world from images and videos

Input images

https://3dgstutorial.github.io/

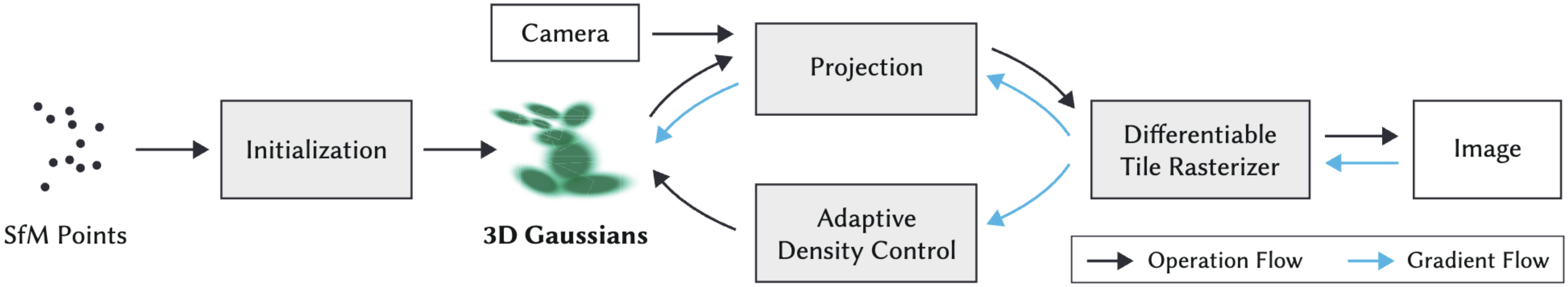

Menu
Views
Capture
▶ 3D Gaussians
▶ Camera Point view
Die Aufnahme wurde begonnen
▼ Metrics
78.72 (12.70 ms)

3D Gaussian Splatting (3DGS) [Kerbl & Kopanas ‘23]

- Splat-based representation
- Use 3D Gaussians instead of points or a mesh.
- It does not include any neural network.

3D Gaussian Splatting (3DGS) [Kerbl & Kopanas ‘23]

- Splat-based representation
- Use 3D Gaussians instead of points or a mesh.
- It does not include any neural network.



Method Overview

Parametrization of 3D Gaussian

How to parametrize 3D Gaussian?

Parametrization of 3D Gaussian

How to parametrize 3D Gaussian?

3D Gaussian parametrized by:

- Covariance **Σ**
- Mean) **μ**
- Opacity **o**
- Color ***c*** – RGB values or spherical harmonics (SH) coefficients.

How to optimize a covariance matrix $\mathbf{\Sigma}$?

- Not all symmetric matrices are covariance matrices and gradient updates can easily make them invalid.
- The covariance matrix $\mathbf{\Sigma}$ of a 3D Gaussian is analogous to describing the configuration of an ellipsoid.
- $\mathbf{\Sigma}$ has a physical meaning if its a positive-semi definite matrix. So factorize as follows:

How to optimize a covariance matrix **Σ**?

- Not all symmetric matrices are covariance matrices and gradient updates can easily make them invalid.
- The covariance matrix Σ of a 3D Gaussian is analogous to describing the configuration of an ellipsoid.
- **Σ** has a physical meaning if its a <u>positive-semi definite matrix</u>. So factorize as follows:

3x3 Covariance Matrix

$$\Sigma = RSS^T R^T$$

3x3 Rotation matrix

Diagonal scaling matrix (3 parameters for scale)

Projection of a covariance matrix **Σ** into 2D

3x3 Covariance Matrix

$$\Sigma = RSS^T R^T$$

3x3 Rotation matrix

Diagonal scaling matrix (3 parameters for scale)

Projection of a covariance matrix $\boldsymbol{\Sigma}$ into 2D

3x3 Covariance Matrix

$$\Sigma = RSS^T R^T$$

3x3 Rotation matrix

Diagonal scaling matrix (3 parameters for scale)

2x2 variance matrix

$$\Sigma' = JW\Sigma W^T J^T$$

$$G_{2D} = \mathcal{N}(\mathbf{x}_{2D}; \mu_{2D}, \Sigma')$$

Projected 2D Gaussian

Jacobian of the affine approximation of the projective transformation

Viewing Transformation

Image Formation Model of NeRF

$$C = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i$$

$$\alpha_i = (1 - \exp(-\sigma_i \delta_i)) \text{ and } T_i = \prod_{j=1}^{i-1} (1 - \alpha_i)$$

Color of a pixel

Transmittance

Color of each point

Image Formation Model of 3D Gaussian Splatting

$$\boxed{C} = \sum_{i \in \mathcal{N}} \boxed{c_i}\boxed{\alpha_i} \boxed{\prod_{j=1}^{i-1} (1 - \alpha_j)},$$

Color of a pixel

Color of each point

$\alpha = \mathbf{o}G_{2D}(\mathbf{x})$

Transmittance

NeRF vs Gaussian Splatting

$$C = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i$$

Nerf

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

Gaussian Splatting

$$\alpha = \mathbf{o} G_{2D}(\mathbf{x})$$

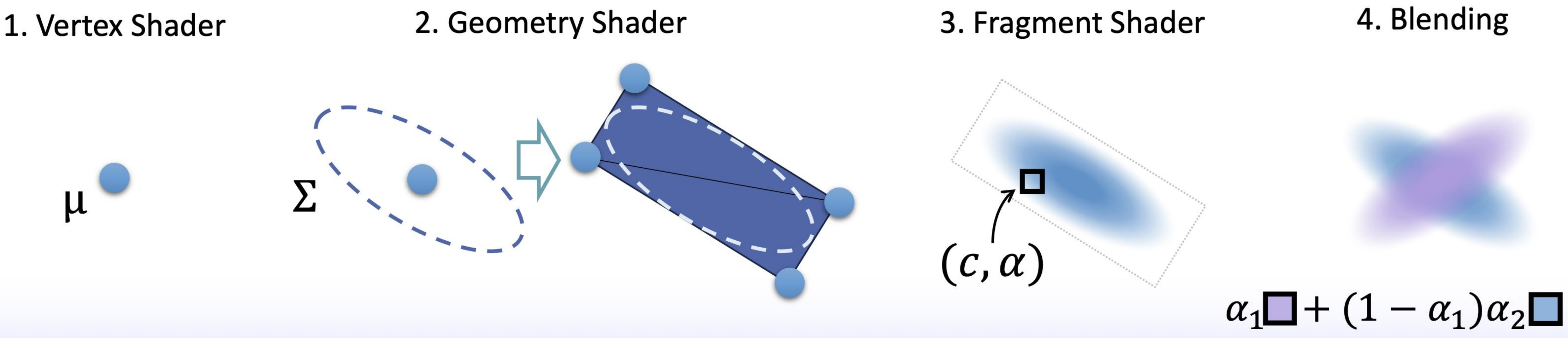
Color of a pixel

Transmittance

Color of each point

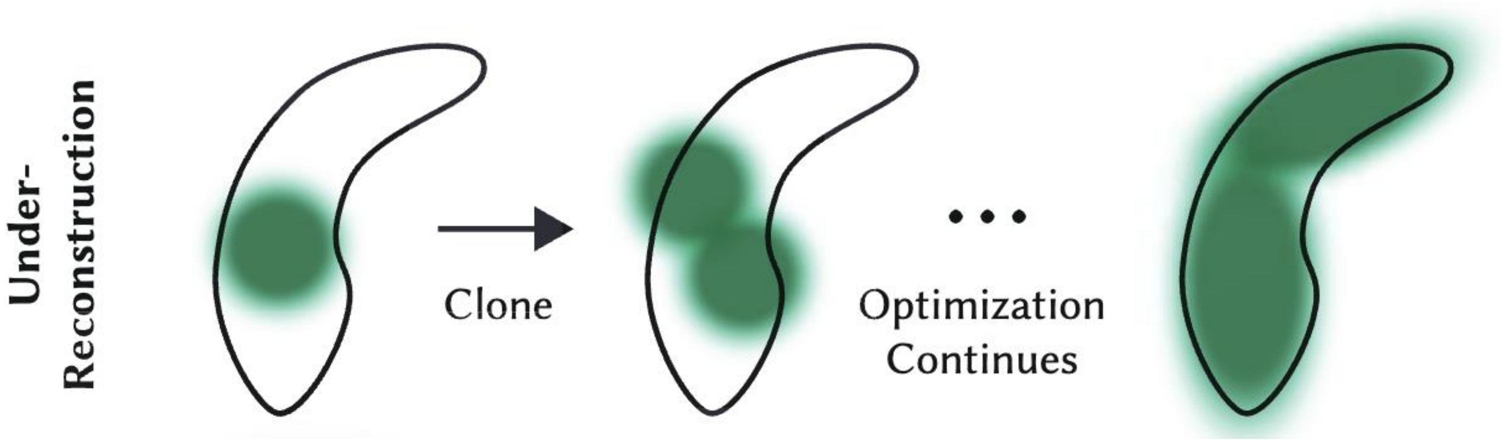
Gaussian Splatting: Why is it fast? A special case of alpha blending

$$I(x) = \sum_i \alpha_i(x) c_i \prod_j^i 1 - a_j(x), \qquad \alpha = oG(x), \qquad G(x) = e^{-0.5(x-\mu)^T \Sigma'^{-1}(x-\mu)}$$

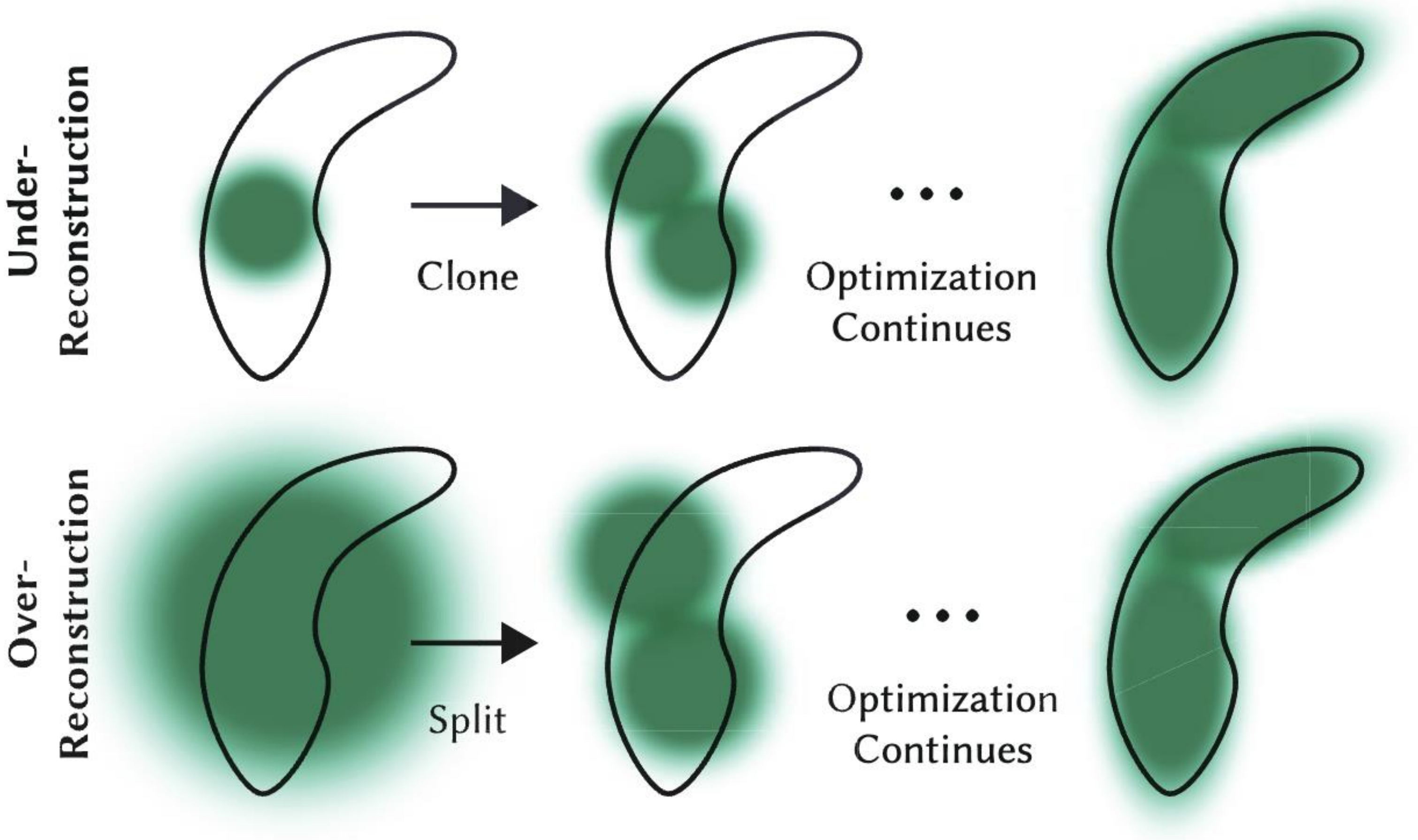


https://blog.42yeah.is/rendering/opengl/2023/12/20/rasterizing-splats.html

Adaptive Control of the Gaussians



Adaptive Control of the Gaussians



Optimization

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

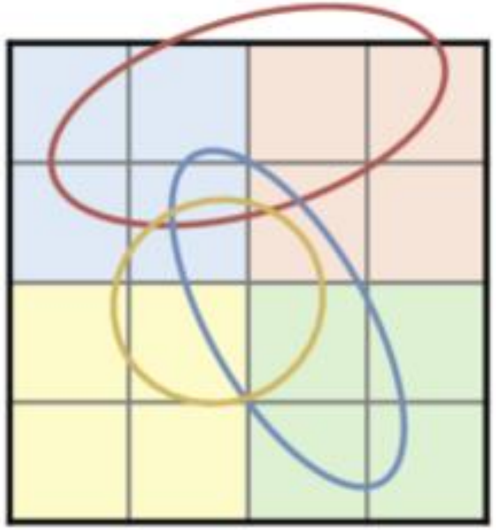
Optimization

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

How to go from 5 FPS to 100+ FPS?

(Using the GPU efficiently)

Optimization

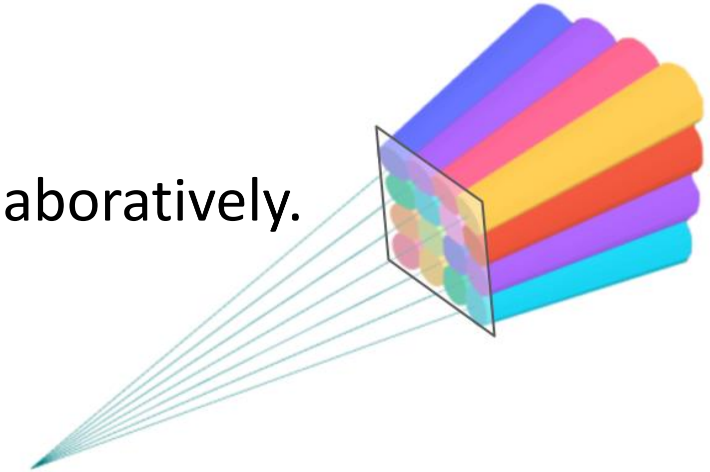


$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

How to go from 5 FPS to 100+ FPS?

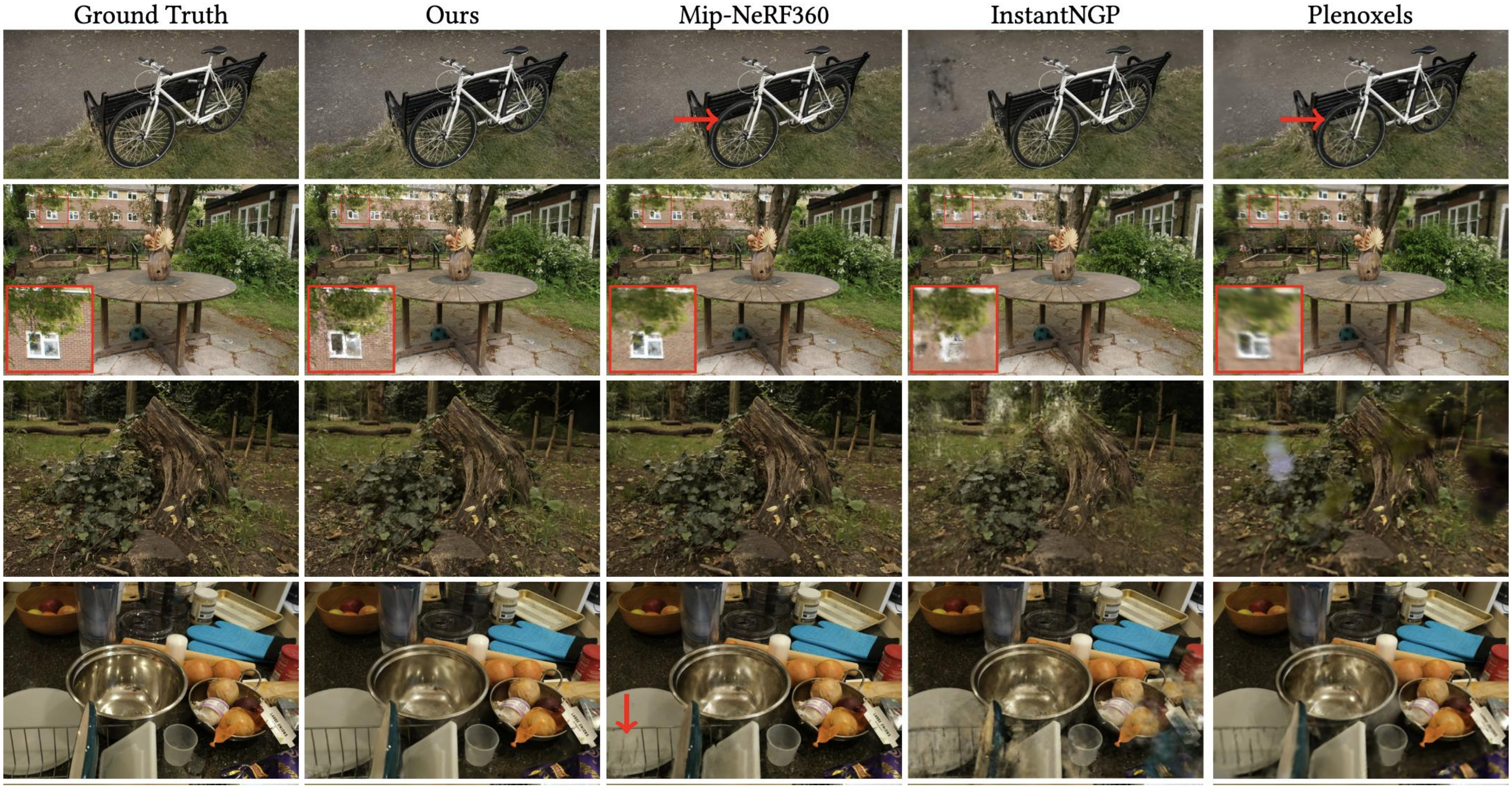
(Using the GPU efficiently)

- 1. Tiling
  - Split the image in 16x16 Tiles – helps threads to work collaboratively.
- 2. Single global sort
  - GPU sorts millions of primitives fast.




Menu
Views
Capture
3D Gaussians
Camera Point view
Metrics
89.37 (11.19 ms)


Menu
Views
Capture
3D Gaussians
Camera Point view
Metrics
57.56 (17.37 ms)
VSync On


Ground Truth
Ours
Mip-NeRF360
InstantNGP
Plenoxels

In this lecture, we will learn …

- **3D Gaussian Splatting (3DGS)** [Kerbl & Kopanas ‘23]
- Applications of 3DGS and addressing its limitations (i.e., dynamic scene, compression, surface reconstruction…)

In this lecture, we will learn …

- **3D Gaussian Splatting (3DGS)** [Kerbl & Kopanas '23]
- Applications of 3DGS and addressing its limitations (i.e., dynamic scene, compression, surface reconstruction…)

Limitations and its follow-up works

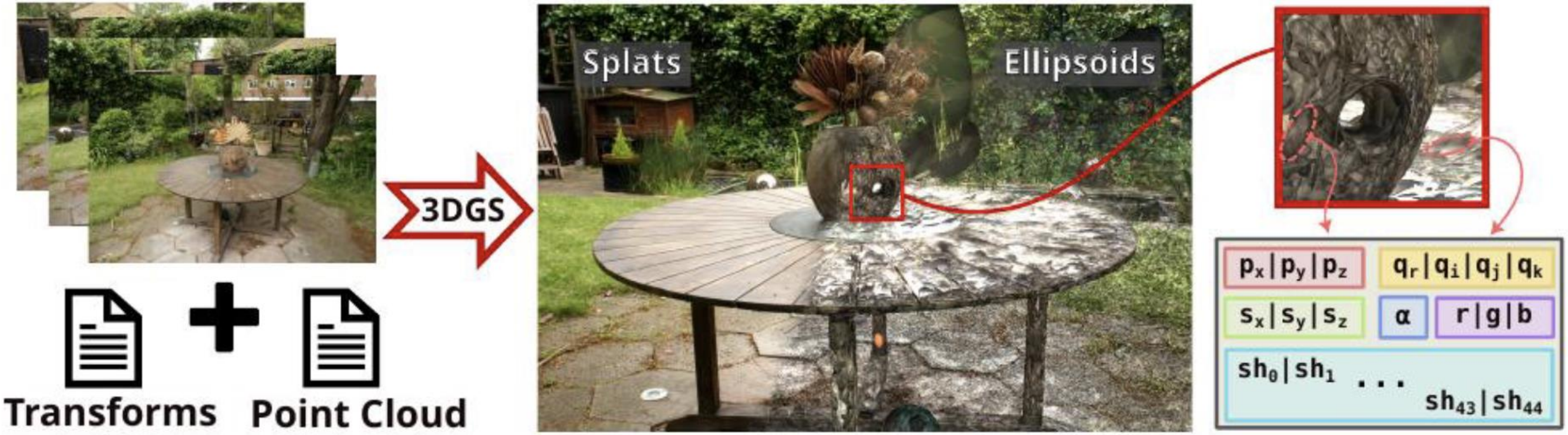
- **3DGS has a high storage cost.**
  - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
  - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
  - Surface Reconstruction (How to obtain surface from gaussian primitives?)

Limitations and its follow-up works

- **3DGS has a high storage cost.**
  - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
  - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
  - Surface Reconstruction (How to obtain surface from gaussian primitives?)

Storage cost of a 3DGS Scene

- 59 x 4 bytes to represent a single Gaussian
- Millions of them!



3DGS Compression – Follow-up works

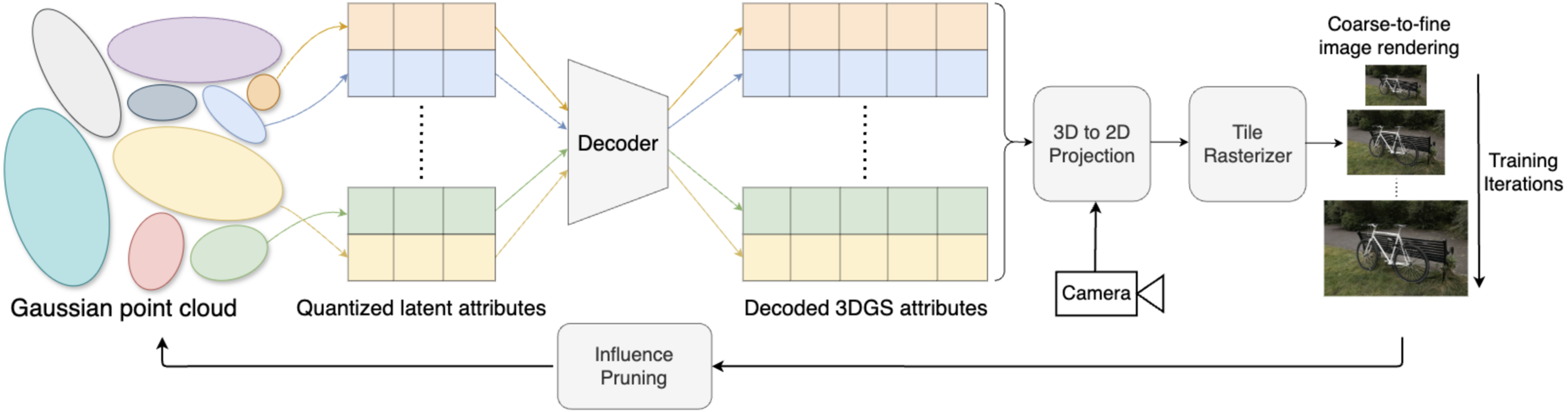
- Compact3D: Smaller and Faster Gaussian Splatting with Vector Quantization
- EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encodings (ECCV 2024)
- LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS (NeurIPS 2024)
- Compact 3D Gaussian Representation for Radiance Field (CVPR 2024)
- Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis (CVPR 2024)
- Reducing the Memory Footprint of 3D Gaussian Splatting (I3D ’24)

3DGS Compression – Follow-up works

- Compact3D: Smaller and Faster Gaussian Splatting with Vector Quantization
- **EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encodings (ECCV 2024)**
- LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS (NeurIPS 2024)
- Compact 3D Gaussian Representation for Radiance Field (CVPR 2024)
- Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis (CVPR 2024)
- Reducing the Memory Footprint of 3D Gaussian Splatting (I3D '24)

https://efficientgaussian.github.io/

EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encodings (ECCV 2024)



- Key components:
  - Quantized embeddings
  - Coarse-to-fine training
  - Influence pruning

Compression results

Limitations and its follow-up works

- **3DGS has a high storage cost.**
  - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
  - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
  - Surface Reconstruction (How to obtain surface from gaussian primitives?)

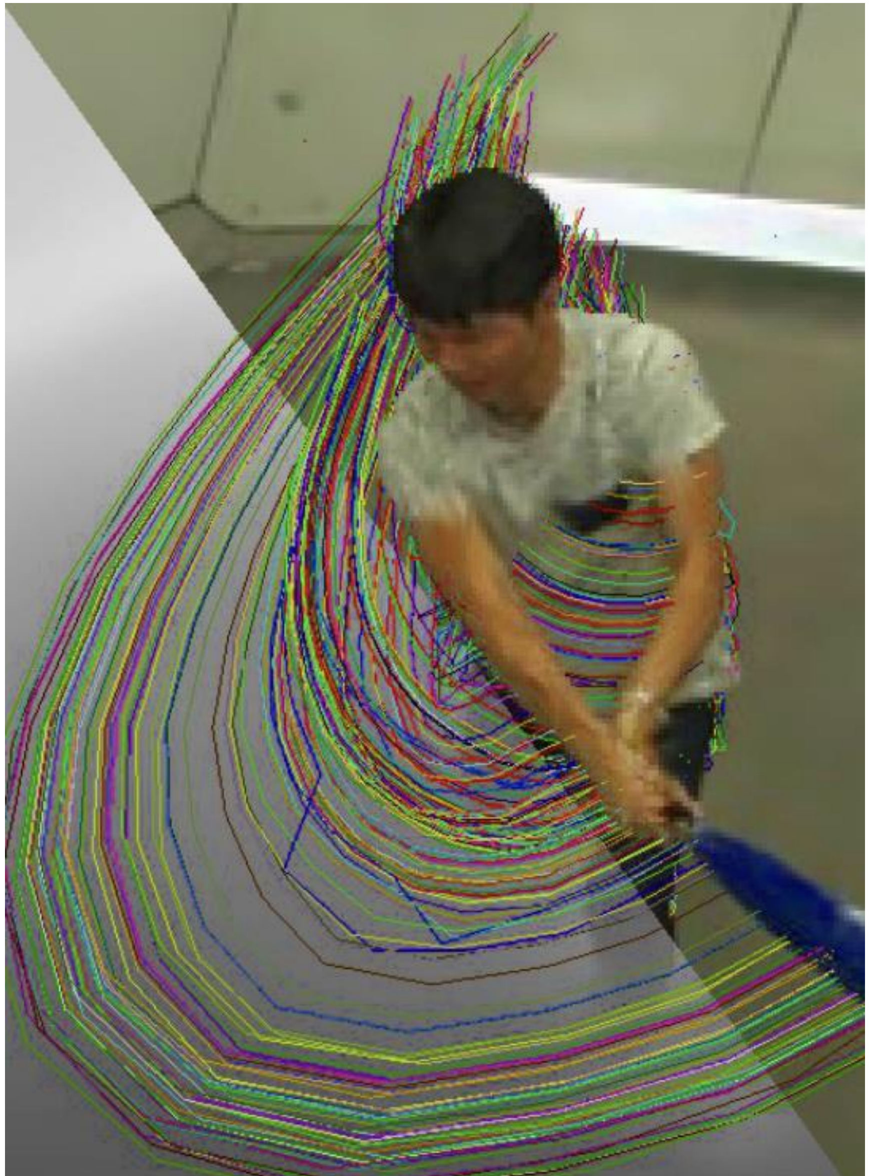
Limitations and its follow-up works

- **3DGS has a high storage cost.**
    - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
    - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
    - Surface Reconstruction (How to obtain surface from gaussian primitives?)

https://dynamic3dgaussians.github.io/

Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis (3DV 2024)

- Fixed / Consistent over time:
  - 3D Size
  - Color
  - Opacity
- Changing over time (per timestep):
  - 3D Center
  - 3D Rotation



Tracking 3D Gaussians over time

Limitations and its follow-up works

- **3DGS has a high storage cost.**
  - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
  - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
  - Surface Reconstruction (How to obtain surface from gaussian primitives?)

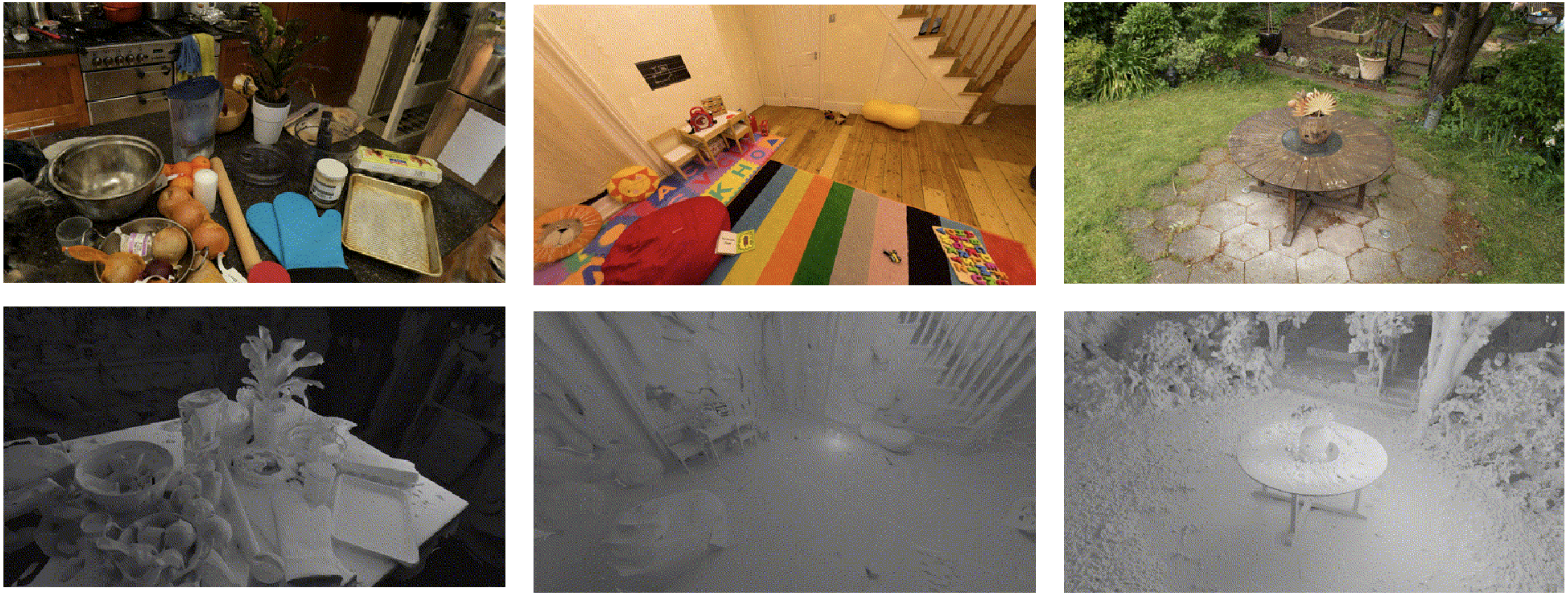
Limitations and its follow-up works

- **3DGS has a high storage cost.**
    - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
    - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
    - Surface Reconstruction (How to obtain surface from gaussian primitives?)

Mesh Extraction

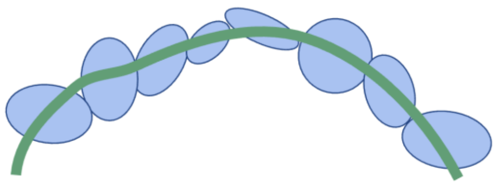
- The naive way of extracting meshes from gaussian splatting is usually done by running TSDF or Marchign Cubes
- Problem!
  - Reconstructed surfaces are not smooth. Hence extracted meshes are very noisy.

SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering (CVPR 2024)

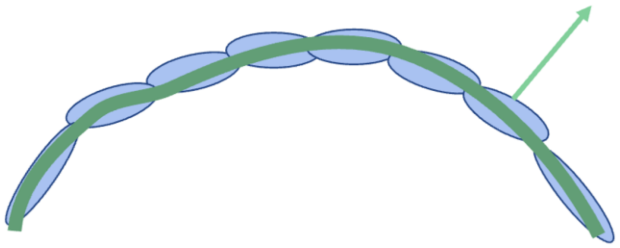


Density constraint: aligning gaussians with the true surface

- Gaussians should have limited overlap and be well-spread on the surface.



- Gaussians should be fully opaque or transparent (otherwise iso surfaces are meaningless)
- Gaussians should be as flat as possible. (One of the three scaling factors should be close to zero.)



In this lecture, we covered …

- **3D Gaussian Splatting (3DGS)** [Kerbl & Kopanas '23]
- Applications of 3DGS and addressing its limitations (i.e., dynamic scene, compression, surface reconstruction…)