3D Gaussian Splatting (3DGS)

Introduction

Motivation

Reconstructing the 3D world from images or videos.




[...]


Desirable characteristics 3D Representations

1. Accurate — PSNR comparable to MipNeRF360.

2. Fast — 100+ fps & trains in less than 1h.

3. Memory Efficient — Renders on mobile devices (< 6GB VRAM).

4. Practical (i.e., easy to integrate in frameworks) — Many implementations on different Graphics frameworks. Format: easy to standardize (.ply).

3D Gaussian Splatting

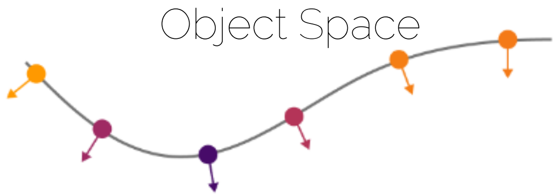
3D Gaussian Splatting Results

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

Background – Point-Based Graphics

Traditional Point-Based Graphics

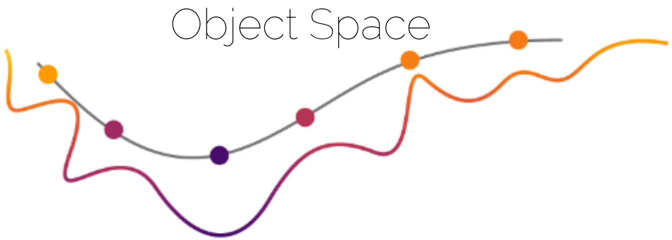
Surface Splatting [Zwicker et al. 2001] using Elliptical Weighted Average (EWA)



1. Considers oriented points (surfels) as discrete samples of a texture function on a surface

Traditional Point-Based Graphics

Surface Splatting [Zwicker et al. 2001] using Elliptical Weighted Average (EWA)

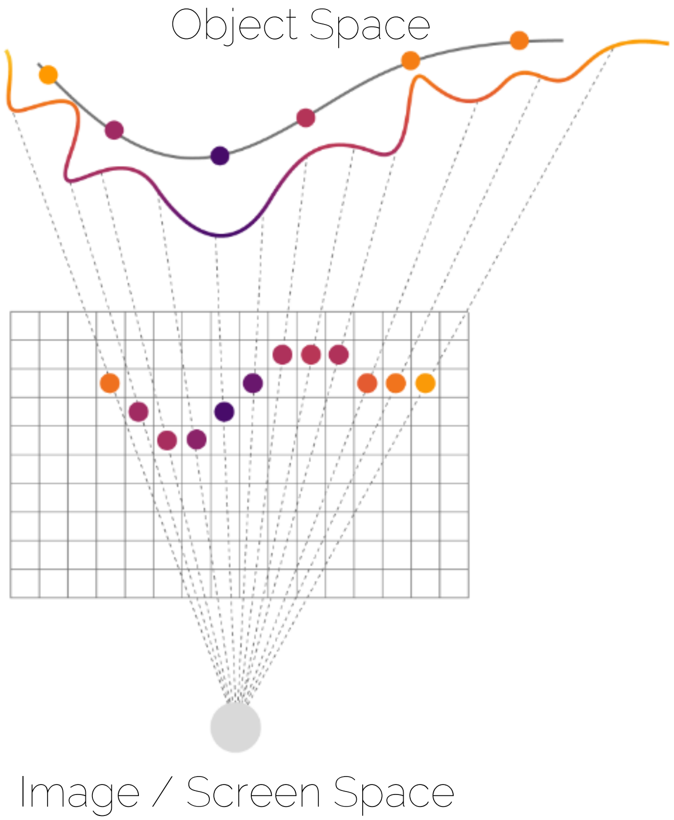


2. A Gaussian reconstruction kernel is used to recover a continuous signal.

Traditional Point-Based Graphics

Surface Splatting [Zwicker et al. 2001] using Elliptical Weighted Average (EWA)

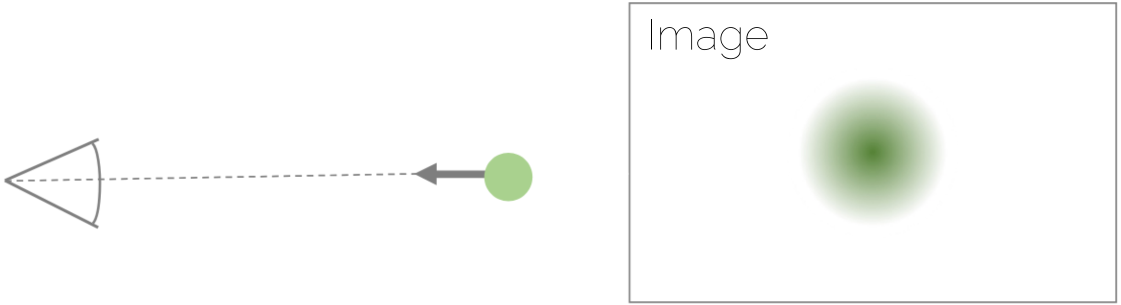
3. Such that we can sample it in screen space.



Traditional Point-Based Graphics

Important outcomes of Surface Splatting:

- Moving camera closer, scales the points so the objects have no holes.
- Slanted normals appear as ellipses, so we can create better edges.
- Each sample can be processed independently.

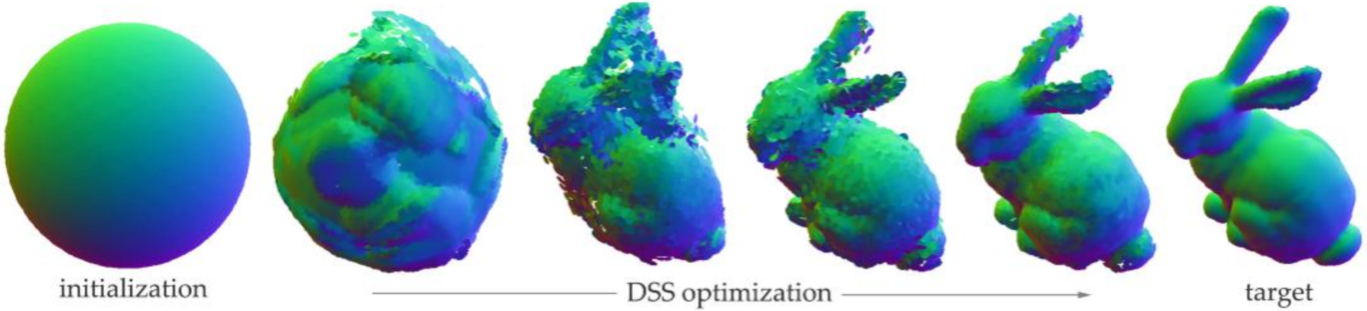


Recent Advances in Point Clouds

Differentiable Surface Splatting [Yifan et al. 2019] showed that this process is end-to-end differentiable

Differentiable Surface Splatting for Point-based Geometry Processing

WANG YIFAN, ETH Zurich, Switzerland
FELICE SERENA, ETH Zurich, Switzerland
SHIHAO WU, ETH Zurich, Switzerland
CENGIZ ÖZTIRELI, Disney Research Zurich, Switzerland
OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland



3D Gaussian Splatting (3DGS)

3D Gaussian Splatting

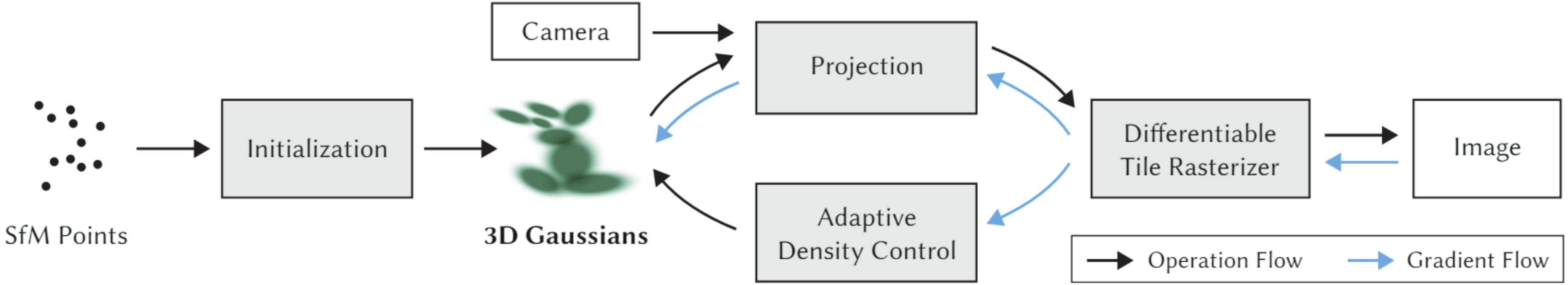
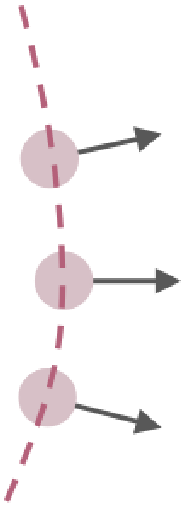
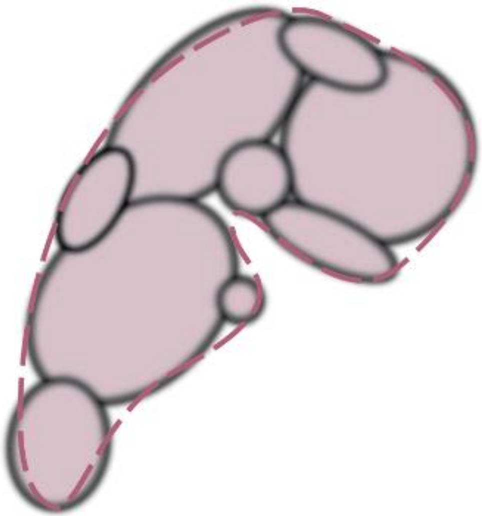


Fig. 2. Optimization starts with the sparse SfM point cloud and creates a set of 3D Gaussians. We then optimize and adaptively control the density of this set of Gaussians. During optimization we use our fast tile-based renderer, allowing competitive training times compared to SOTA fast radiance field methods. Once trained, our renderer allows real-time navigation for a wide variety of scenes.

[Kerbl et al 2023, Siggraph] 3D Gaussian Splatting for Real-time Radiance Field Rendering

Surface Splatting vs. Volume Splatting





Parameters

position $\quad$ normal

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad n = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

σ std dev

f appearance

position $\quad$ covariance

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{xz} \\ & \sigma_y & \sigma_{yz} \\ & & \sigma_z \end{bmatrix}$$

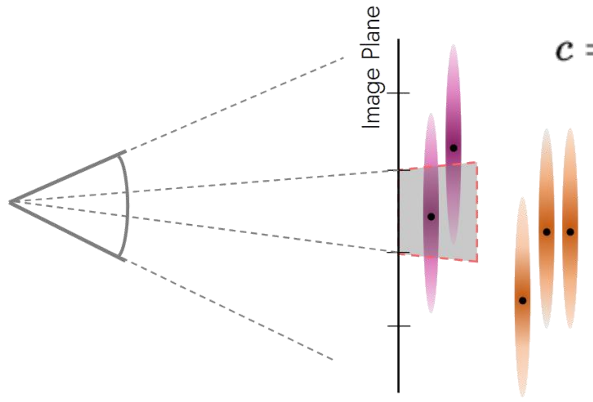
o opacity

SH spherical harmonics

Surface Splatting vs. Volume Splatting

How to blend points in screen space:

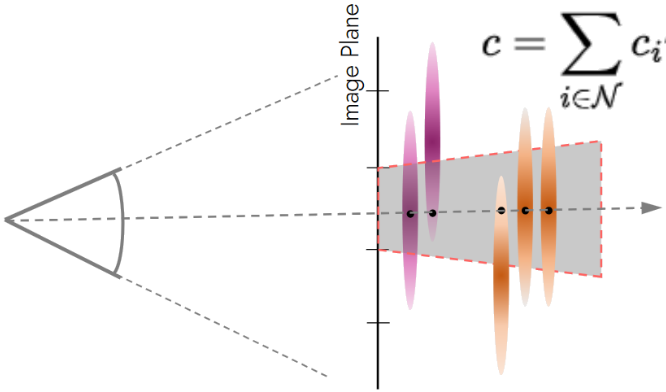
[Zwicker1 '01] / [Yifan '19]



$$c = \frac{\sum_{i \in \Delta\epsilon} c_i w_i}{\sum_{i \in \Delta\epsilon} w_i}$$

Top-Down View

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



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

Top-Down View

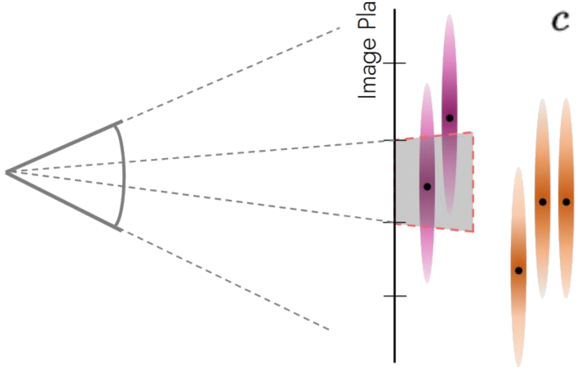
[Zwicker1 '01] Surface Splatting
[Yifan '19] Differentiable Surface Splatting for Point-Based Geometry Proccessing
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Surface Splatting vs. Volume Splatting

How to blend points in screen space:

[Zwicker1 '01] / [Yifan '19]

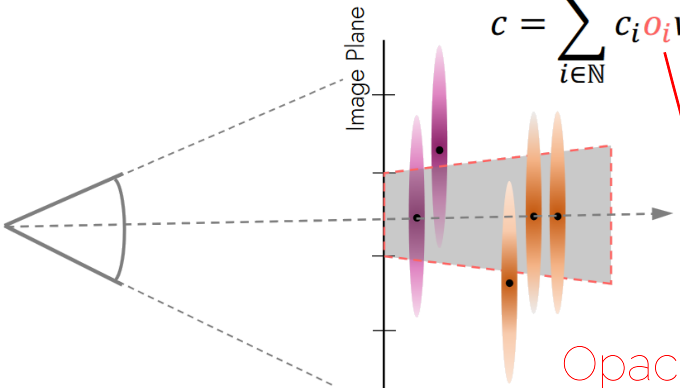
$$c = \frac{\sum_{i\in\Delta\epsilon} c_i w_i}{\sum_{i\in\Delta\epsilon} w_i}$$



Top-Down View

[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

Opacity for each point, allows to let points disappear.

[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

How to blend points in screen space:

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

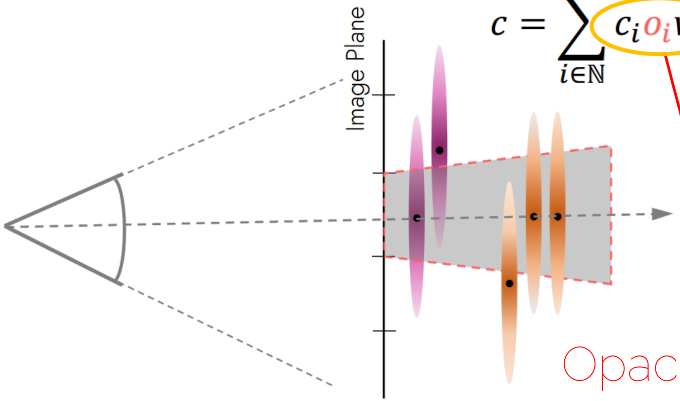
$$c = \sum_{i \in \mathbb{N}} c_i o_i w_i \prod_{j=1}^{i-1} (1 - o_i w_j)$$

Screenshot from NeRF [Mildenhall '20]

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



Top-Down View

Opacity for each point, allows to let points disappear.

Surface Splatting vs. Volume Splatting

What are the benefits of 3D Gaussians?

Initialization:

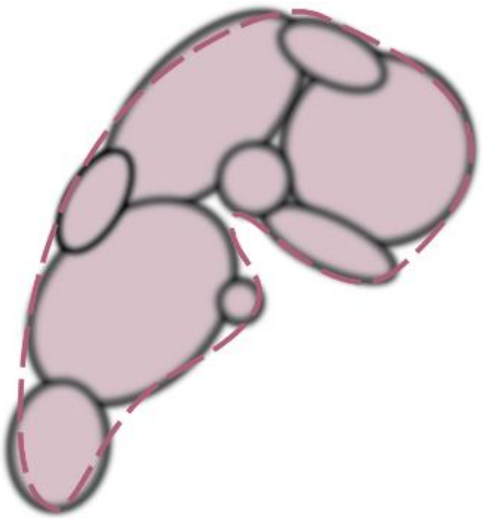
- No Multi-View-Stereo needed → SfM points (no normals) are enough
- Start with isotropic Gaussians

Quality:

- Complicated geometry (i.e., thin structures, vegetation etc.) are more volumetric than surface-like

How are 3D Gaussians rendered?

1. **Sort**: globally based on depth
2. **Splat**: compute the shape of the Gaussian after projection
3. **Blend**: alpha composite



Parameters per Gaussian:

position

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

covariance

$$\Sigma = \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{xz} \\ & \sigma_y & \sigma_{yz} \\ & & \sigma_z \end{bmatrix}$$

o opacity

SH spherical harmonics

Optimization

How can we optimize a covariance matrix?

Problem: Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

Optimization

How can we optimize a covariance matrix?

Problem: Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

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

Solution: For any rotation and scale this is a valid covariance matrix.

Optimization

How can we optimize a covariance matrix?

Problem: Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

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

Solution: For any rotation and scale this is a valid covariance matrix.

R does not optimize well → use quaternions.

Optimization

Now, we're ready to optimize:

1. Initialize isotropic 3D Gaussians to SfM points.

Optimization

Now, we're ready to optimize:

1. Initialize isotropic 3D Gaussians to SfM points.
2. Run SVD to optimize 3D Gaussian parameters.

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

Compare rendered images with target images:

- L1 Color Loss
- Structural Dissimilarity Measure D-SSIM (= 1 – SSIM)

Optimization

→Not enough Gaussians to represent structure realistically.

Optimization

→Not enough Gaussians to represent structure realistically.

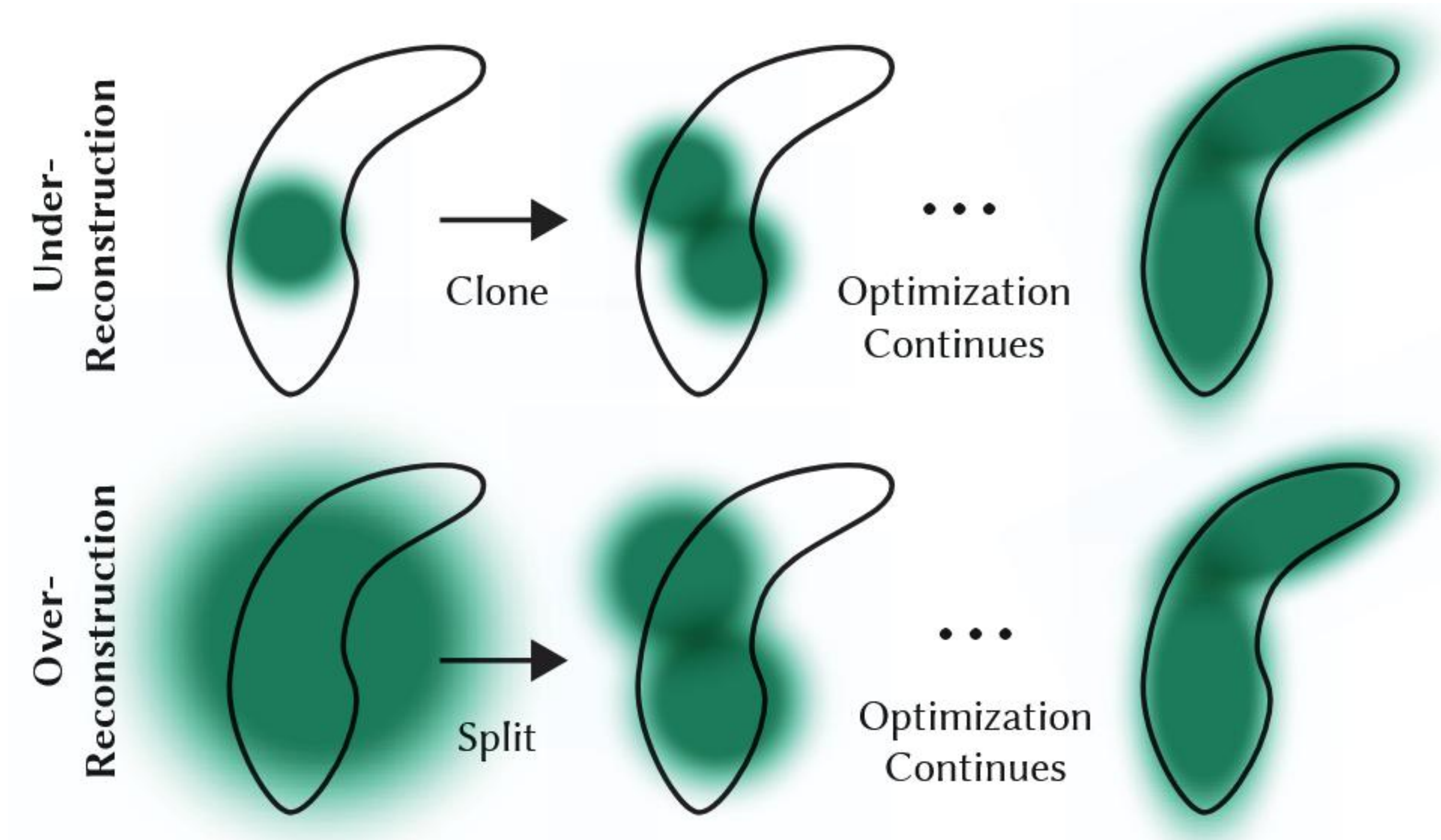
→ Densification of 3D Gaussians in needed.

Densification

Increase the number of points where necessary:

- Points with **high positional gradients** correspond to regions that are not well reconstructed yet.
- Add more Gaussians - Densify these regions.

2 Ways of Densification



→ Increase total volume of the system and number of Gaussians

→ Conserve total volume of the system but increase number of Gaussians

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

Pruning

Idea: Prune unneeded Gaussians, but how to identify them?

Simple, but effective approach:

- Reset opacity to small value every 3000 iterations.
- Prune Gaussians whose opacity remains below a threshold.

Results with Densification and Pruning

→ Poorly reconstructed areas have disappeared. Enough Gaussians to represent structure realistically.

Results with Densification and Pruning

w/o Densification and Pruning

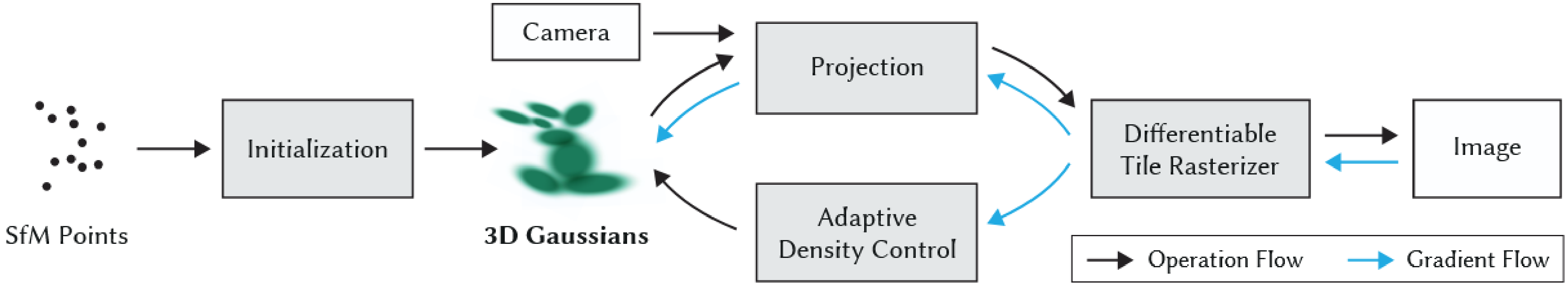
3D Gaussian Splatting

How to get 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.

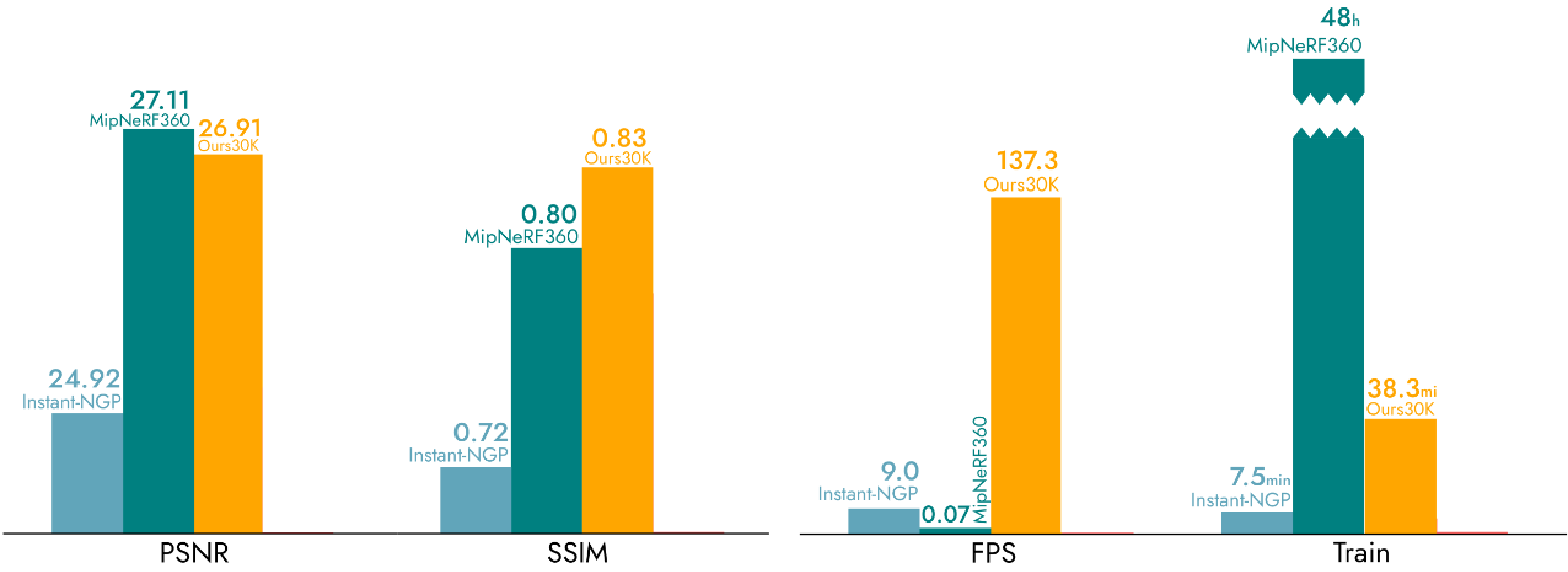
Summary - 3D Gaussian Pipeline



i.e., Densification
and Pruning of
Gaussian primitives

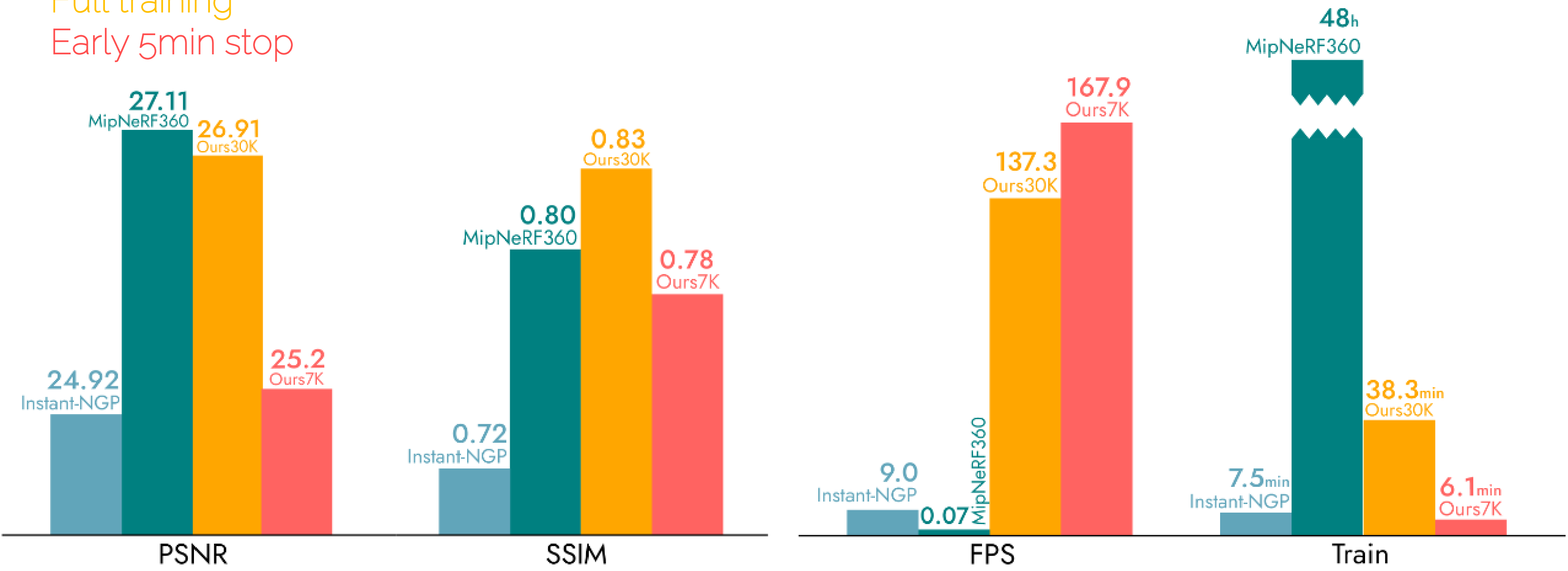
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Evaluation


27.11
MipNeRF360
26.91
Ours30K
24.92
Instant-NGP
PSNR
0.83
Ours30K
0.80
MipNeRF360
0.72
Instant-NGP
SSIM
137.3
Ours30K
9.0
Instant-NGP
0.07
MipNeRF360
FPS
48h
MipNeRF360
38.3mi
Ours30K
7.5min
Instant-NGP
Train

Evaluation

Full training
Early 5min stop



→ 3DGS reaches comparable quality to MipNeRF360 at 2000x faster rendering and 70x faster optimization.

Limitations

1. Handcrafted heuristics for densification and pruning.
2. Popping artifacts because of the mean-based sorting.
3. Representation size:
   a. 3DGS: 350 - 700MB (3-6M of Gaussians)
   b. INGP: 15 - 50MB
   c. MipNeRF360: 8.6MB

Dynamic 3D Gaussians

Dynamic Setting

Challenge: The dynamic setting is highly underconstrained.

→ Good priors are needed.

Dynamic Gaussian Splatting exploded!

~50 papers in the first 7 months!

Dynamic 3D Gaussians:
Tracking by Persistent Dynamic View Synthesis

Jonathon Luiten[1,2] Georgios Kopanas[3] Bastian Leibe[2] Deva Ramanan[1]
[1] Carnegie Mellon University, USA [2] RWTH Aachen University, Germany [3] Inria & Université Côte d'Azur, France
luiten@vision.rwth-aachen.de

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering

Guanjun Wu[1]*, Taoran Yi[2]*, Jiemin Fang[3]†, Lingxi Xie[3], Xiaopeng Zhang[3], Wei Wei[1], Wenyu Liu[2], Qi Tian[3], Xinggang Wang[2]‡
[1] School of CS, Huazhong University of Science and Technology
[2] School of EIC, Huazhong University of Science and Technology [3] Huawei Inc.
{guajuwu, taoranyi, weiw, liuwy, xgwang}@hust.edu.cn
{jaminfong, 198808xc, zxphistory}@gmail.com tian.qi1@huawei.com

An Efficient 3D Gaussian Representation for Monocular/Multi-view Dynamic Scenes

Kai Katsumata Duc Minh Vo Hideki Nakayama
The University of Tokyo, Japan
{katsumata,vmduc,nakayama}@nlab.ci.i.u-tokyo.ac.jp

4D Gaussian Splatting:
Towards Efficient Novel View Synthesis for Dynamic Scenes

Yuanxing Duan[1]* Fangyin Wei[2]* Qiyu Dai[1,4] Yuhang He[1] Wenzheng Chen[3]† Baoquan Chen[1,4]†
[1] Peking University [2] Princeton University [3] NVIDIA [4] National Key Lab of General AI, China

GauFRe: Gaussian Deformation Fields for Real-time Dynamic Novel View Synthesis

Yiqing Liang[†], Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin[†], Lei Xiao
Meta [†] Brown University

DynMF: Neural Motion Factorization for Real-time Dynamic View Synthesis with 3D Gaussian Splatting

Agelos Kratimenos Jiahui Lei Kostas Daniilidis
University of Pennsylvania
Project Page: https://agelosk.github.io/dynmf/

REAL-TIME PHOTOREALISTIC DYNAMIC SCENE REPRESENTATION AND RENDERING WITH 4D GAUSSIAN SPLATTING

Zeyu Yang, Hongye Yang, Zijie Pan, Li Zhang*
Fudan University
https://fudan-zvg.github.io/4d-gaussian-splatting

MD-Splatting: Learning Metric Deformation from 4D Gaussians in Highly Deformable Scenes

Bardienus P. Duisterhof, Carnegie Mellon University, bduister@cmu.edu
Zhao Mandi, Stanford University, mandi@stanford.edu
Yunchao Yao, Carnegie Mellon University, yunchaoy@andrew.cmu.edu
Jia-Wei Liu, National University of Singapore, jiawei.liu@u.nus.edu
Mike Zheng Shou, National University of Singapore, mike.zheng.shou@gmail.com
Shuran Song, Stanford University, shuran@stanford.edu
Jeffrey Ichnowski, Carnegie Mellon University, jeffi@cmu.edu

Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle

Youtian Lin[1] Zuozhuo Dai[2] Siyu Zhu[3] Yao Yao[1]
[1] Nanjing University [2] Alibaba Group [3] Fudan University

Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction

Ziyi Yang[1,2] Xinyu Gao[1] Wen Zhou[2] Shaohui Jiao[2] Yuqing Zhang[1] Xiaogang Jin[1]†
[1] State Key Laboratory of CAD&CG, Zhejiang University [2] ByteDance Inc.

Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis

Zhan Li[1,2]* Zhang Chen[1]† Zhong Li[1]† Yi Xu[1]
[1] OPPO US Research Center [2] Portland State University
lizhan@pdx.edu zhang.chen@oppo.com zhong.li@oppo.com yi.xu@oppo.com
https://oppo-us-research.github.io/SpacetimeGaussians-website/

Dynamic 3DGS Approaches - Overview

We differentiate:

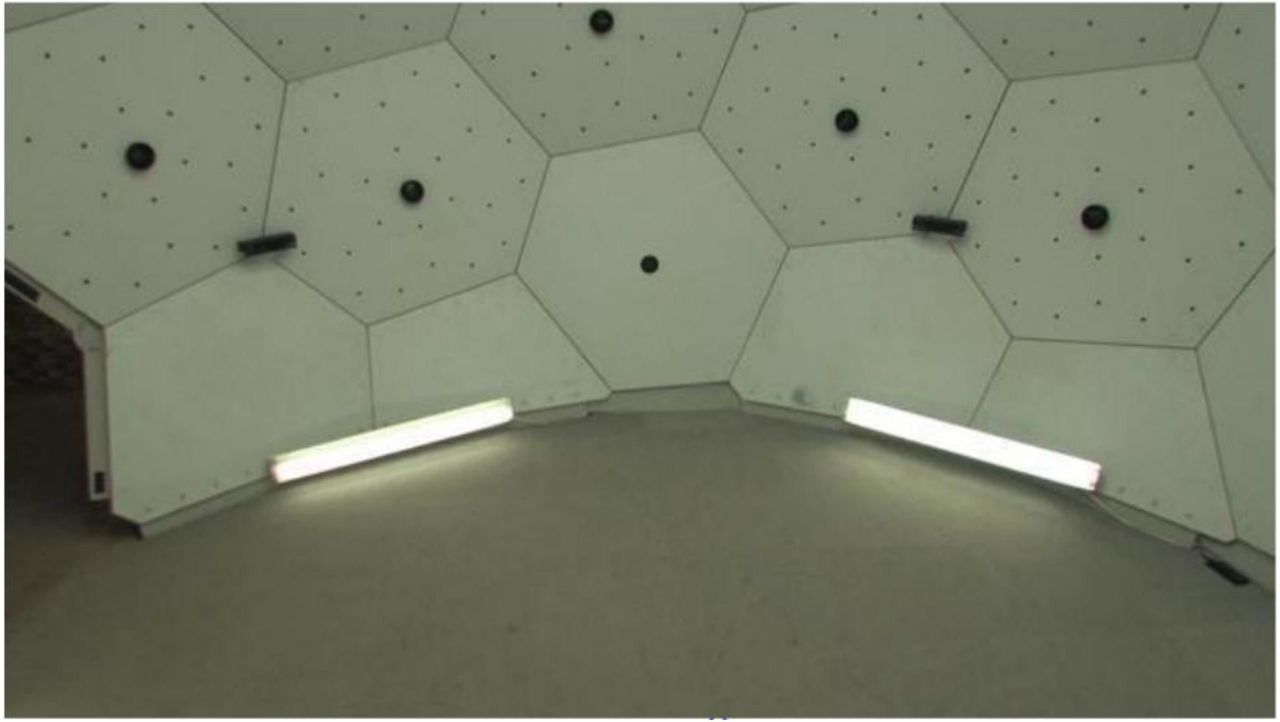
- **Generic** approaches suitable for any type of scene
  - Dynamic 3D Gaussians   per-scene
- Approaches for **heads**
  - Gaussian Avatars   per-scene
  - Neural Parametric Gaussian Avatars   per-scene
  - Avat3r   generalization
- Approaches for **humans**
  - Animatable Gaussians   per-scene

Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

[Luiten '23] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

Good data is essential for per-scene dynamic 3DGS

Multi view input videos are needed.



Panoptic Studio
31 cameras, each
150 frames at 30fps

[Luiten '23] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

Making 3D Gaussians move

Gaussian parameters that are **fixed** over time

- Scale
- Color
- Opacity

→ Optimized on the first frame.

Gaussian parameters that **change** over time

- 3D position
- 3D rotation

→ Optimized for each timestep relative to the previous.

[Luiten '23] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

Physically-Based Priors as Regularizers

- Local rigidity prior

$$\mathcal{L}^{\text{rigid}}_{i,j} = w_{i,j} \left\| (\mu_{j,t-1} - \mu_{i,t-1}) - R_{i,t-1} R^{-1}_{i,t} (\mu_{j,t} - \mu_{i,t}) \right\|_2$$

- Local rotational-similarity prior

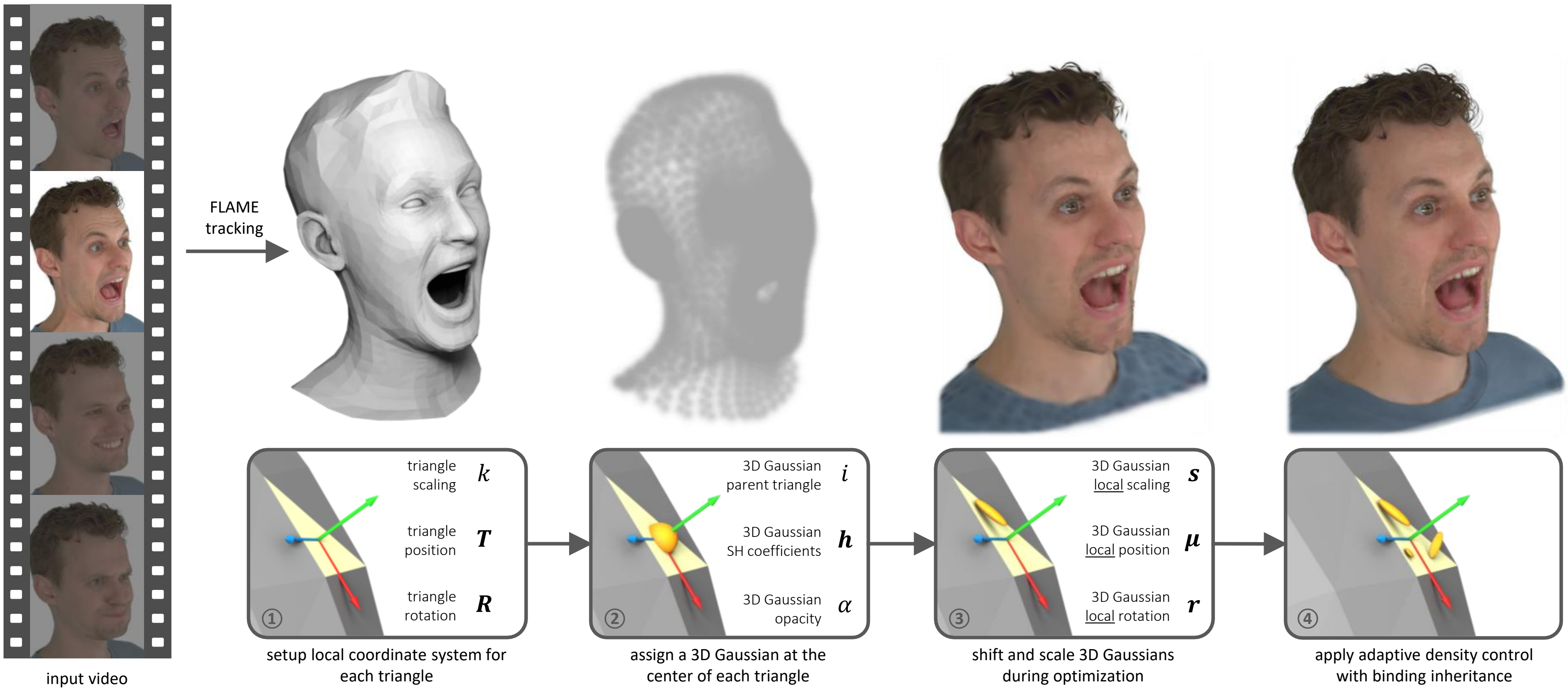
$$\mathcal{L}^{\text{rot}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left\| \hat{q}_{j,t} \hat{q}^{-1}_{j,t-1} - \hat{q}_{i,t} \hat{q}^{-1}_{i,t-1} \right\|_2$$

- Long-term local isometry prior

$$\mathcal{L}^{\text{iso}} = \frac{1}{k|\mathcal{S}|} \sum_{i \in \mathcal{S}} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left| \|\mu_{j,0} - \mu_{i,0}\|_2 - \|\mu_{j,t} - \mu_{i,t}\|_2 \right|$$

[Luiten '23] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis

Controllable Avatars

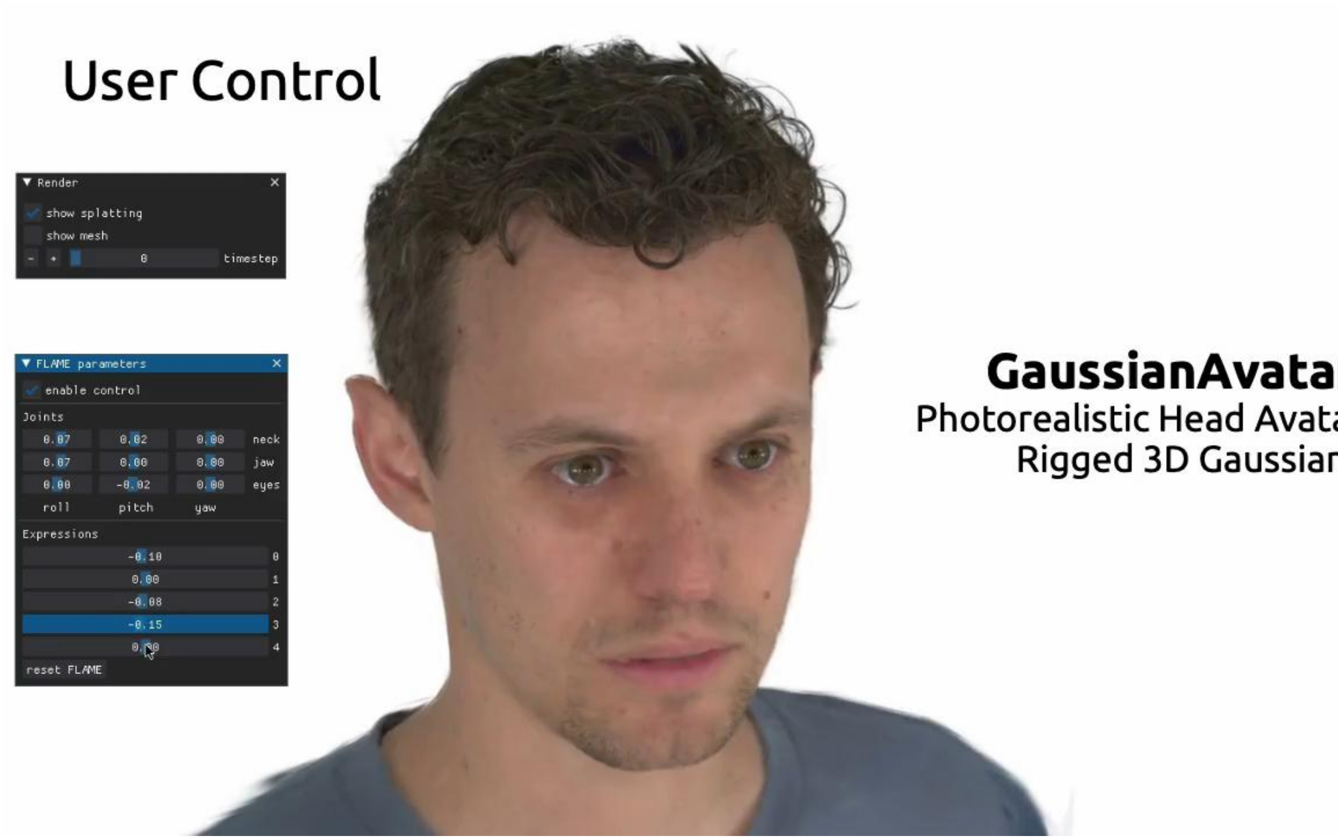


CVPR'24 [Qian et al.] GaussianAvatars

Controllable Avatars

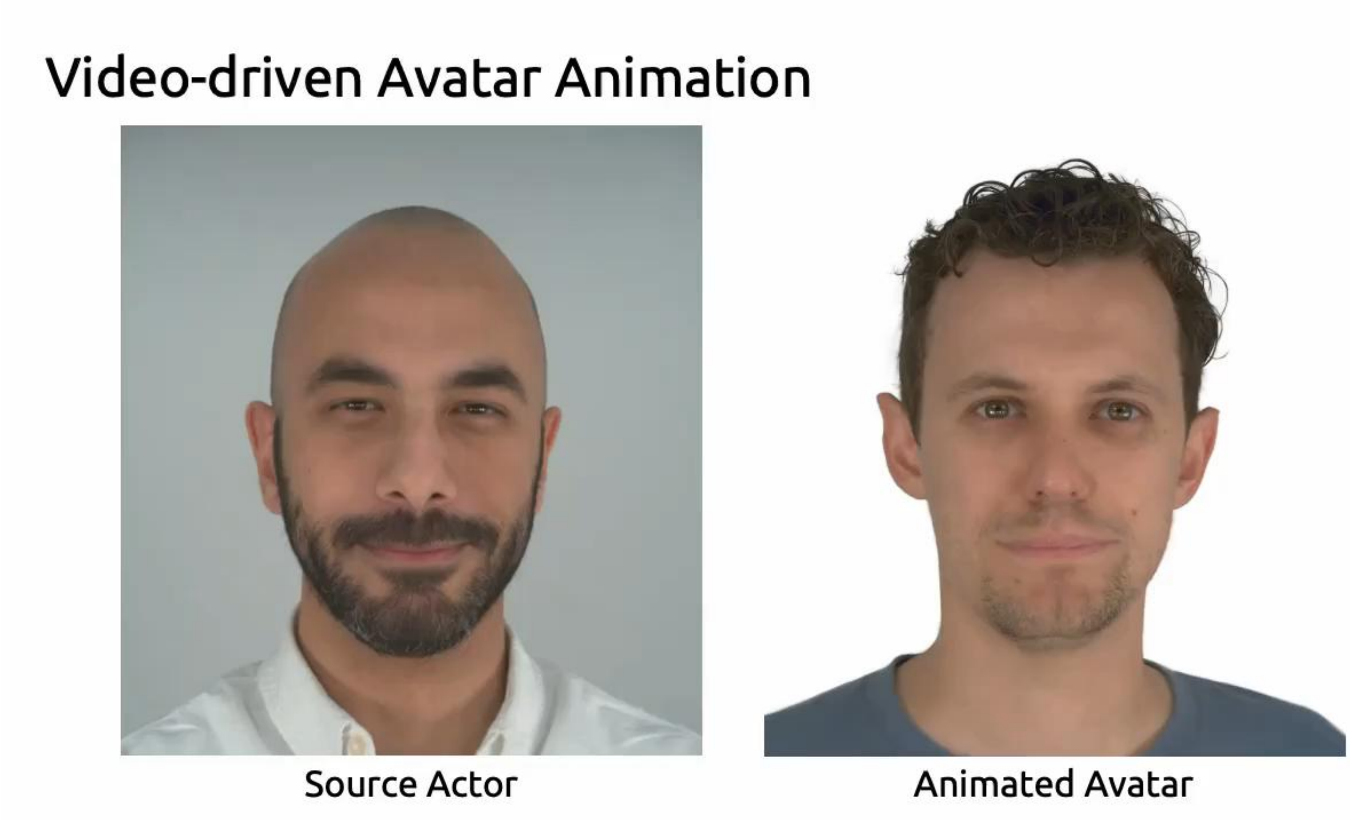
CVPR'24 [Qian et al.] GaussianAvatars

Controllable Avatars



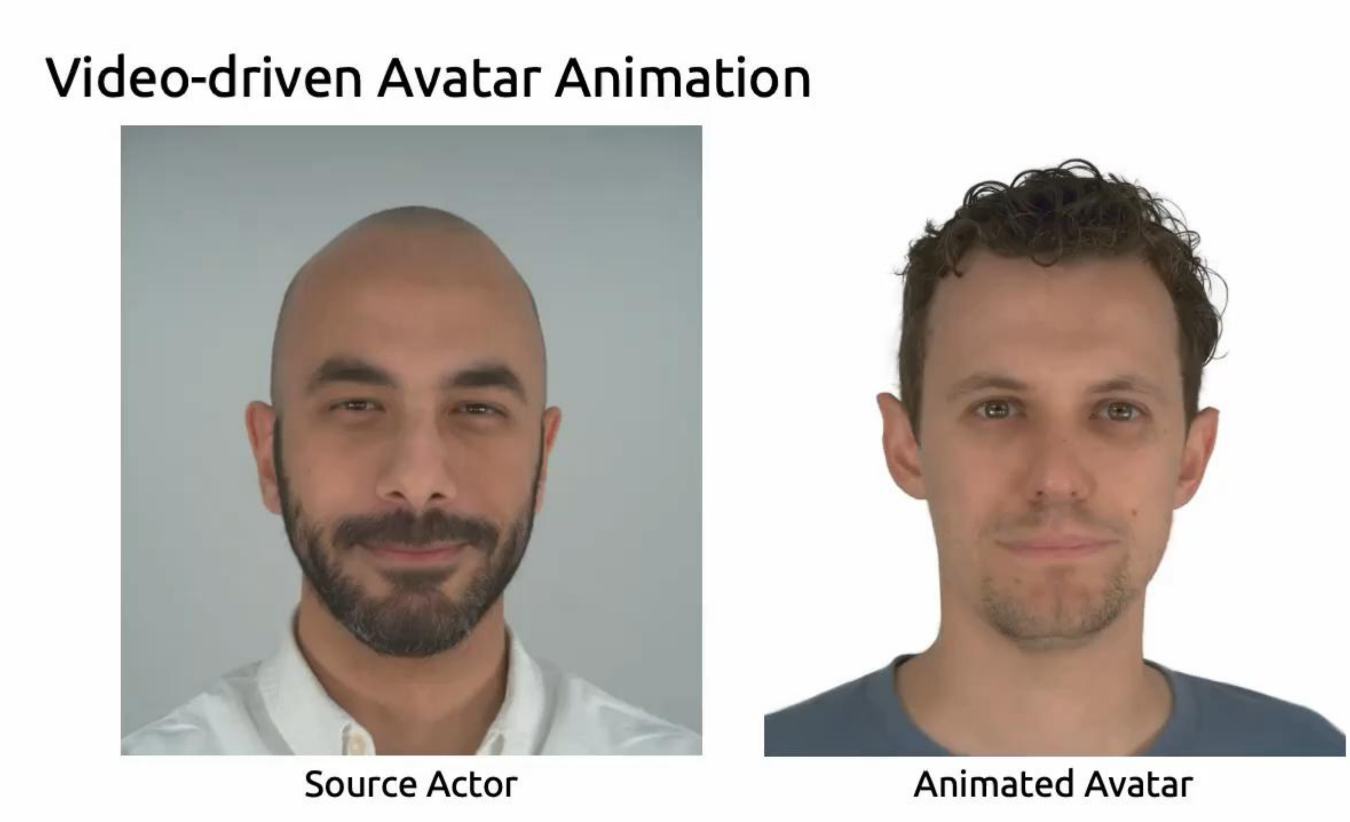
CVPR'24 [Qian et al.] GaussianAvatars

Controllable Avatars



CVPR'24 [Qian et al.] GaussianAvatars

Controllable Avatars



CVPR'24 [Qian et al.] GaussianAvatars

With NPHM Base Model

SIGGRAPH Asia'24 [Giebenhain et al.]: Neural Parametric Gaussian Avatars

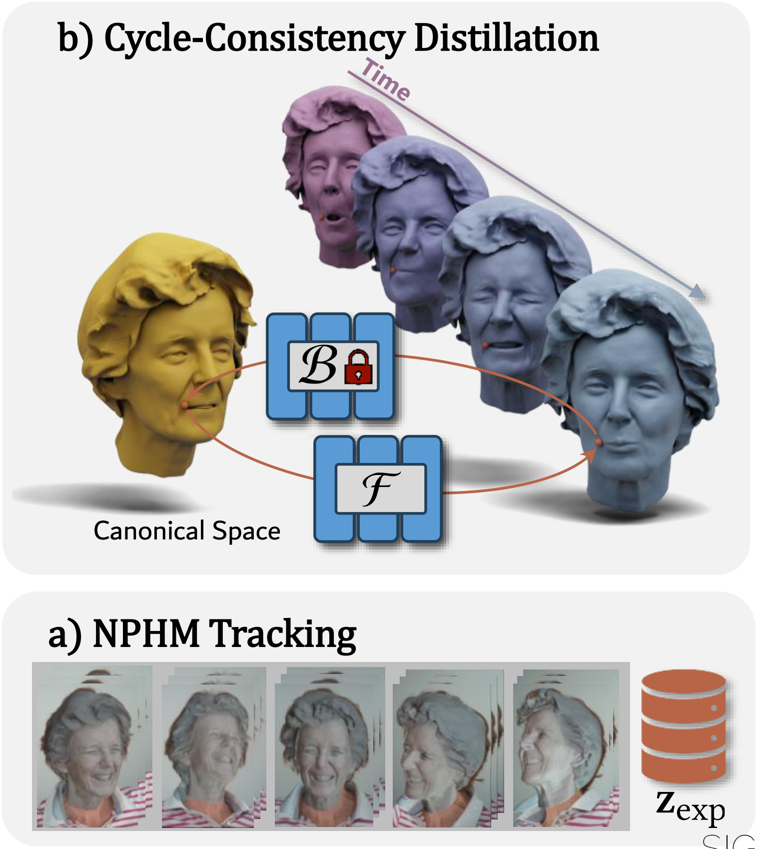
3DGS + NPHM Base Model

SIGGRAPH Asia'24 [Giebenhain et al.]: Neural Parametric Gaussian Avatars

Method Overview

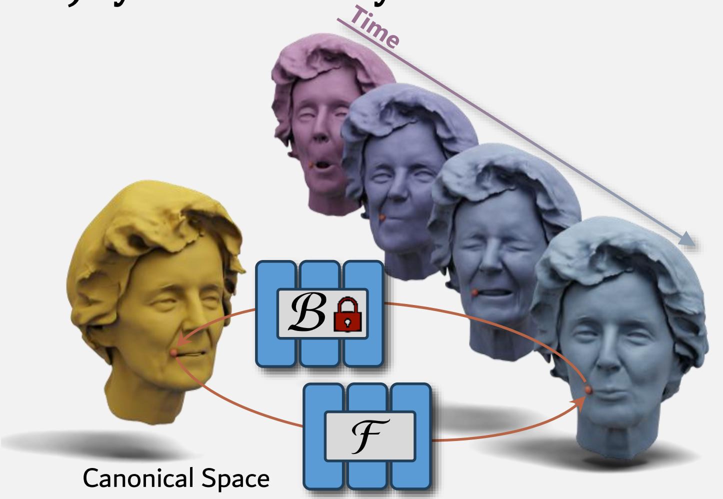
a) NPHM Tracking

Method Overview

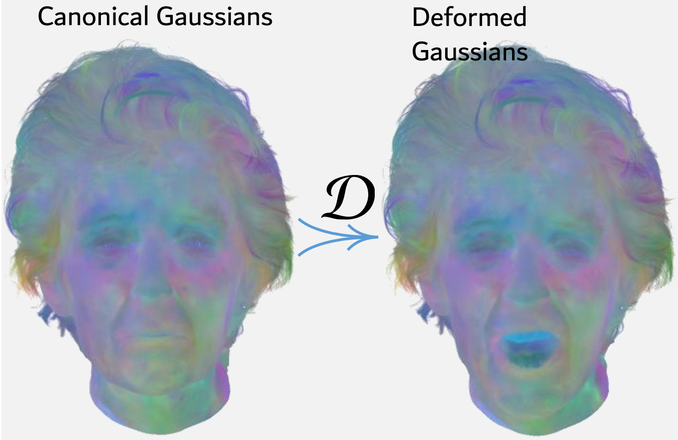


Method Overview

b) Cycle-Consistency Distillation

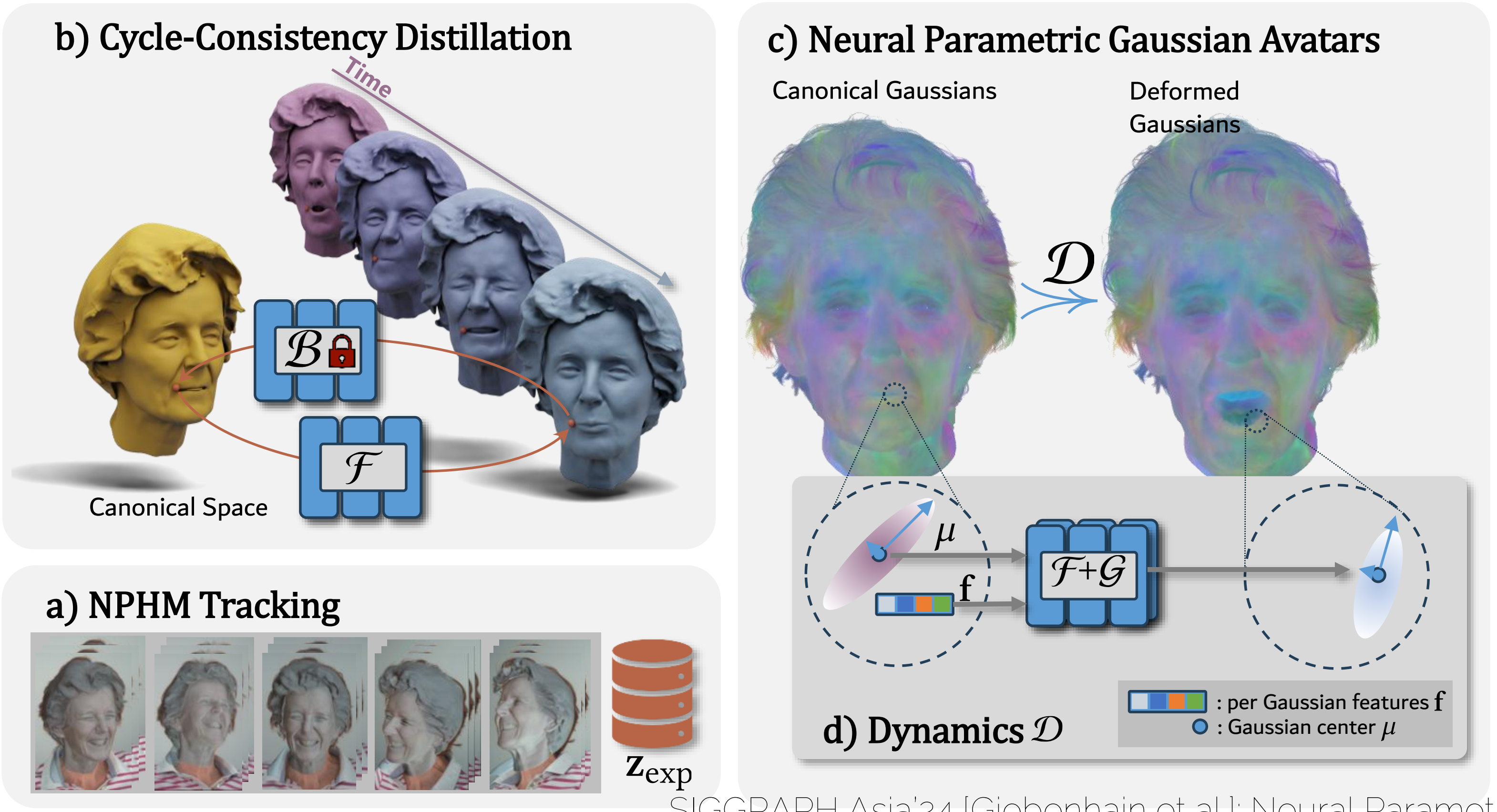


c) Neural Parametric Gaussian Avatars

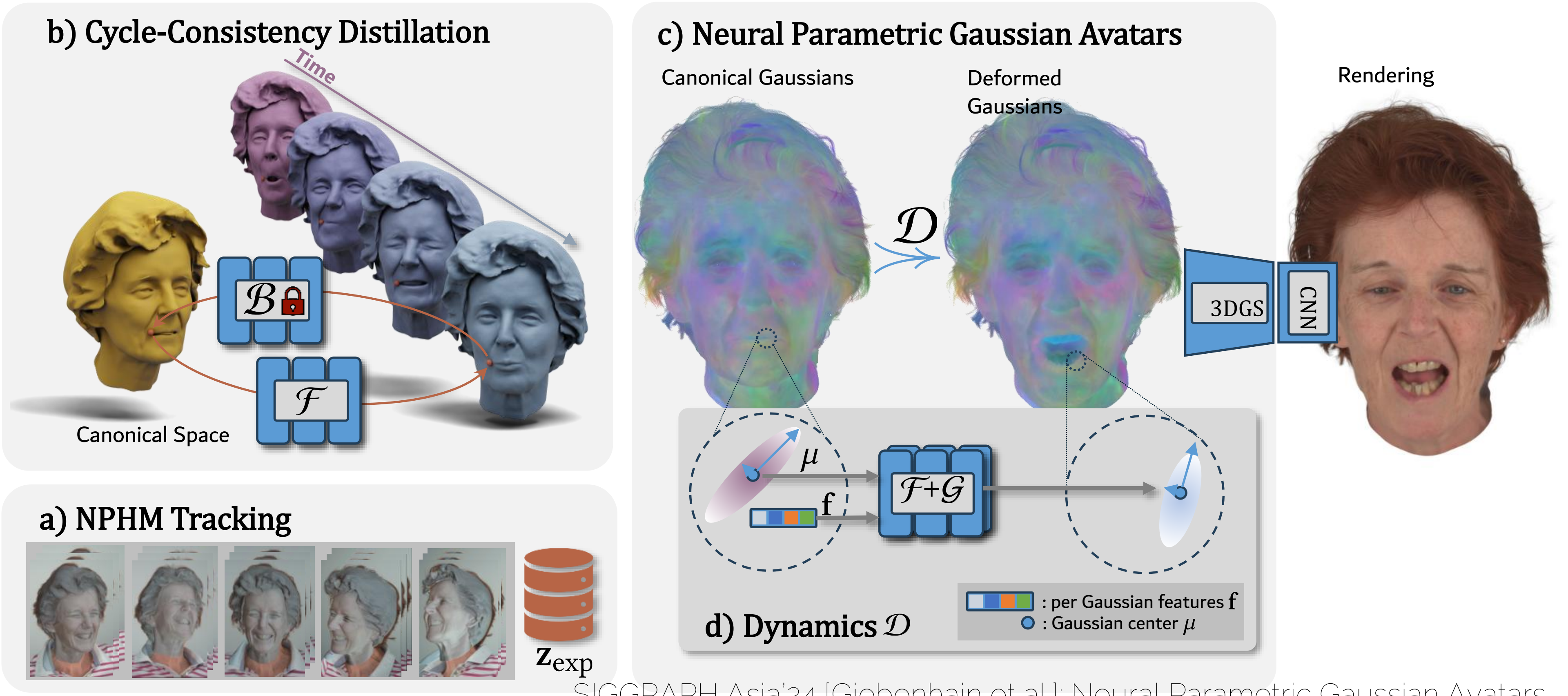


a) NPHM Tracking

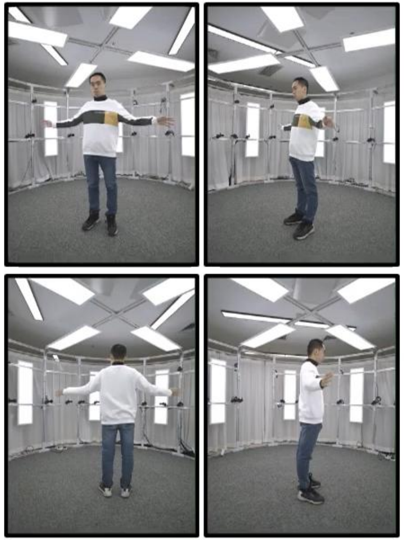
Method Overview



Method Overview

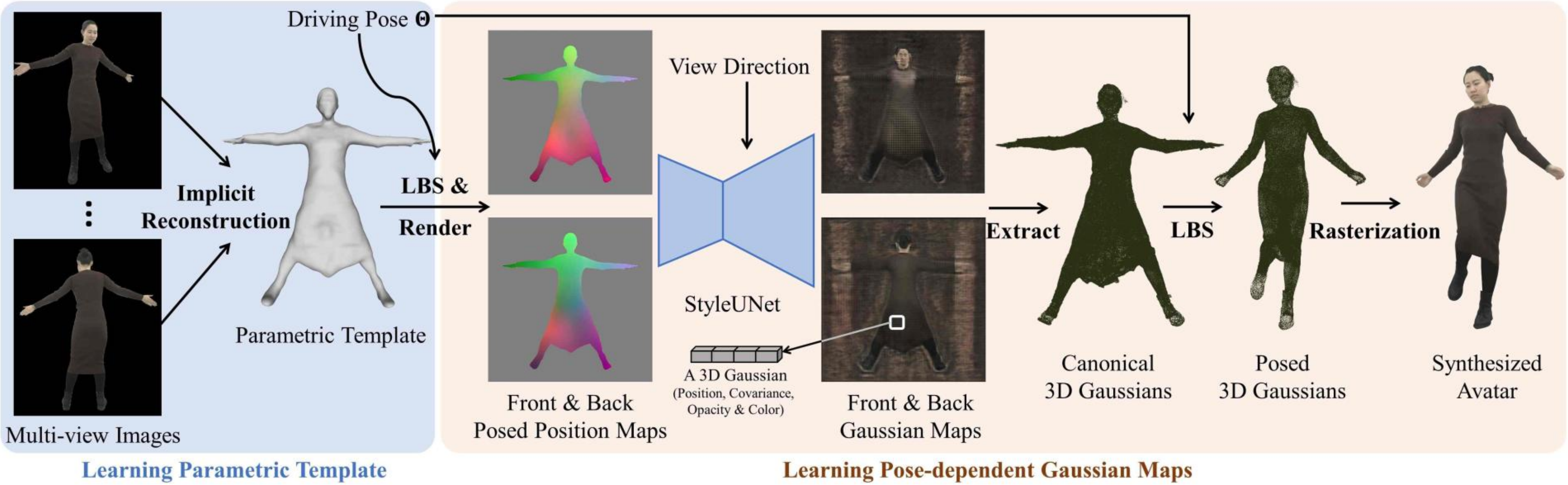



Animatable Gaussians



Training Data: Multi-view RGB Videos

Animatable Gaussians



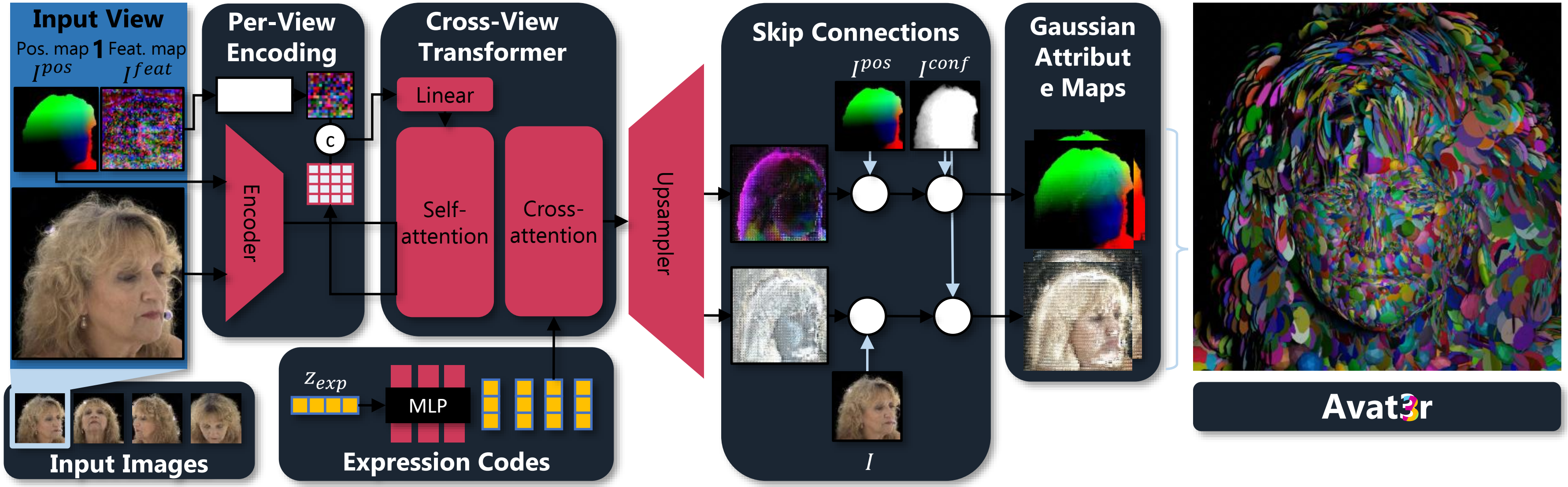
[Li et al. '24] Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling

Animatable Gaussians

Animated with novel motion sequence

[Li et al. '24] Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling

Generalizable 3D Head Avatars



arXiv'25 [Kirschstein et al.] Avat3r

Generalizable 3D Head Avatars

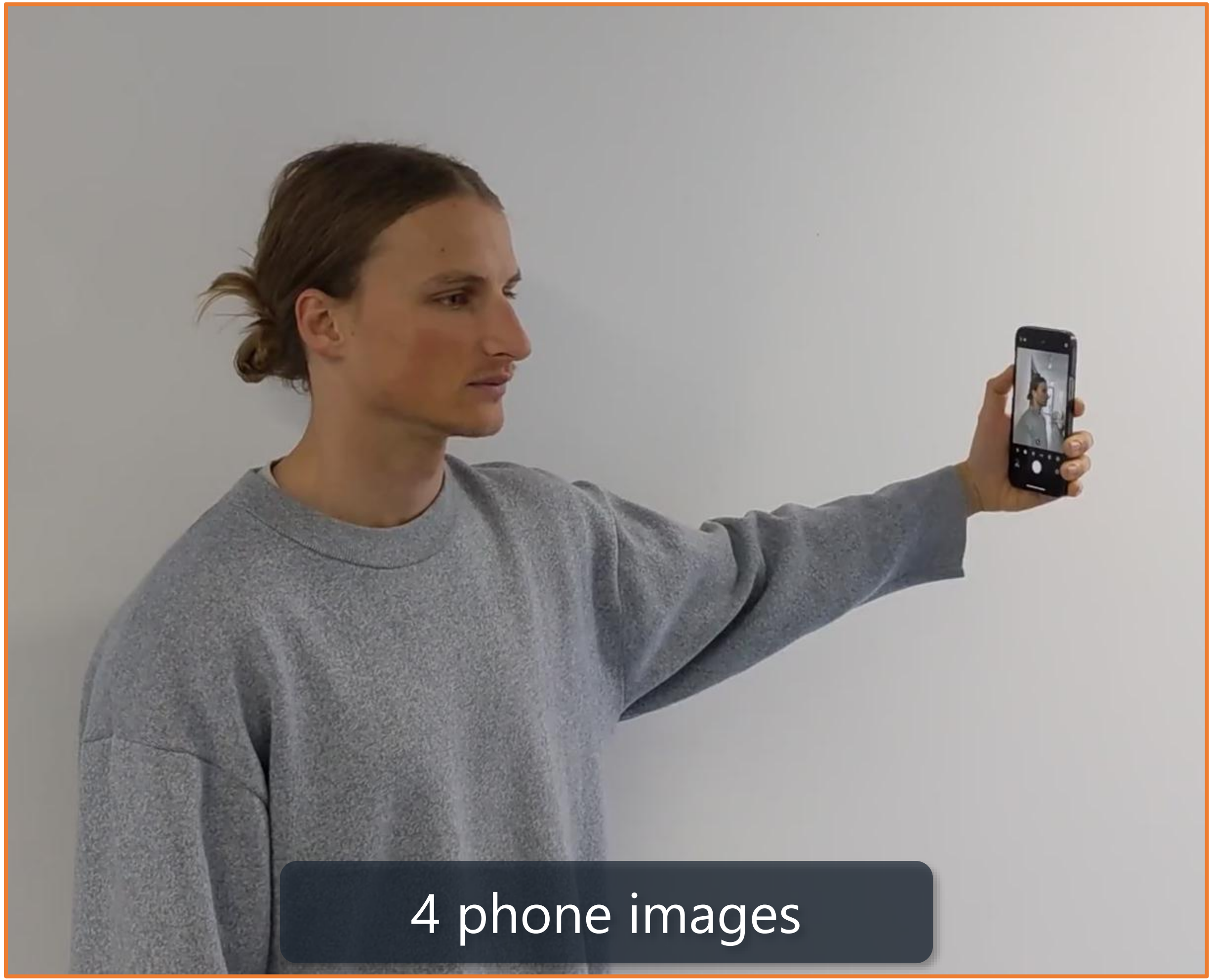
arXiv'25 [Kirschstein et al.] Avat3r

Avat3r

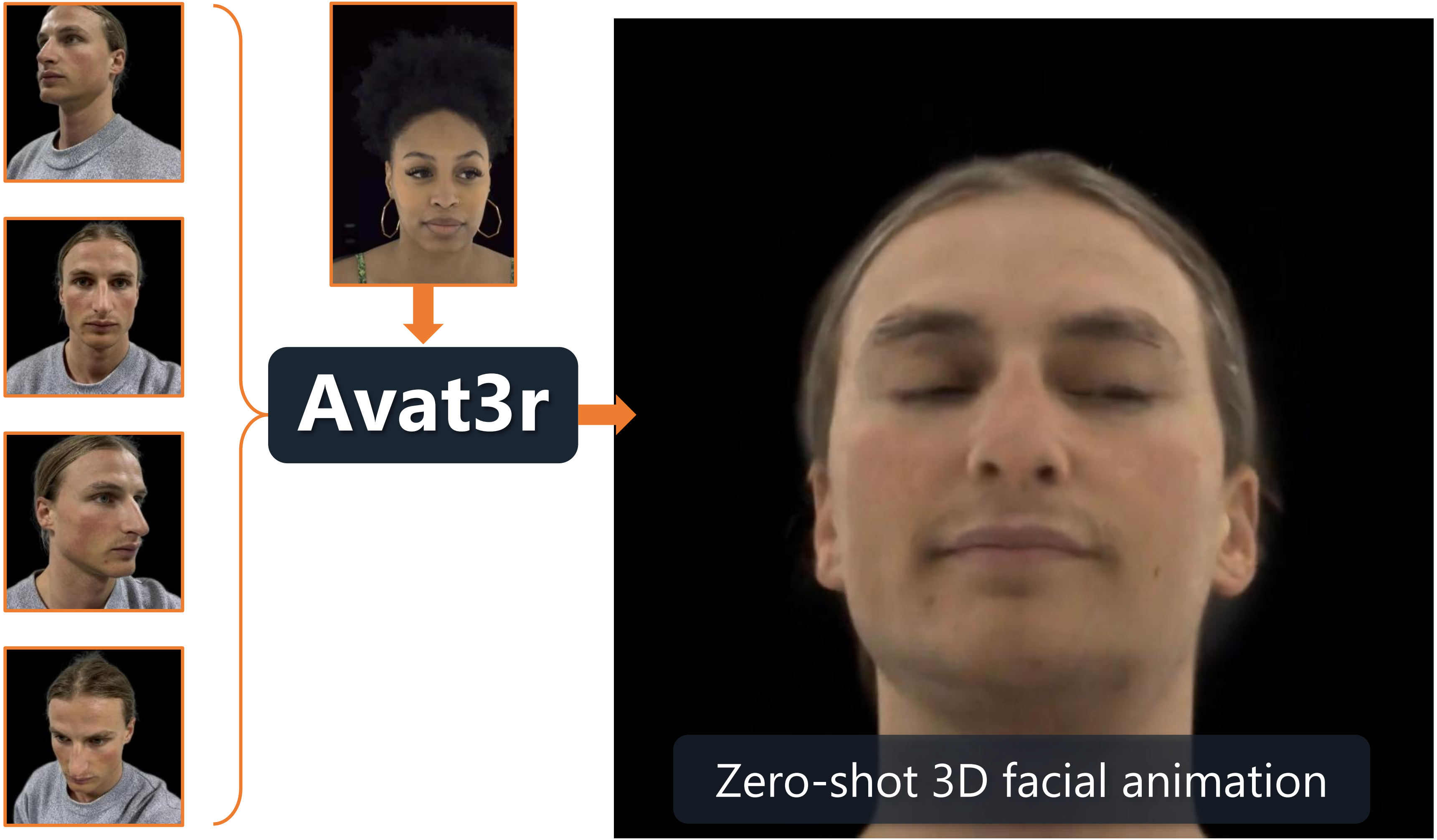
Large Animatable Reconstruction Model for High-fidelity 3D Head Avatars

Tobias Kirschstein[1,2] - Javier Romero[2] - Artem Sevastopolsky[1,2] - Matthias Nießner[1] - Shunsuke Saito[2]

[1]Technical University of Munich [2]Meta Reality Labs



arXiv'25 [Kirschstein et al.] Avat3r


Avat3r
Zero-shot 3D facial animation

Summary: NeRF, 3DGS, etc.

Textured Meshes vs NerRFs/3D Gaussians

[Mildenhall et al. 20]: NeRF

Textured Meshes vs NerRFs/3D Gaussians

Reconstruction:
- Optimization-based
- Need good gradients
- Not just surface rep!

Rendering:
- Need only surface
- Fast access / efficient
- No gradients needed

[Neural Radiance Fields]   [3D Gaussians]   [3D Meshes]

Reading Homework

- [Kerbl & Kopanas et al. 2023] 3D Gaussian Splatting for Real-Time Radiance Field Rendering
  – https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/3d_gaussian_splatting_low.pdf

Literature

- [Kerbl & Kopanas et al. 2023] 3D Gaussian Splatting for Real-Time Radiance Field Rendering
- [Zwicker et al. 2001] Surface Splatting
- [Yifan et al. 2019] Differentiable Surface Splatting for Point-based Geometry Processing
- [Luiten 2023] Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis
- [Qian et al. 2024] GaussianAvatars

Literature

- SIGGRAPH Asia'24 [Giebenhain et al.]: Neural Parametric Gaussian Avatars
- [Kirschstein et al. 2025] Avat3r
- [Li et al. 2024] Animatable Gaussians: Learning Pose-dependent Gaussian Maps for High-fidelity Human Avatar Modeling

Thanks for watching!

Some Slides adapted from…

- 3D Gaussian Splatting Tutorial 3DV 2024, Georgios Kopanas, Bernhard Kerbl, Antoine Guédon and Jonathon Luiten