Virtual Humans – Winter 24/25

Lecture 10_3 – 3D Gaussian Splatting (3DGS) and Representing Digital Humans with 3DGS

Prof. Dr. Gerard Pons-Moll

University of Tübingen / MPI-Informatics

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

Goal: Reconstructing 3D world from images and videos

Input images

https://3dgstutorial.github.io/

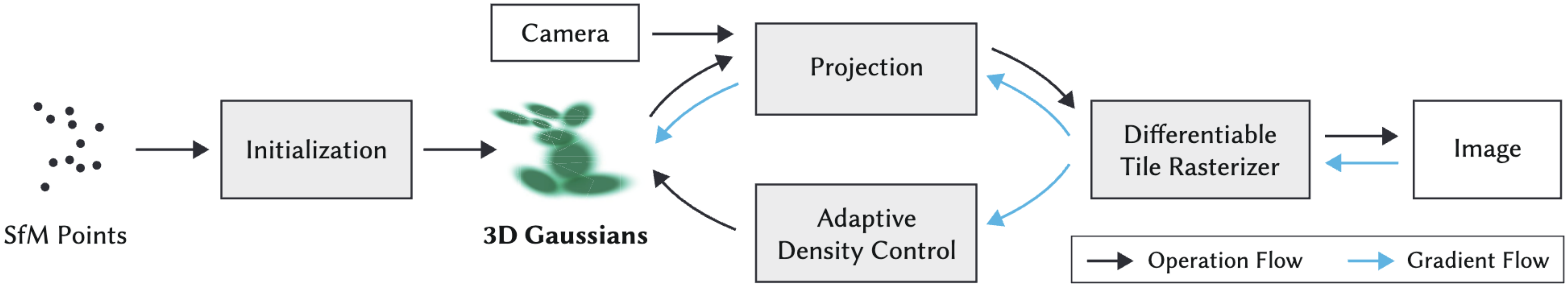

Menu
Views
Capture
3D Gaussians
Camera Point view
Die Aufnahme wurde begonnen
Metrics
78.72 (12.70 ms)

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

$$G(x) = e^{-\frac{1}{2}(x)^T\Sigma^{-1}(x)}$$

How to parametrize 3D Gaussian?

3D Gaussian parametrized by:
- 3D point (mean) $\boldsymbol{\mu}$
- Covariance $\boldsymbol{\Sigma}$
- Opacity alpha α
- Color $\boldsymbol{c}$ – RGB values or spherical harmonics (SH) coefficients.

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.
- $\boldsymbol{\Sigma}$ 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 $\mathbf{\Sigma}$ 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 **Σ** into 2D

3x3 Covariance Matrix

$$\Sigma = RSS^TR^T$$

3x3 Rotation matrix

Diagonal scaling matrix (3 parameters for scale)

2x2 variance matrix

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

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

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

Color of a pixel	Color of each point	Transmittance

NeRF vs Gaussian Splatting

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

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

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

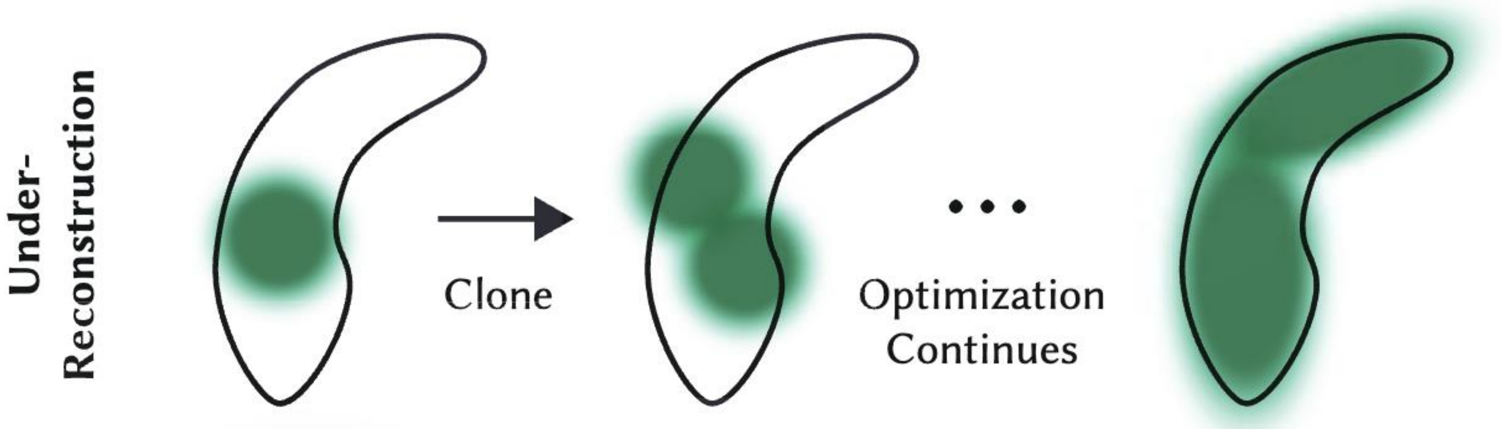
Color of a pixel

Transmittance

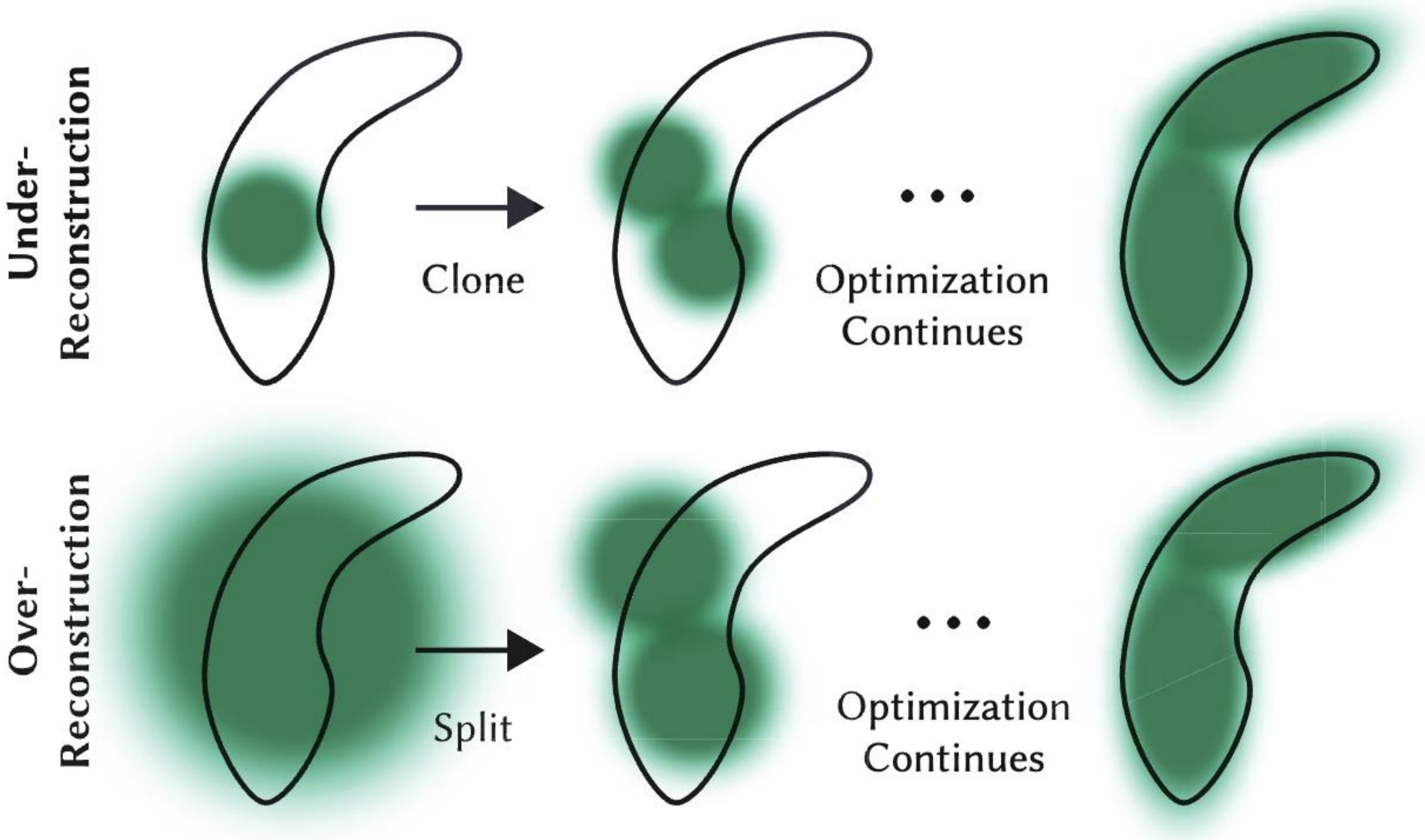
Color of each point

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

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

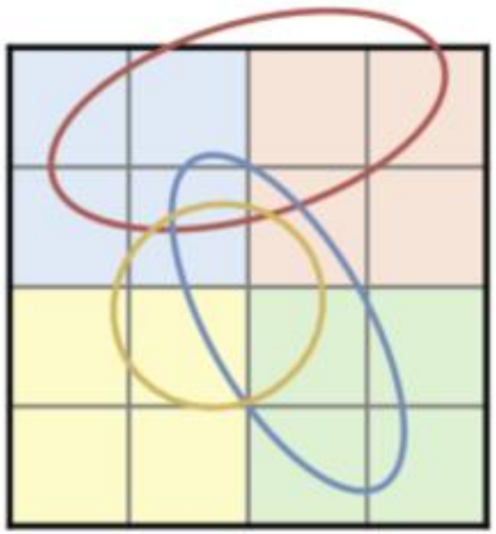
How to go from 5 FPS to 100+ FPS?

(Using the GPU efficiently)

https://3dgstutorial.github.io/

Optimization

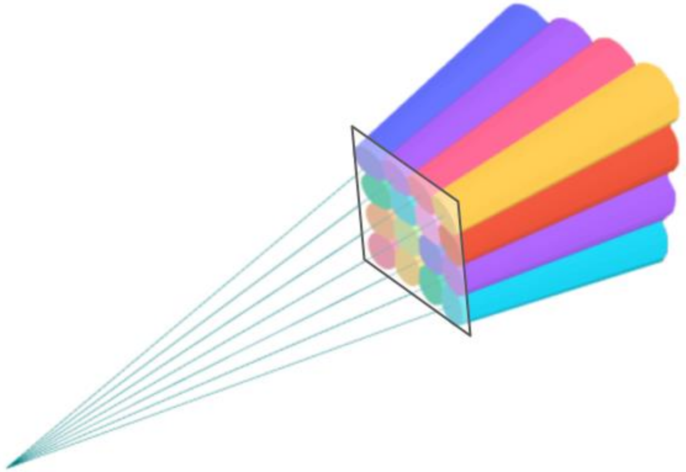
$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$



How to go from 5 FPS to 100+ FPS?

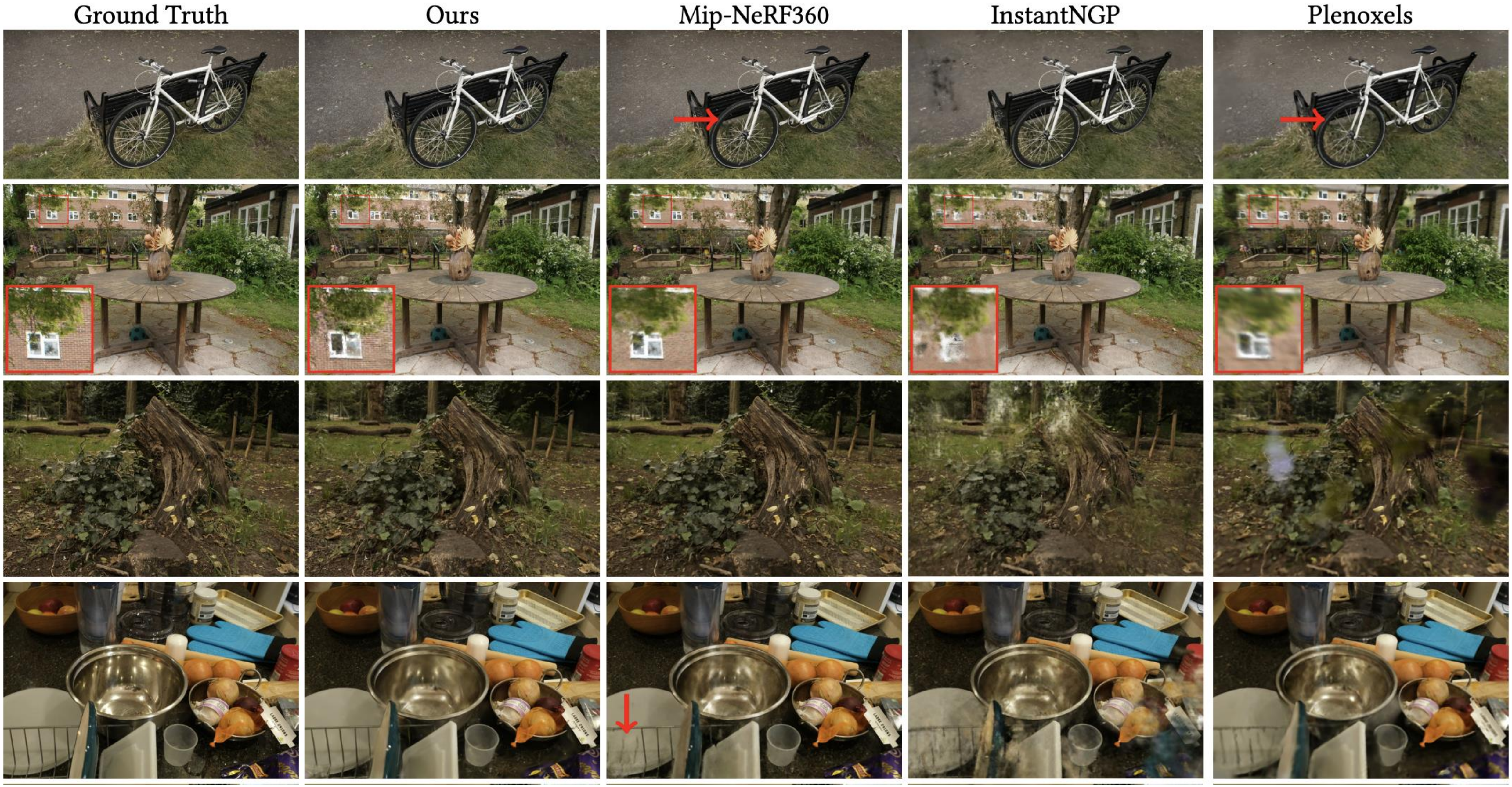
(Using the GPU efficiently)

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




Menu
Views
Capture
3D Gaussians
Camera Point view
Musical Instruments
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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

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?)

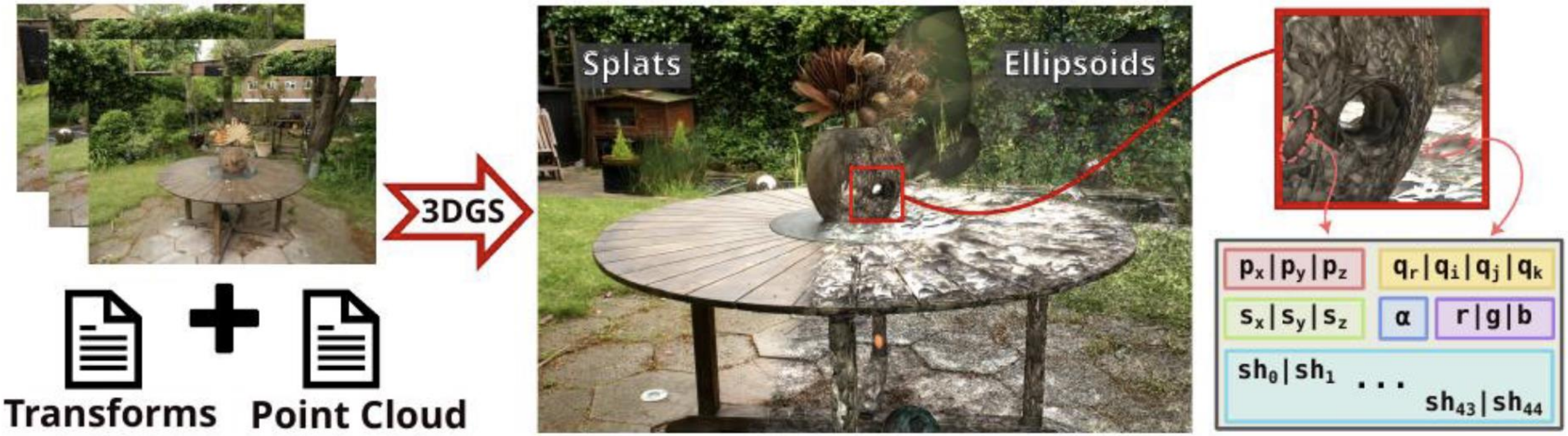
Limitations and its follow-up works

- **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://3dgstutorial.github.io/

Storage cost of a 3DGS Scene

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



3DGS Compression – Follow-up works

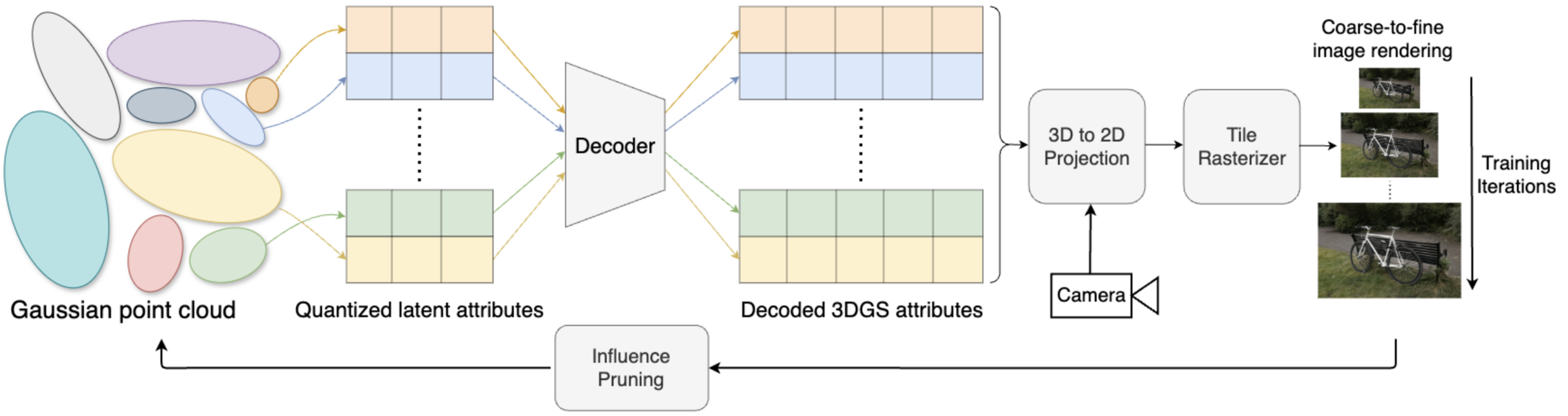
- Compact3D: Smaller and Faster Gaussian Splatting with Vector Quantization
- EAGLES: Efficient Accelerated 3D Gaussians with Lightweight Encodings (ECCV 2024)
- LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS (NeurIPS 2024)
- Compact 3D Gaussian Representation for Radiance 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

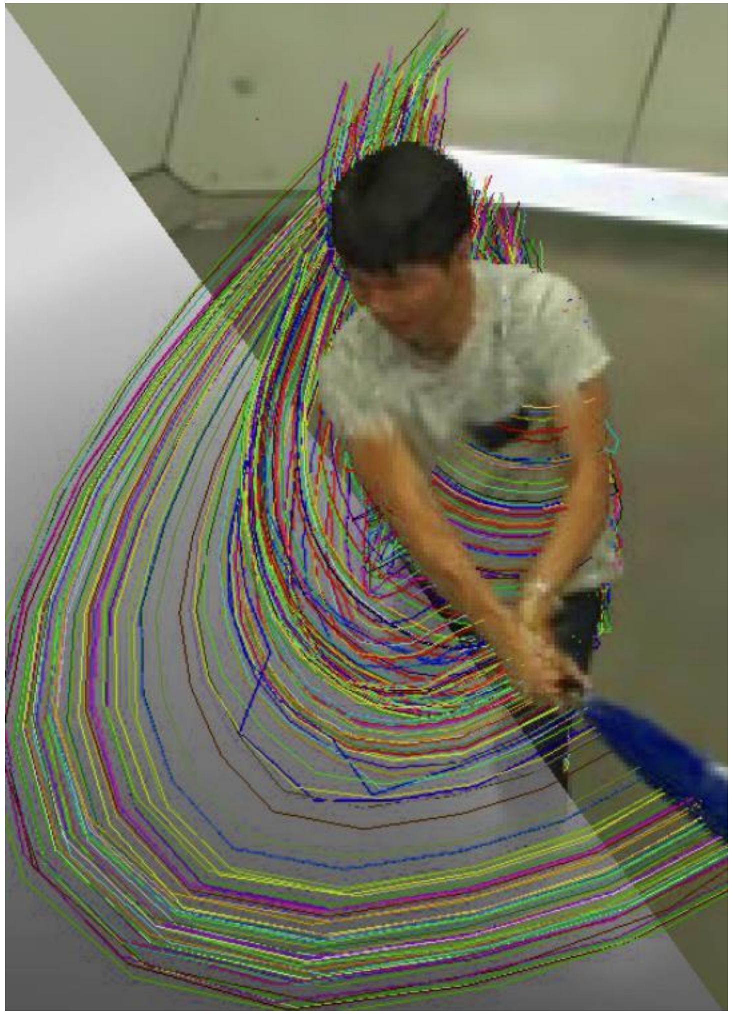
- **3DGS has a high storage cost.**
  - Compression
- **3DGS is a novel view synthesis method (mostly static scenes).**
  - Extending into dynamic scenes - Dynamic 3DGS
- **Unlike meshes, 3DGS does not provide a clean/compact surface.**
  - Surface Reconstruction (How to obtain surface from gaussian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

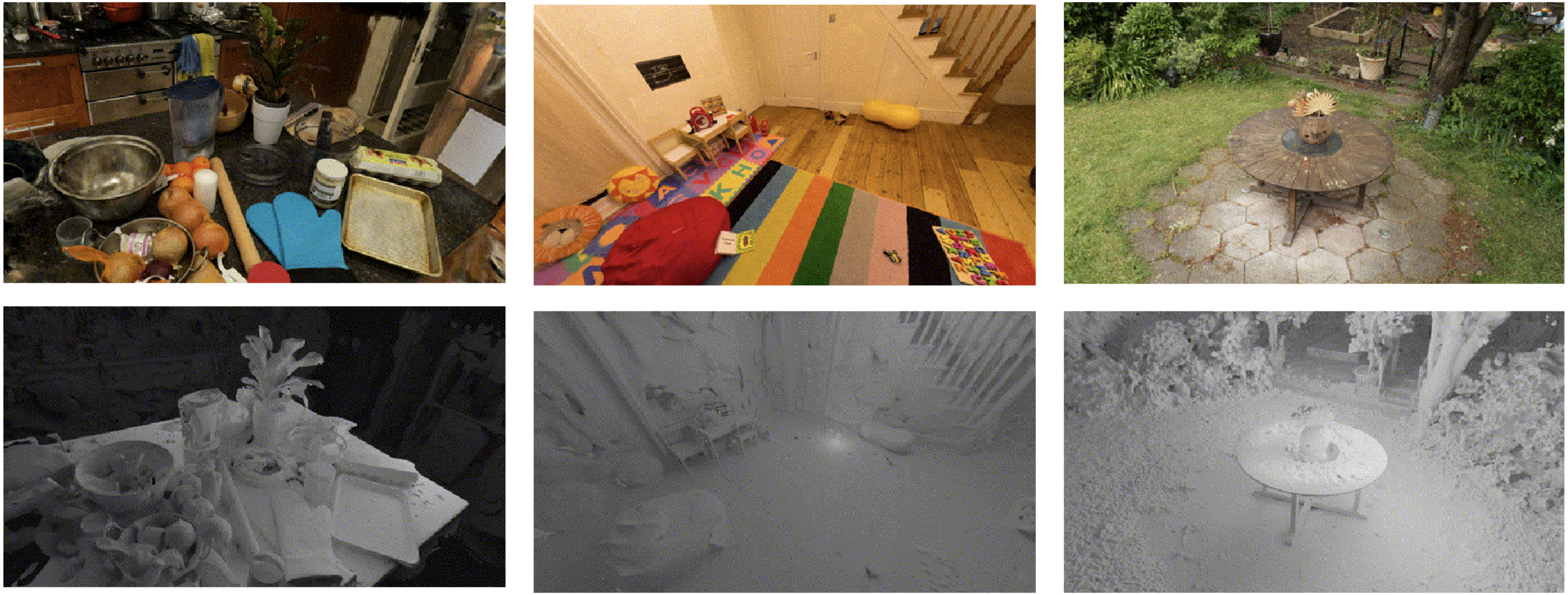
Limitations and its follow-up works

- **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?)

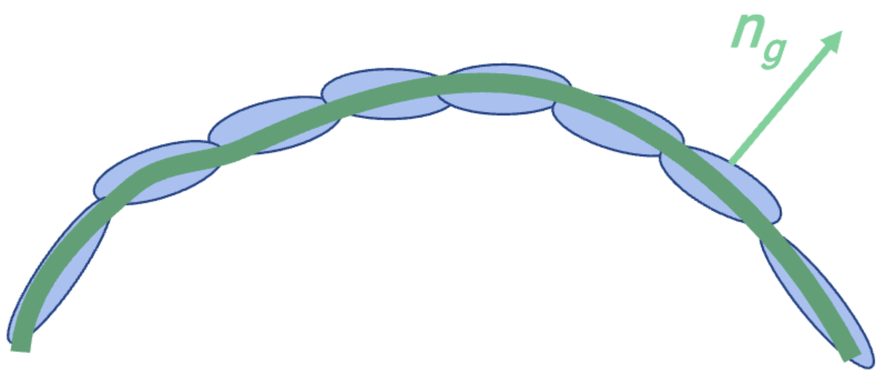
SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering (CVPR 2024)



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.
- Gaussians should be as flat as possible. (One of the three scaling factors should be close to zero.)

$$d(p) = \sum_g \alpha_g \exp\left(-\frac{1}{2}(p-\mu_g)^T \Sigma_g^{-1}(p-\mu_g)\right)$$



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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

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…)
- **Synthetizing/animating photorealistic humans** using 3DGS

Representing virtual humans with Gaussian Splatting

Works on representing digital humans

- **Animatable Gaussians (CVPR 24)**
- Drivable 3D Gaussian Avatars (3DV 25)
- GART (CVPR 2024)
- Human Gaussian Splatting (CVPR 24)
- HUGS (CVPR 24)
- Gaussian Shell Maps (CVPR 2024)
- **GaussianAvatars (CVPR 24)**
- Gaussian Head Avatars (CVPR 2024)
- **PhysAvatar (ECCV 2024)**

  …

Animatable Gaussians

We present ***Animatable Gaussians***, a new avatar representation for creating lifelike human avatars with ***highly dynamic***, ***realistic*** and ***generalized*** details from multi-view RGB videos

Training Data: Multi-view RGB Videos

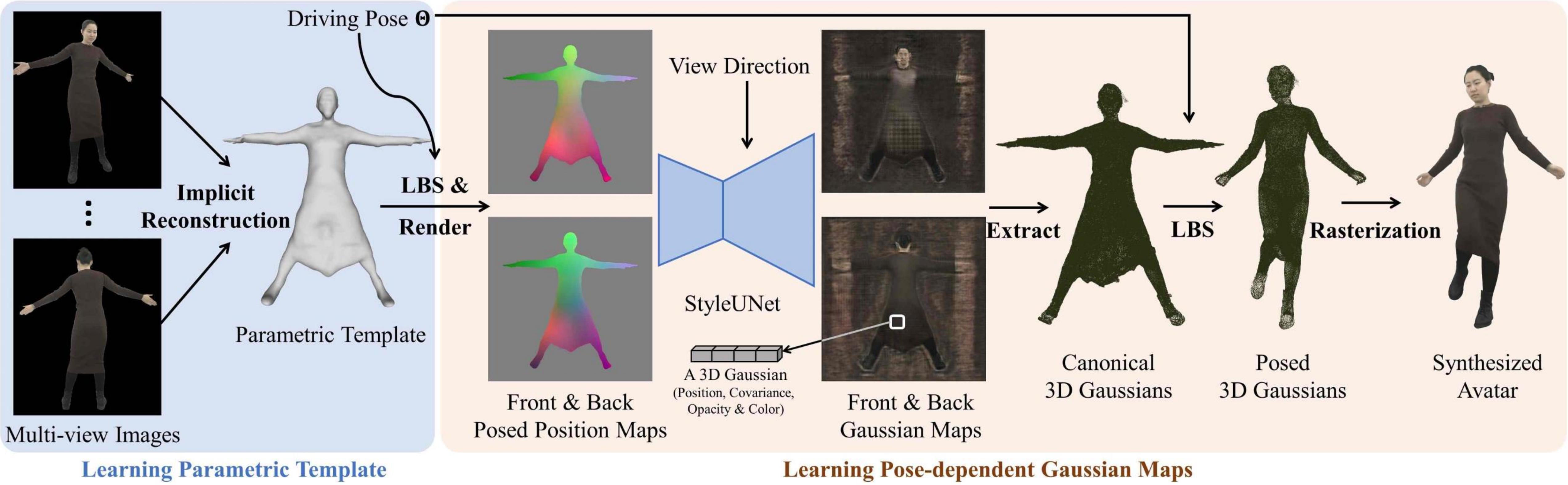
https://animatable-gaussians.github.io/

Avatars animated motions from Amass

https://animatable-gaussians.github.io/

Method

1. Reconstruct a character-specific template from multi-view images
2. Key idea is to predict pose-dependent Gaussian maps through the **StyleUNet**, and render the synthesized avatar by LBS and differentiable rasterization.

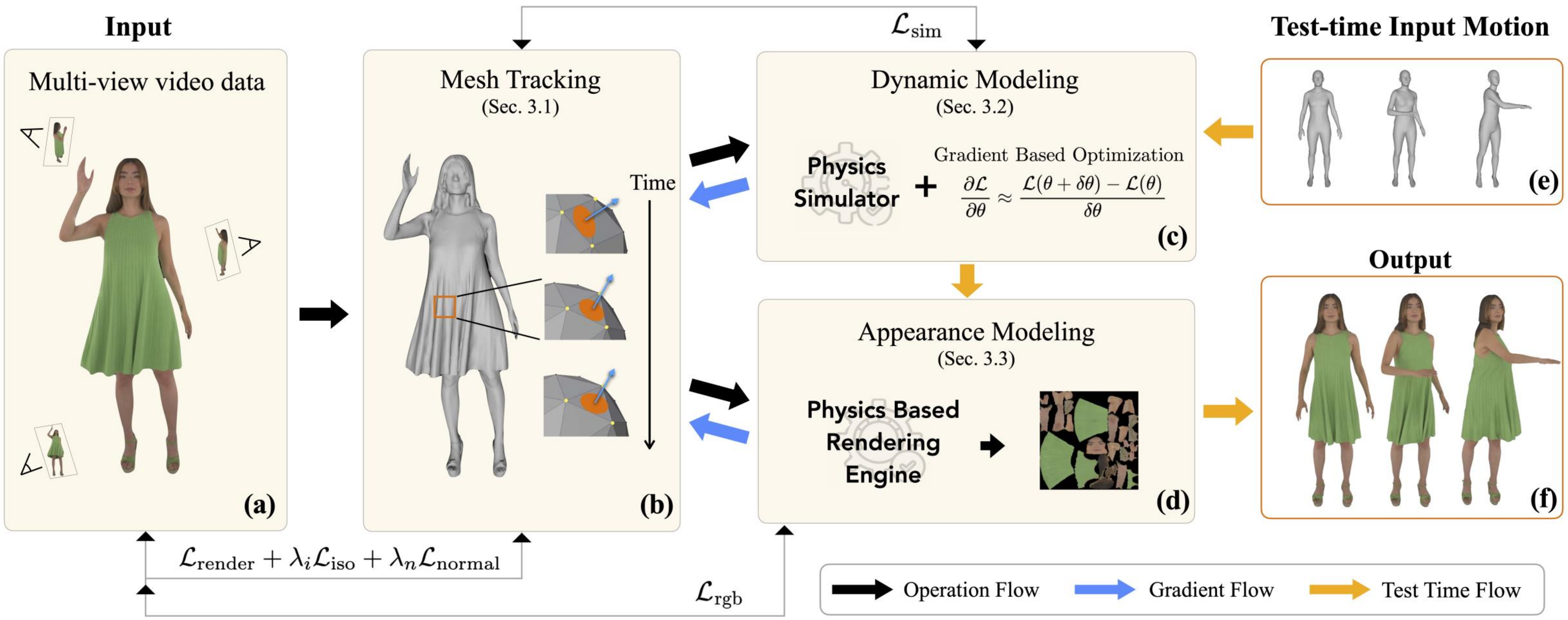


https://animatable-gaussians.github.io/

PhysAvatar: Learning the Physics of Dressed 3D Avatars from Visual Observations

https://qingqing-zhao.github.io/PhysAvatar

Method



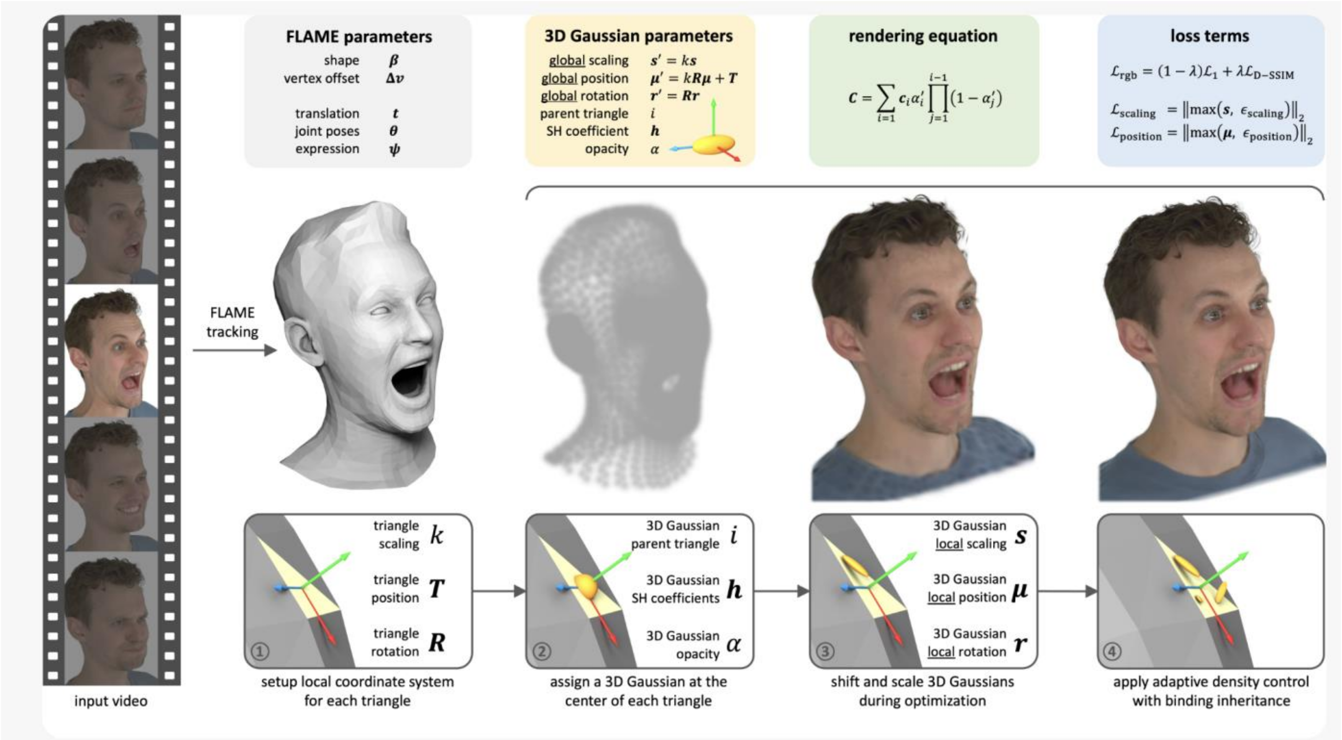
Input
Multi-view video data
(a)
Mesh Tracking
(Sec. 3.1)
Time
(b)
$\mathcal{L}_{\text{sim}}$
Dynamic Modeling
(Sec. 3.2)
Physics Simulator
+
Gradient Based Optimization
$\frac{\partial \mathcal{L}}{\partial \theta} \approx \frac{\mathcal{L}(\theta + \delta\theta) - \mathcal{L}(\theta)}{\delta\theta}$
(c)
Appearance Modeling
(Sec. 3.3)
Physics Based Rendering Engine
(d)
Test-time Input Motion
(e)
Output
(f)
$\mathcal{L}_{\text{render}} + \lambda_i \mathcal{L}_{\text{iso}} + \lambda_n \mathcal{L}_{\text{normal}}$
$\mathcal{L}_{\text{rgb}}$
Operation Flow
Gradient Flow
Test Time Flow

PhysAvatar: Learning the Physics of Dressed 3D Avatars from Visual Observations

https://qingqing-zhao.github.io/PhysAvatar

GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians (CVPR 2024)

- 3D Gaussian splats are rigged to FLAME
- Parameterizing each splat by a local coordinate frame of a triangle



Rigged FLAME model

Source Actor

Animated Avatar

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…)
- **Synthetizing/animating photorealistic humans** using 3DGS