3D Gaussian Splatting

3DV Tutorial

18 March, 2018

George Kopanas
Google

Bernhard Kerbl
TU Wien

Jonathon Luiten
Meta Reality Labs

Antoine Guédon
Ecole des Ponts ParisTech

Speakers

George Kopanas

Research Scientist

Google

Jonathon Luiten

Research Scientist

FACEBOOK REALITY LABS

Bernhard Kerbl

Principal Investigator

TU WIEN
TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

Antoine Guédon

PhD Candidate – IMAGINE/ENPC

École des Ponts
ParisTech

Schedule

14:30 – 15:00	Introduction to 3DGS	George Kopanas
15:00 – 15:15	Q&A	
15:15 – 16:00	3D Gaussians in Practice	Bernhard Kerbl
16:00 – 16:30	Coffee Break	
16:30 – 17:00	Research Overview: Dynamic 3D Gaussians	Jonathon Luiten
17:00 – 17:15	Q&A	
17:15 – 17:45	Research Overview: Surface Reconstruction	Antoine Guédon
17:45 – 18:00	Q&A	

www.3dgstutorial.github.io

Motivation

Reconstructing the 3D world
from images + videos.

Motivation

Reconstructing the 3D world from images + videos.

Input Images

Motivation

Reconstructing the 3D world from images + videos.

Screen Capture from “RealityCapture”

Motivation

Reconstructing the 3D world from images + videos.

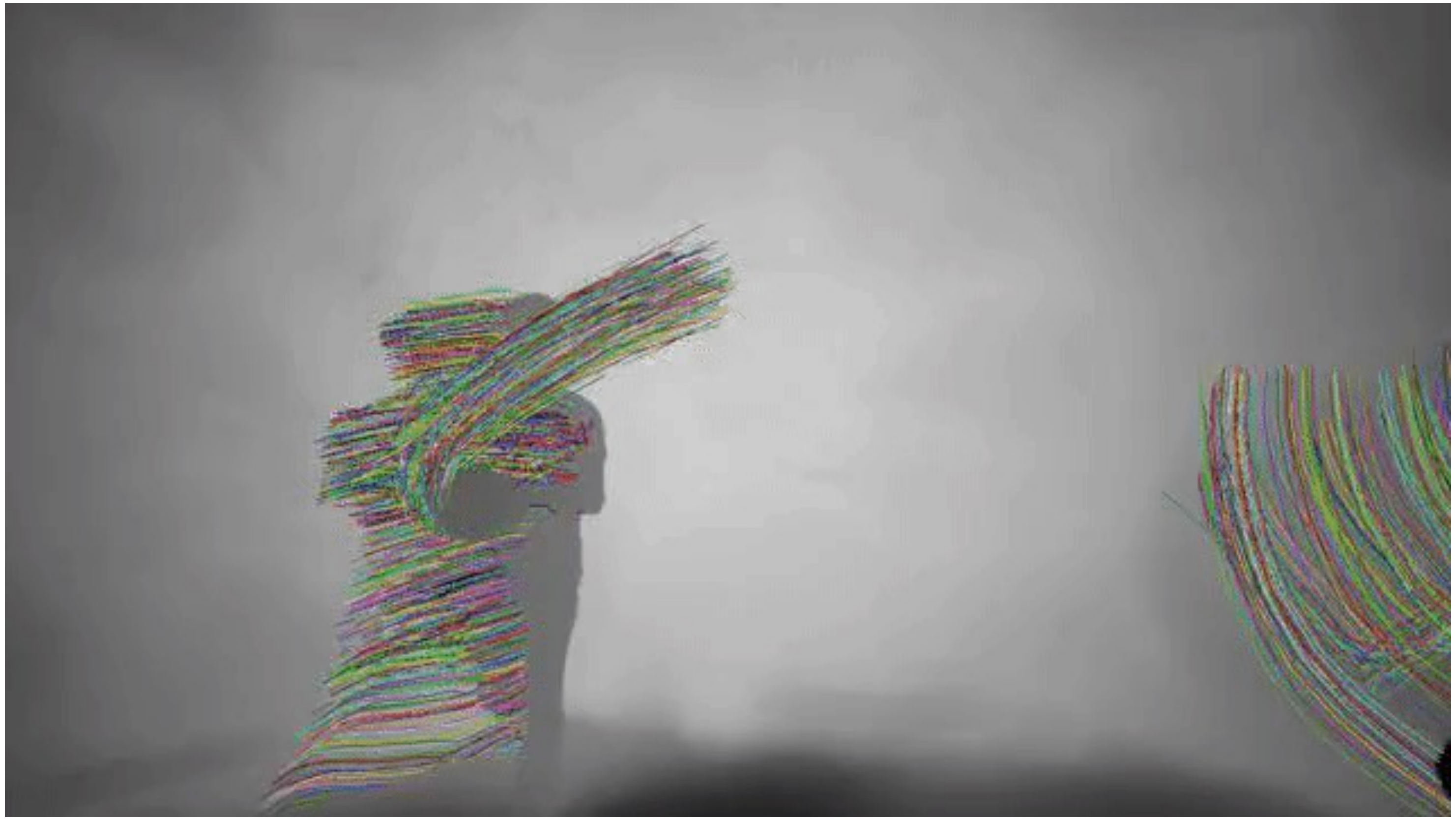
1. Exploration
   (Re-render from Novel Views)

"3D Gaussian Splatting for Real-Time Radiance Field Rendering"

Motivation

Reconstructing the 3D world
from images + videos.

1. Exploration
   (Re-render from Novel Views)
2. Understanding
   (3D Tracking, 3D Video editing etc)



"Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis."

Motivation

Ideal 3D Representations:

1. **Accurate**
2. **Fast**
3. **Memory Efficient**
4. **Practical:** easy to integrate in frameworks

Motivation

Ideal 3D Representations:

1. **Accurate**
2. **Fast**
3. **Memory Efficient**
4. **Practical:** easy to integrate in frameworks

Gaussian Splatting:

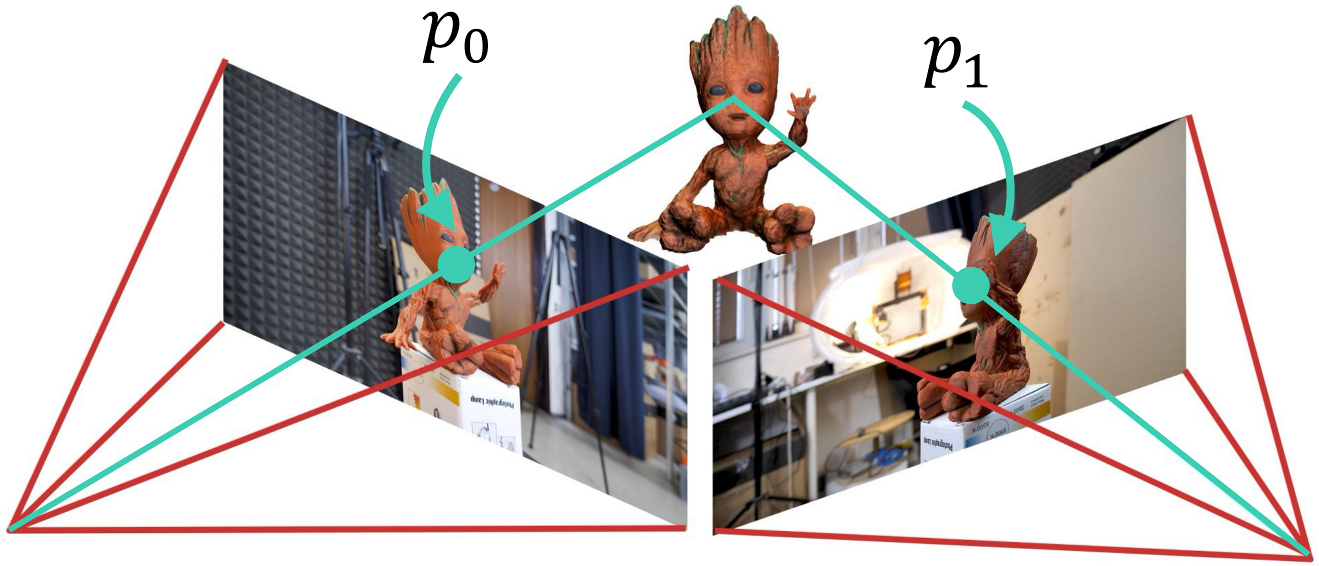
1. Comparable **PSNR** with MipNeRF360.
2. 100+ Frames per Second and trains in less than 1h.
3. Renders on **Mobile Devices** (< 6GB VRAM).
4. Many implementations on different Graphics frameworks.
    a. Format: **easy to standardize** (.ply files).

Related Work

Related Work

Mesh-Based Representations

- Estimate Geometry with Multi-View Stereo

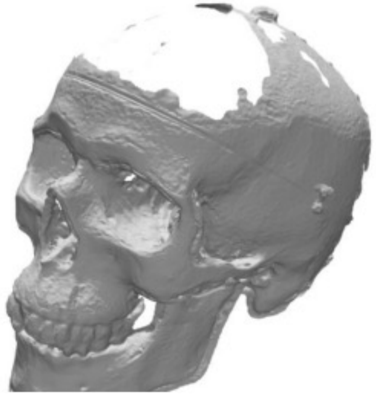


https://blog.prusa3d.com/wp-content/uploads/2018/03/epipolar_geometry.jpg

Related Work

Mesh-Based Representations

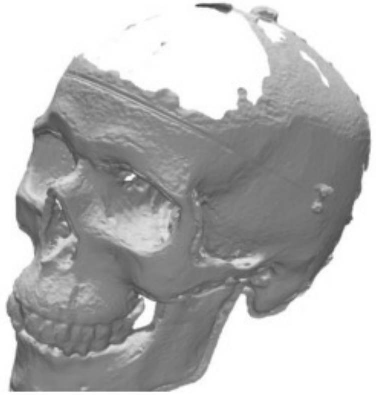
- Estimate Geometry with Multi-View Stereo
- Fixing errors in a triangle mesh – Very Challenging



Colmap Reconstruction - MVS

Related Work
Mesh-Based Representations

- Estimate Geometry with Multi-View Stereo
- Fixing errors in a triangle mesh – Very Challenging
- Fixing them by learning to ignore them:
    - Deep Blending [Hedman 2018]
    - Stable View Synthesis [Riegler 2020]



Colmap Reconstruction - MVS

Related Work
Mesh-Based Representations

Deep Blending [Hedman 2018] - Museum

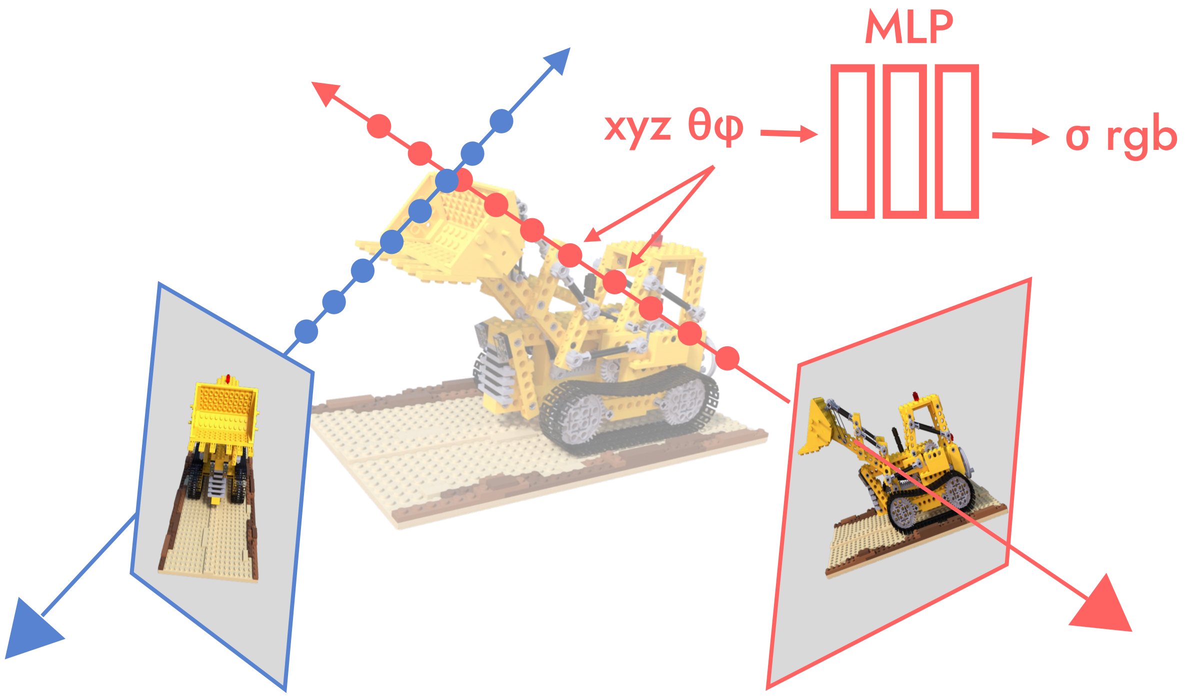
Related Work

Mesh-Based Representations

Deep Blending [Hedman 2018] – Concave Bowl

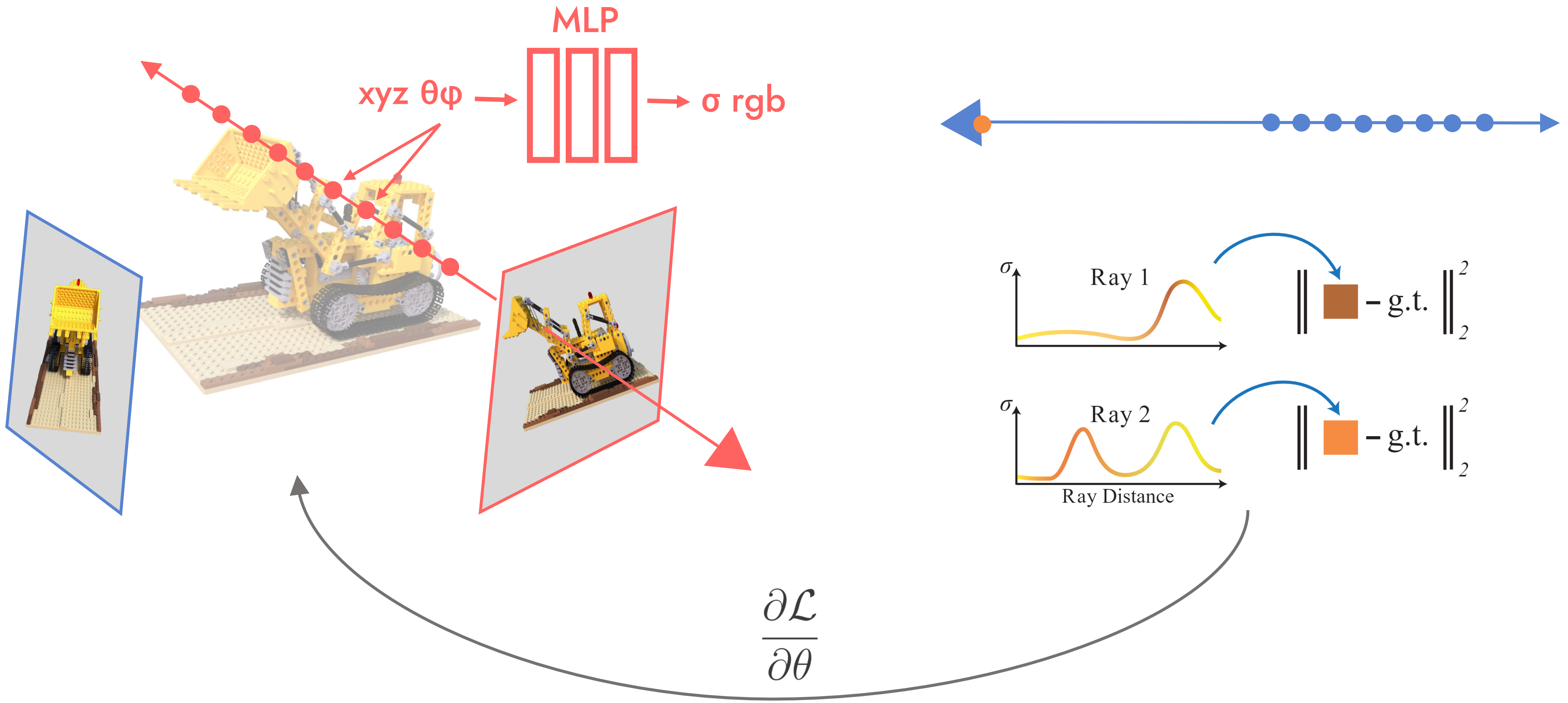
Related Work

Neural Radiance Fields



Related Work

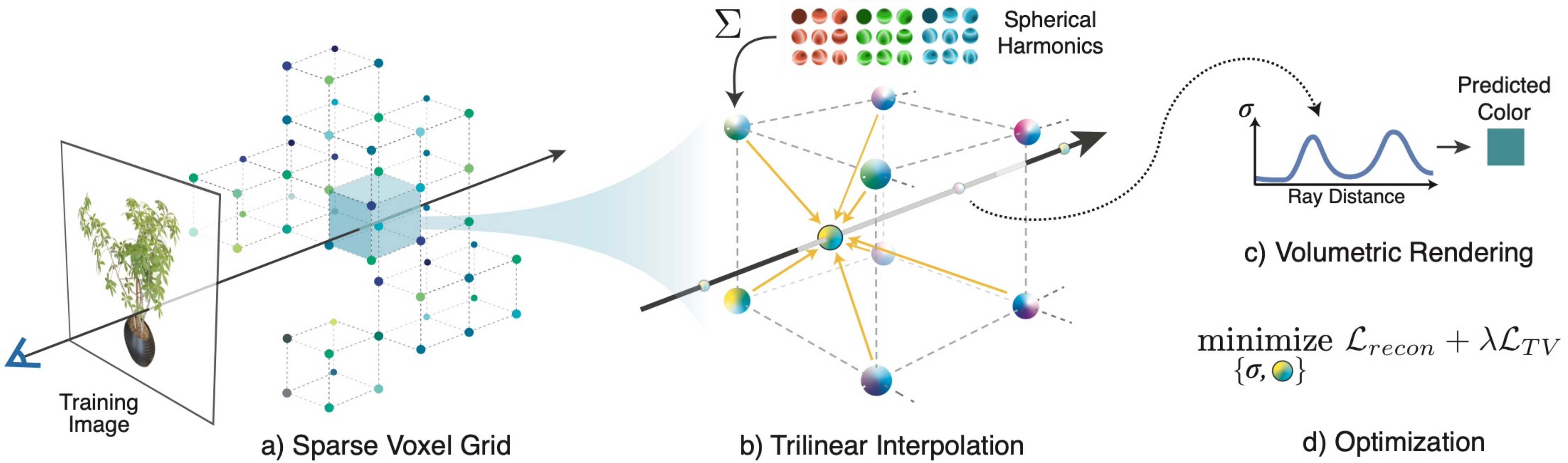
Neural Radiance Fields



Related Work
Neural Radiance Fields

NeRF suffers from slow training and rendering

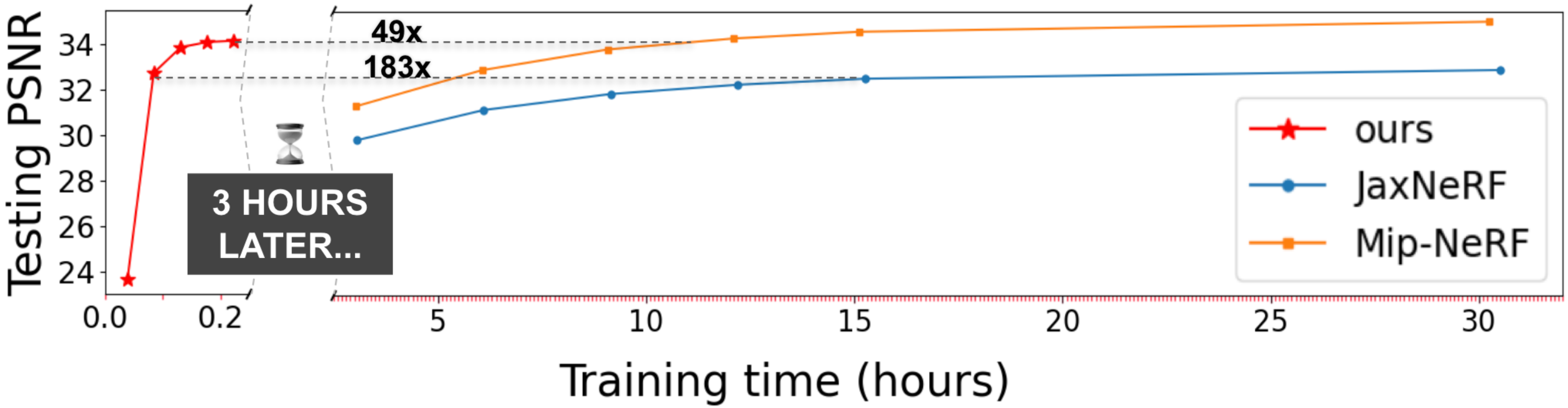
- DVGO [Sun 2022]
- Instant-NGP [Müller 2022]
- Plenoxels [Fridovich-Keil and Yu 2022]
- TensoRF [Chen and Xu 2022]



Plenoxels [Fridovich-Keil and Yu 2022]

Related Work
Neural Radiance Fields

NeRF suffers from slow training and rendering
- DVGO [Sun 2022]
- Instant-NGP [Müller 2022]
- Plenoxels [Fridovich-Keil and Yu 2022]
- TensoRF [Chen and Xu 2022]



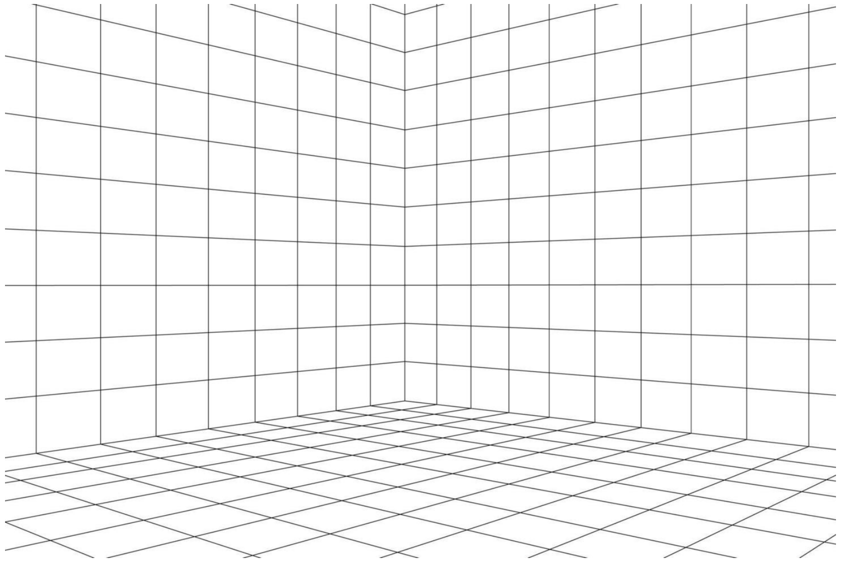
DVGO [Sun 2022]

Related Work

Point-Based Representations

Eulerian (NeRFs)

- Queries in 3D Space



Lagrangian

- Access to primitives

Background

Traditional Point-Based Graphics

Background

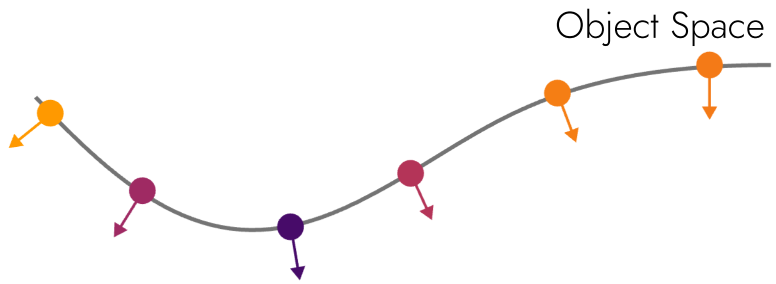
Traditional Point-Based Graphics

Surface Splatting - Zwicker et al. 2001

(EWA – Elliptical Weighted Average)

Background

Traditional Point-Based Graphics



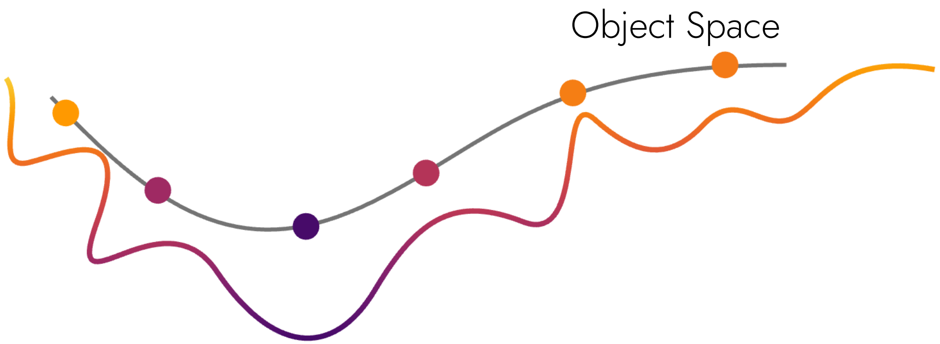
Surface Splatting - Zwicker et al. 2001

(EWA – Elliptical Weighted Average)

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

Background

Traditional Point-Based Graphics



Surface Splatting - Zwicker et al. 2001

(EWA – Elliptical Weighted Average)

1. Considers **oriented** points (surfels) as discrete samples of a texture function on a surface.
2. A Gaussian reconstruction kernel is used to recover a continuous signal.

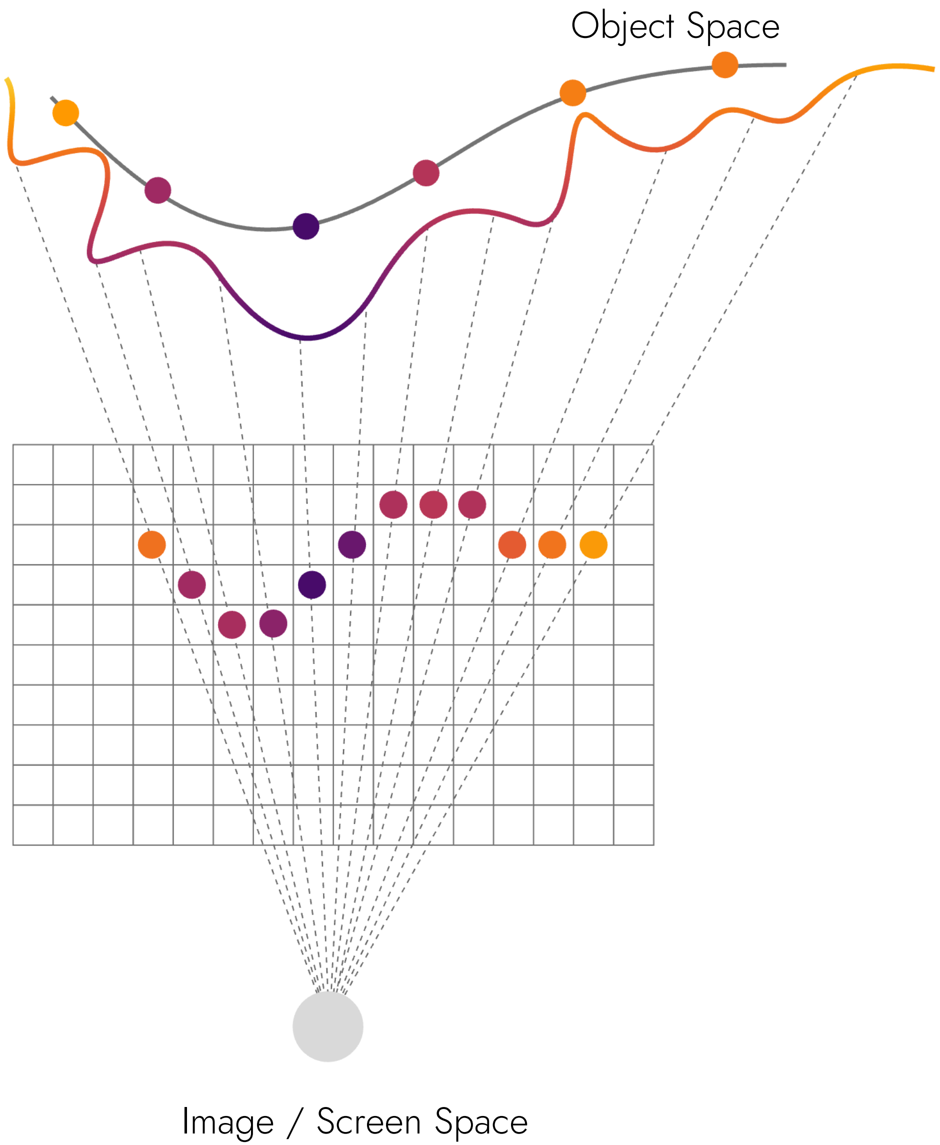
Background

Traditional Point-Based Graphics

Surface Splatting - Zwicker et al. 2001

(EWA – Elliptical Weighted Average)

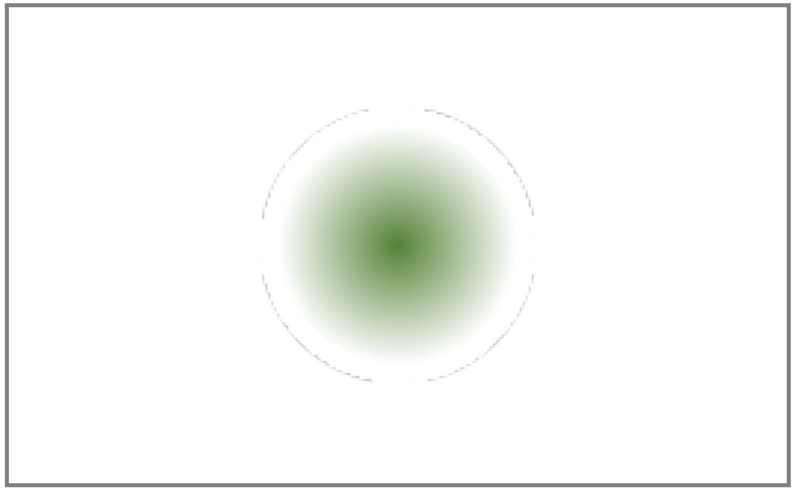
1. Considers **oriented** points (surfels) as discrete samples of a texture function on a surface.
2. A Gaussian reconstruction kernel is used to recover a continuous signal.
3. Such that we can sample it in screen space.



Background

Traditional Point-Based Graphics

The important outcomes of this algorithm are:

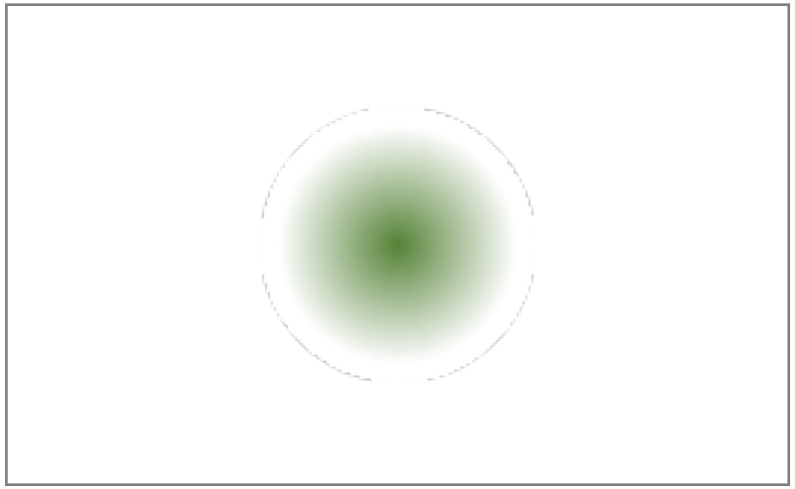


Image

Background
Traditional Point-Based Graphics

The important outcomes of this algorithm are:

1. Moving camera closer, scales the points so the objects have no holes.

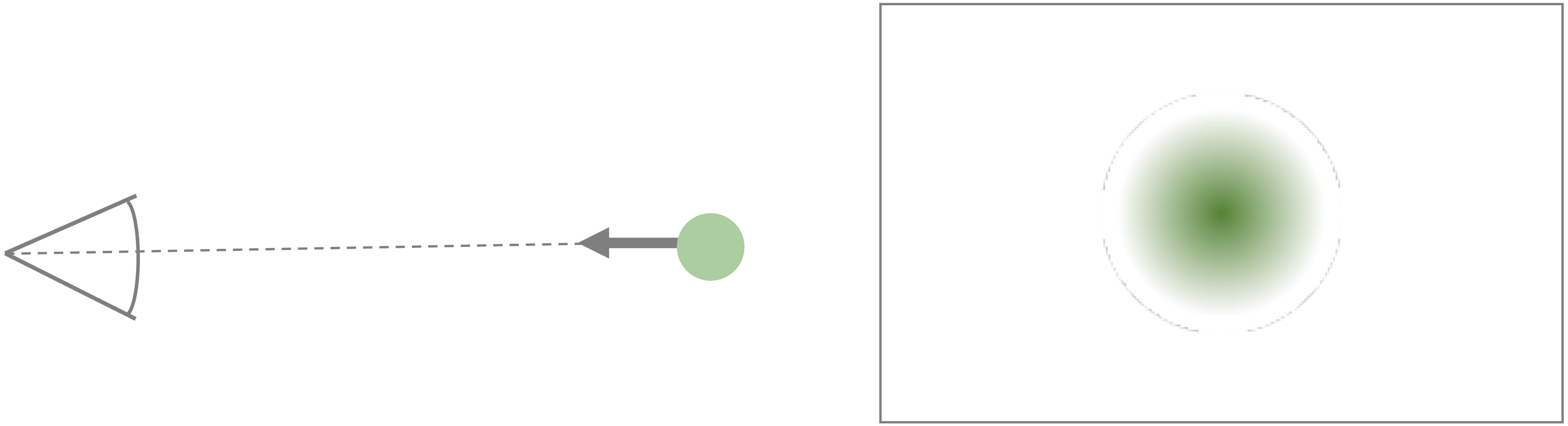


Image

Background
Traditional Point-Based Graphics

The important outcomes of this algorithm are:

1. Moving camera closer, scales the points so the objects have no holes.
2. Slanted normals appear as ellipses, so we can create better edges.



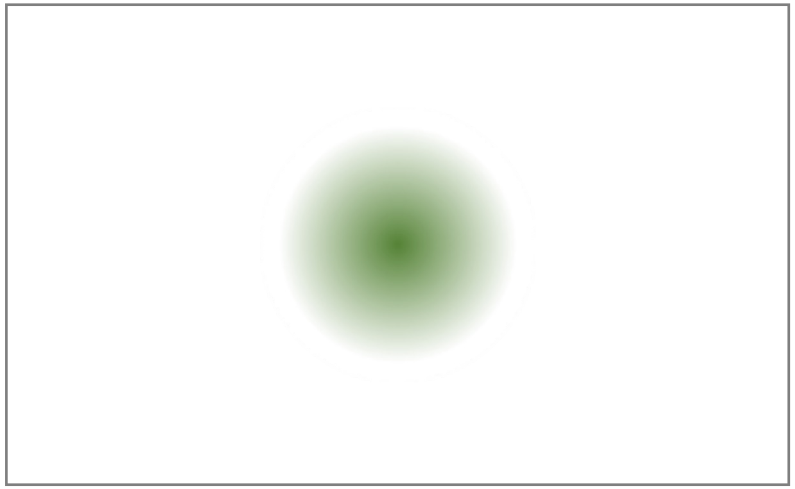
Image

Background

Traditional Point-Based Graphics

The important outcomes of this algorithm are:

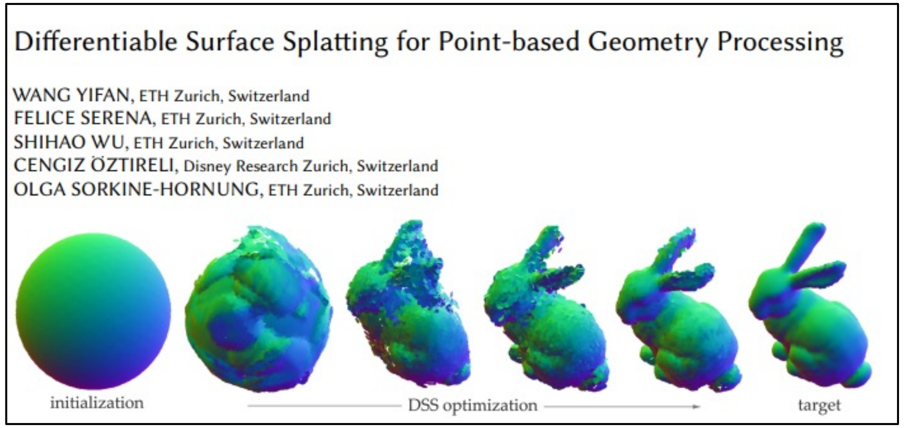
1. Moving camera closer, scales the points so the objects have no holes.
2. Slanted normals appear as ellipses, so we can create better edges.
3. Each sample can be processed independently



Image

Background

Recent Advances in Point Clouds

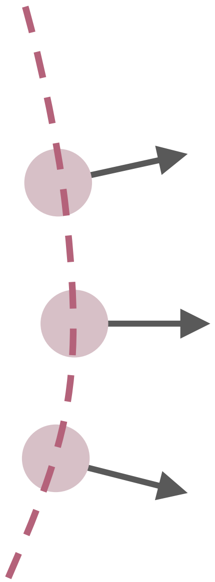

Differentiable Surface Splatting for Point-based Geometry Processing

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

- Differentiable Surface Splatting [Yifan '19] showed that this process is end-to-end differentiable.
- 3DGS is heavily inspired and builds on top of this line of work.

Surface Splatting vs Volume Splatting

Surface Splatting vs Volume Splatting



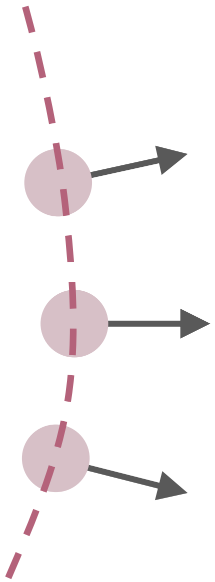
position normal

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

σ std dev

f appearance

Surface Splatting vs Volume Splatting

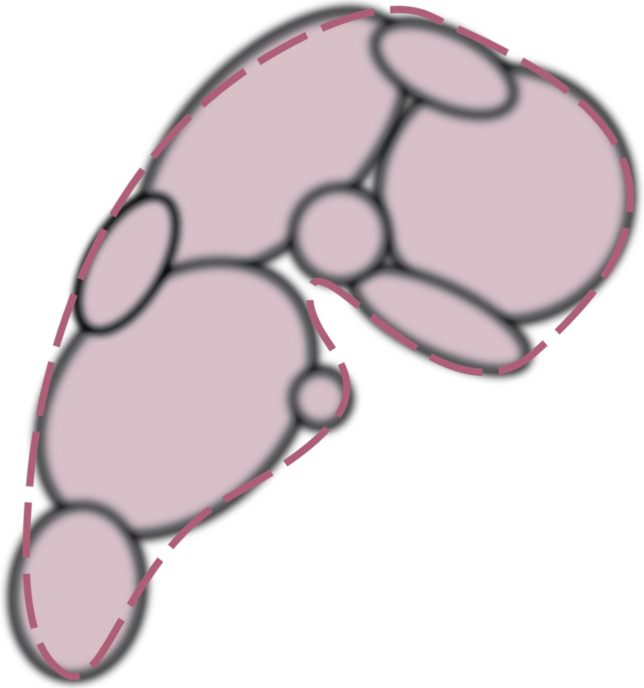


position normal

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

σ std dev

f appearance



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

covariance matrix

o SH spherical harmonics

Surface Splatting vs Volume Splatting

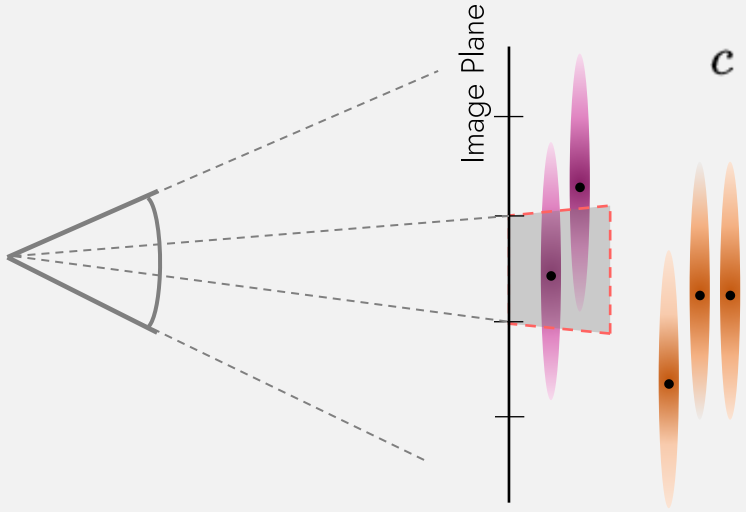
1. How do we blend points in screen space?

Surface Splatting vs Volume Splatting

1. How do we 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

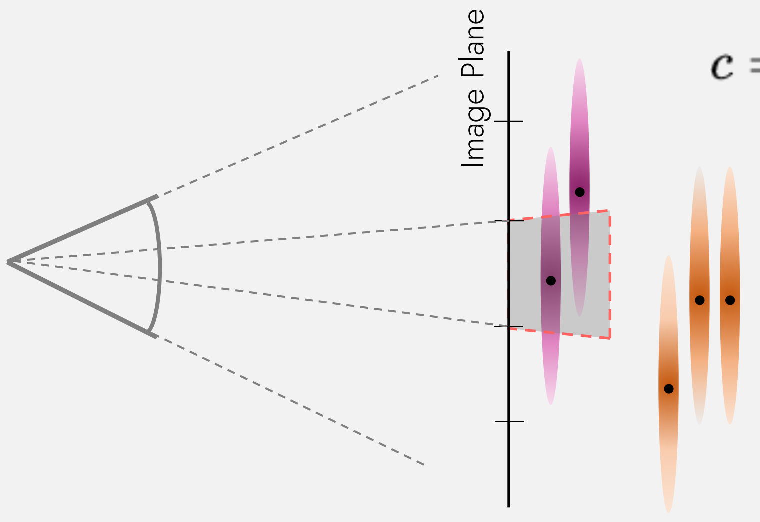
[Zwicker1 '01] Surface Splatting
[Yifan '19] Differentiable Surface Splatting for Point-Based Geometry Proccessing

Surface Splatting vs Volume Splatting

1. How do we blend points in screen space?

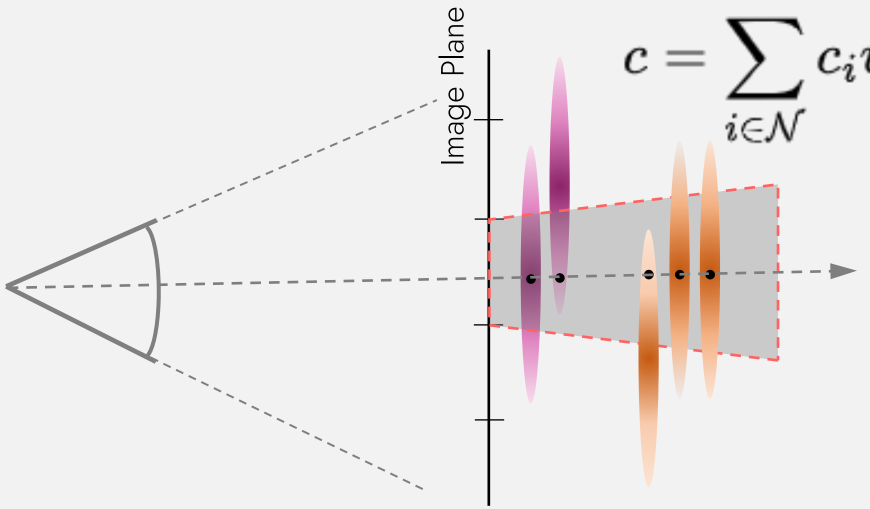
[Zwicker1 '01] / [Yifan '19]

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



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

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



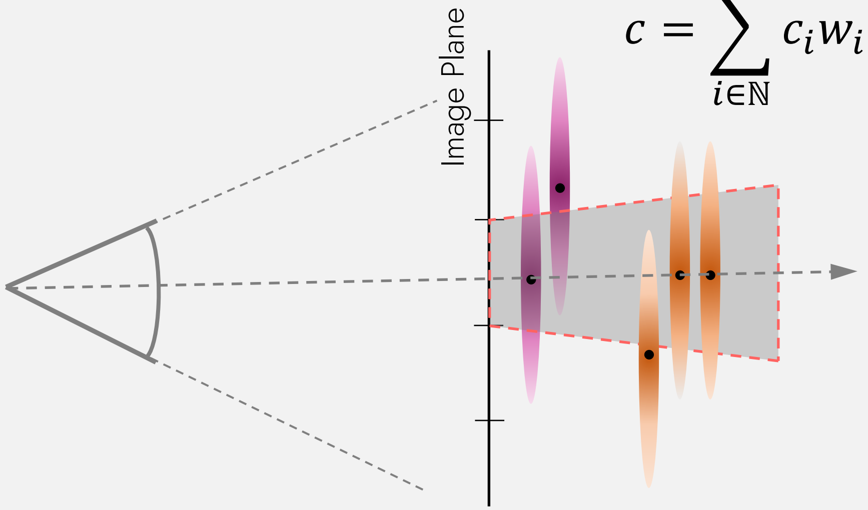
[Zwicker1 '01] Surface Splatting
[Yifan '19] Differentiable Surface Splatting for Point-Based Geometry Proccessing
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

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]

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



Top-Down View

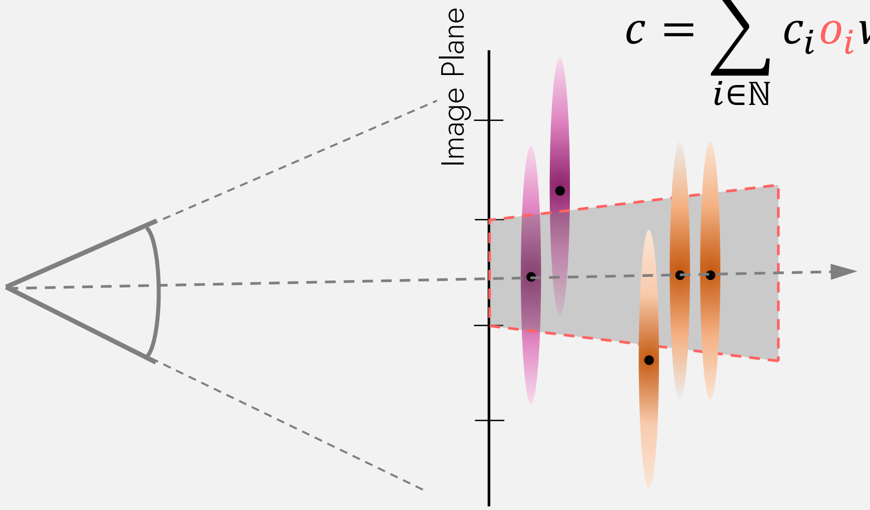
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

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.

[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

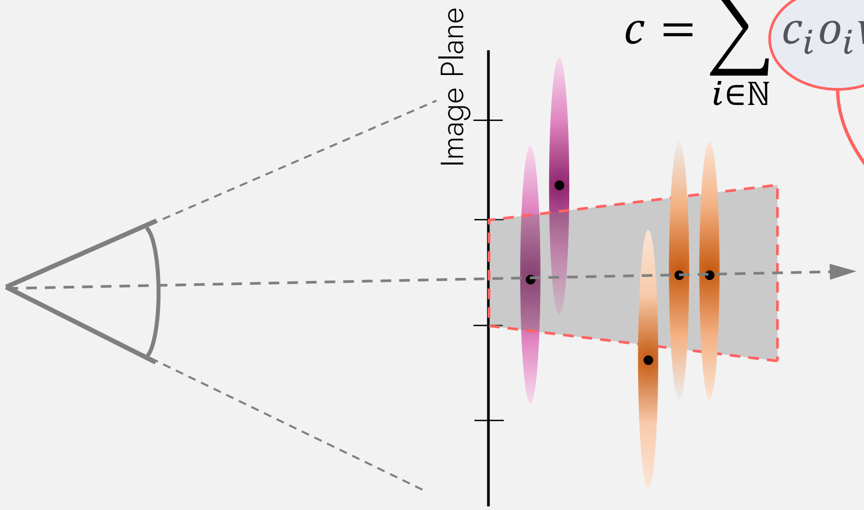
[Zwicker2 '01] EWA Volume Splatting
[Kerbl & Kopanas '23] 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Surface Splatting vs Volume Splatting

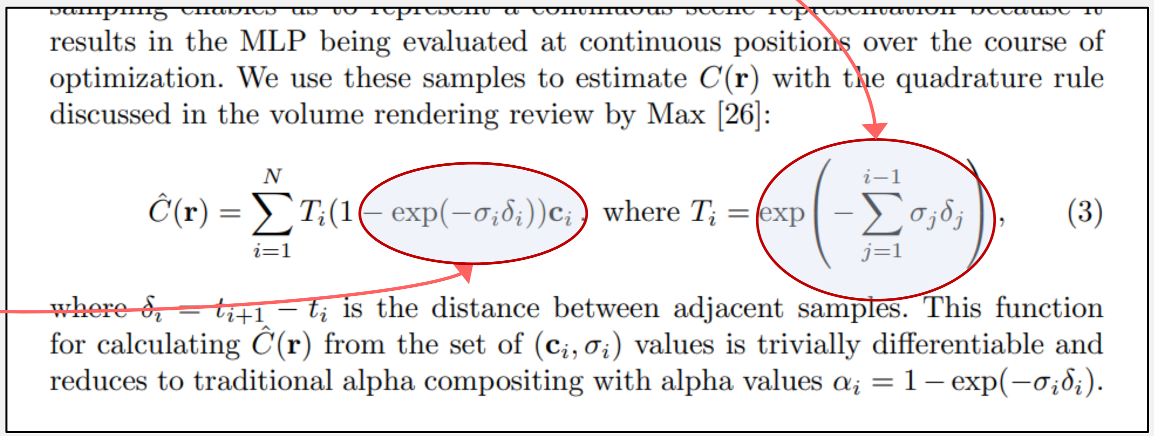
1. How do we blend points in screen space?
2. Opacity for each point, allows us to make points disappear.

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]

Visualization of the 3D ellipsoids

Visualization of the 3D ellipsoids

What are the benefits of 3D Gaussians?

Initialization

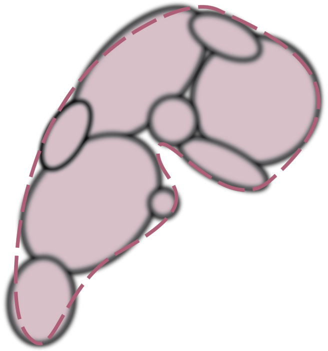
- No Multi-View-Stereo → SfM
- SfM points → No Normals
- Start with isotropic Gaussians
- Can even start from **random** initialization

Quality

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

How do we render?

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



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

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

Optimization

Optimization

- How do you optimize a covariance matrix?

Optimization

- How do you optimize a covariance matrix?
  - Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

Optimization

- How do you optimize a covariance matrix?
  - Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

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

  - For any rotation and scale this is a valid covariance matrix

Optimization

- How do you optimize a covariance matrix?
  - Not all symmetric matrices are covariance matrices. Gradient updates can easily make them invalid.

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

  - For any rotation and scale this is a valid covariance matrix
  - And because R does not optimize well, we use Quaternions.

How did we go from 5 FPS to 100+ FPS?

and from 18h to 40min for training

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.

Optimization

Now we have all the building blocks to run SGD.

What will happen?

Optimization

Ablation Run – No densification/adaptive control

Optimization

Ablation Run – No densification/adaptive control

Optimization

Full Run

Optimization

Full Run

Densification

Increase the number of points where necessary:

Densification

Increase the number of points where necessary:

- Points with **high positional gradients** correspond to regions that are **not well reconstructed** yet.

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.

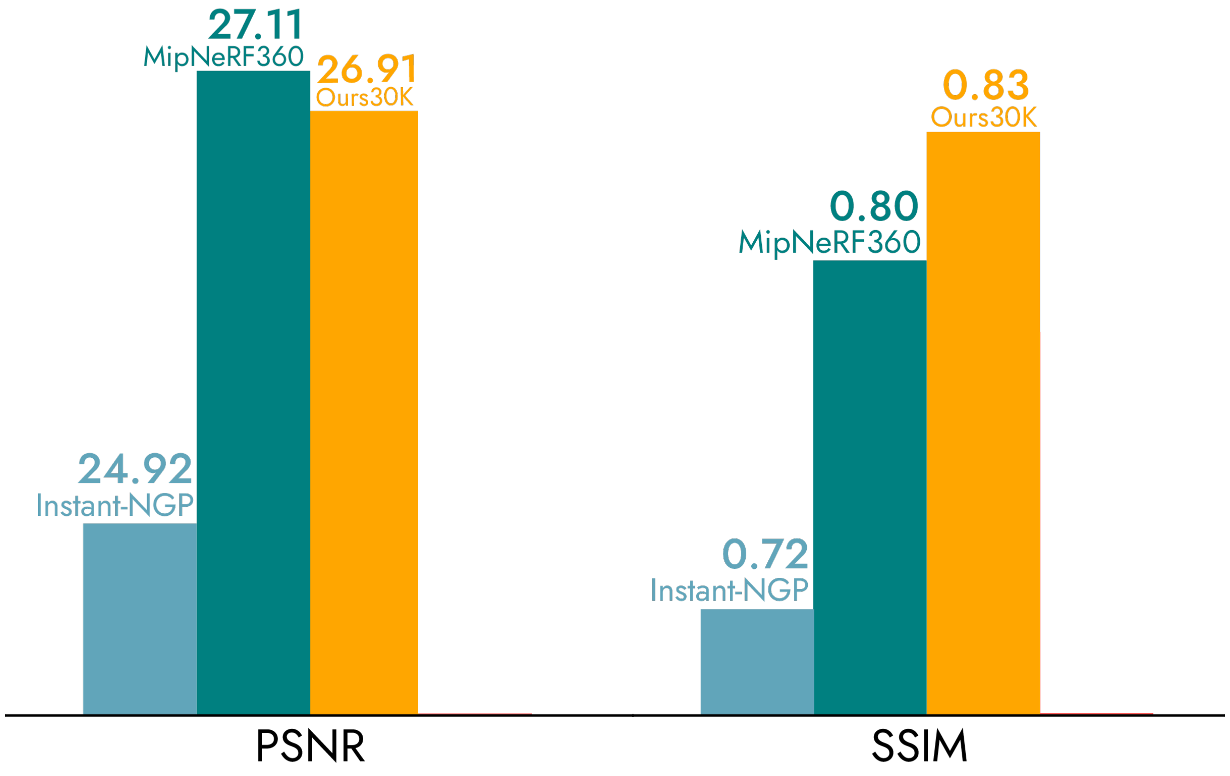
Interactive Results

https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
or
Google search: "3D Gaussian Splatting"


Menu
Views
Capture
3D Gaussians
Camera Point view
Die Aufnahme wurde begonnen
Metrics
78.72 (12.70 ms)
60

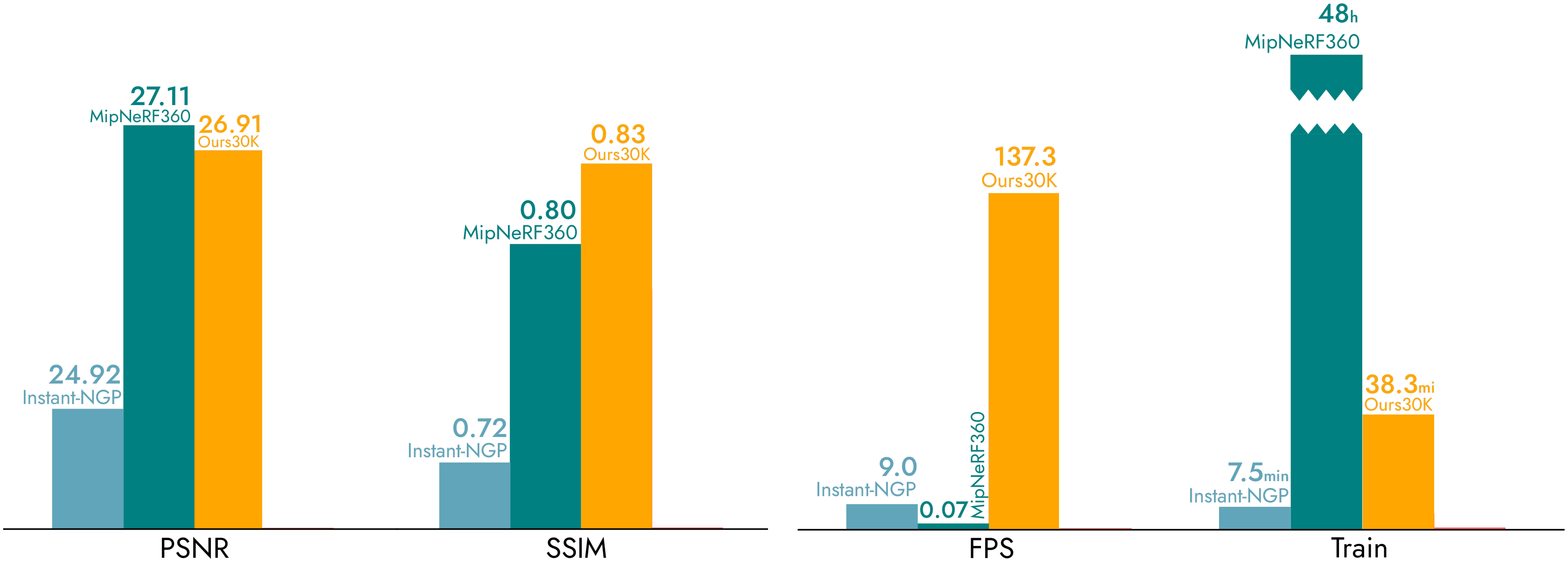
Evaluation

Full MipNeRF360 Dataset + 2 Tanks and Temples + 2 Deep Blending



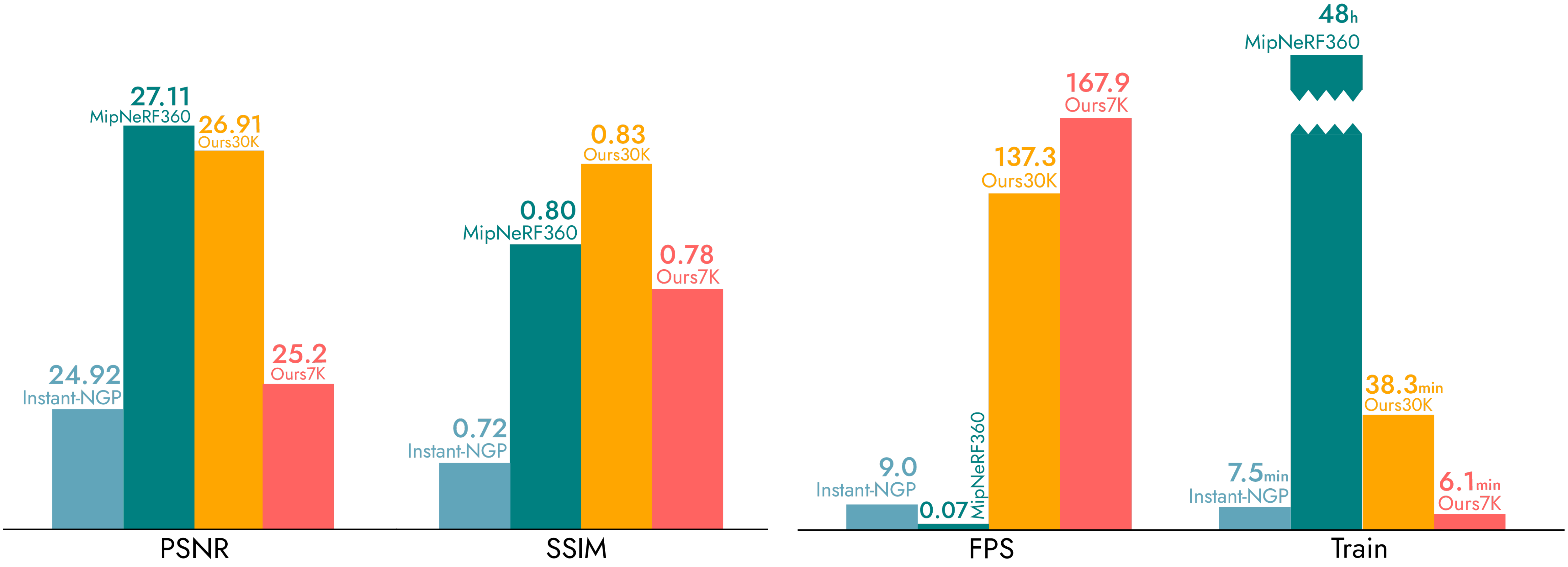
Evaluation

Full MipNeRF360 Dataset + 2 Tanks and Temples + 2 Deep Blending



Evaluation

Full MipNeRF360 Dataset + 2 Tanks and Temples + 2 Deep Blending



We evaluate our algorithm with full training and an early 5min stop.

Comparisons


Ours
MipNeRF360


Flipping between ours and INGP

Ablation Study - Anisotropy

Ground Truth

Full

Isotropic

Applications

Long Term:

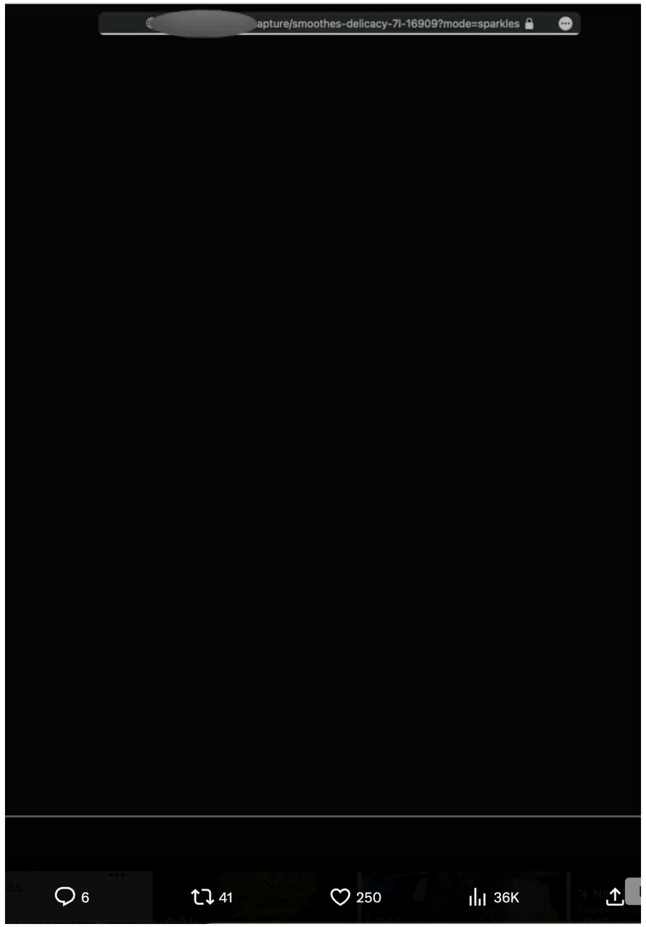
1. Robust, efficient and dynamic 3D reconstruction

Short Term:

1. VfX
2. Retail – E-commerce
3. 3D Grounded Video Editing

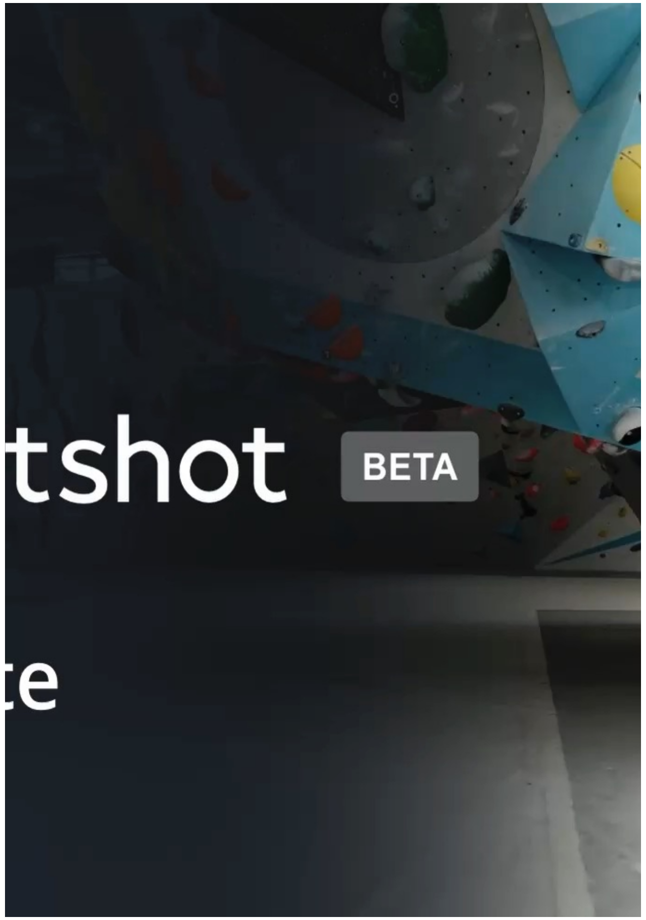
3DGS End-to-End Applications

Luma AI



@LumaLabsAI

PostShot

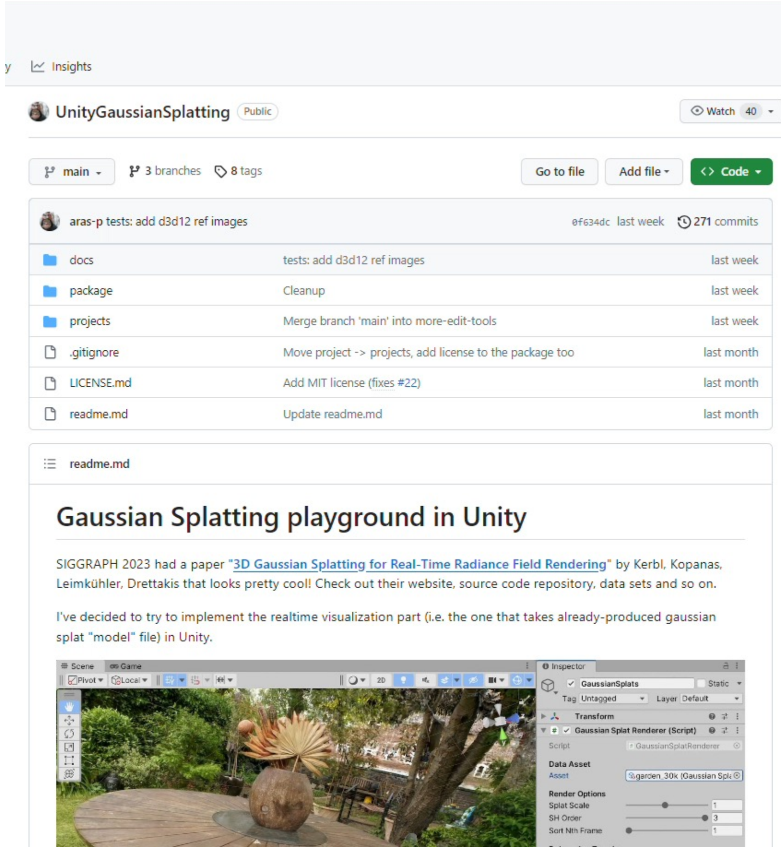


https://radiancefields.com/postshot-releases-v0-2/

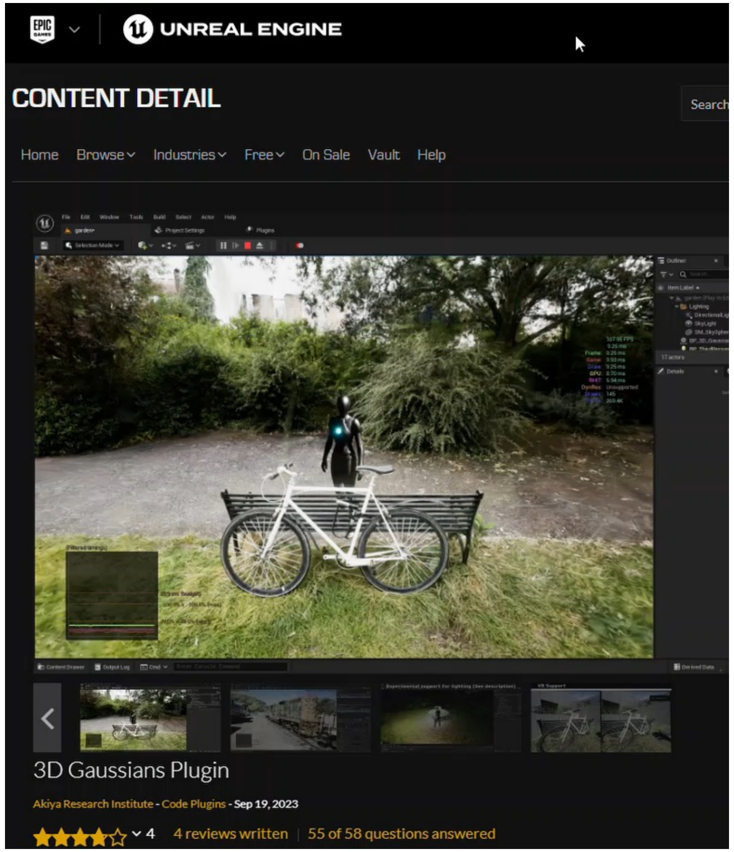
PolyCam

@PolyCam3D

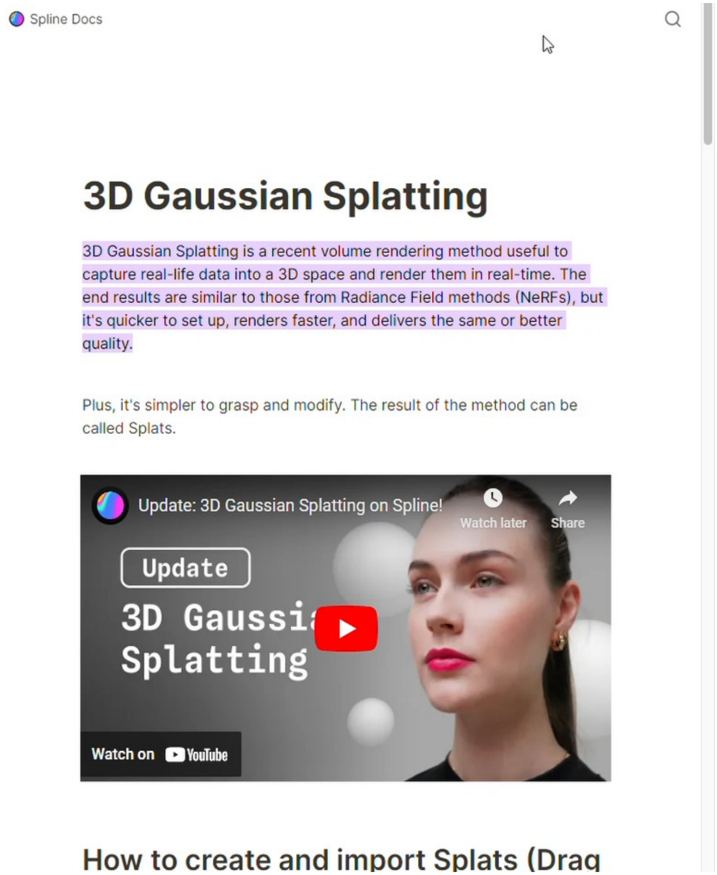
Gaussian Splatting in Graphics Engines



Gaussian Splatting in Unity

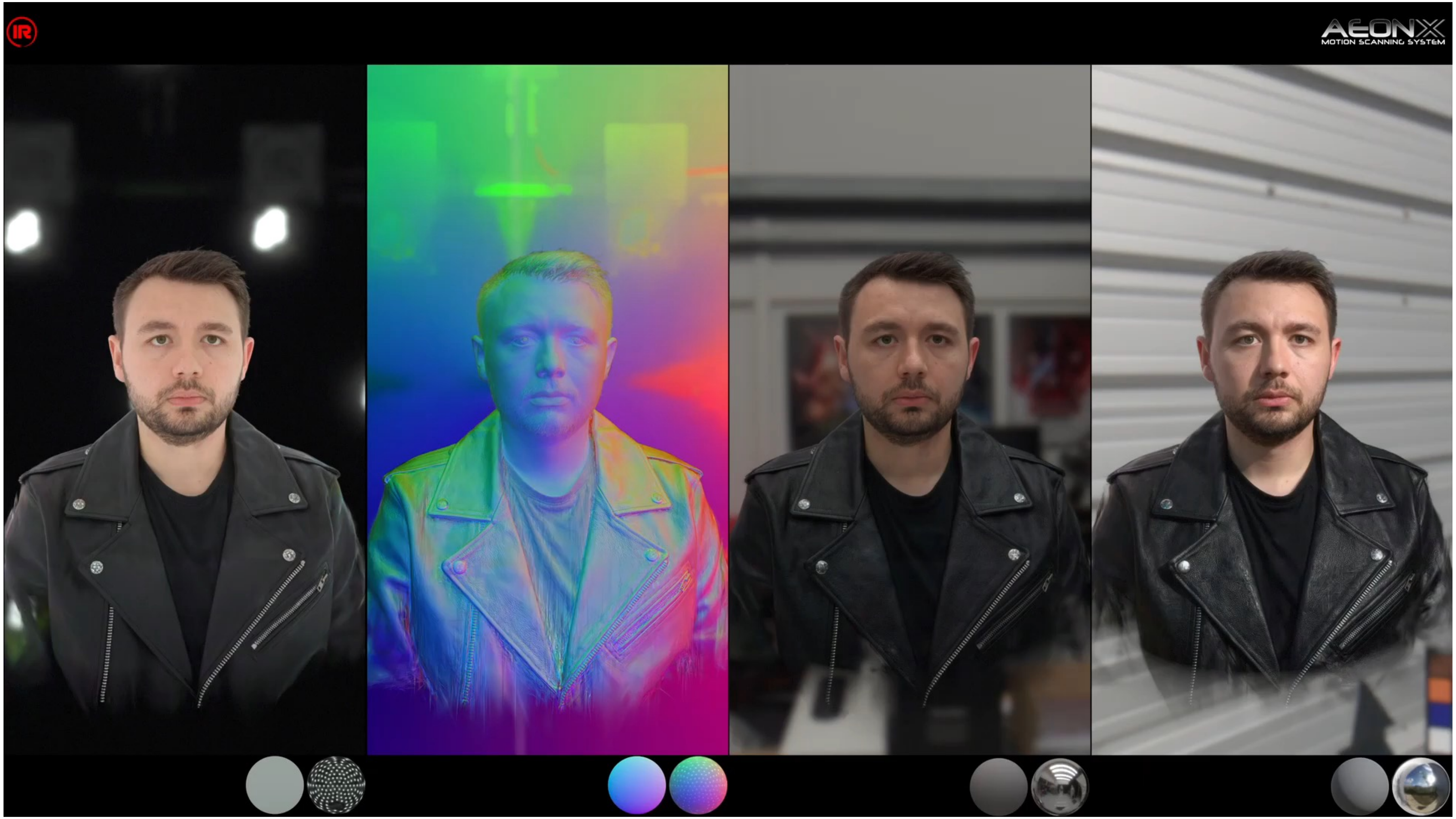


Gaussian Splatting in Unreal



Gaussian Splatting in Spline

Gaussian Splatting OLAT captures



Capture and video from "Infinite Realities"

Gaussian Splatting
Limitations and Progress

Limitations

1. Handcrafted heuristics for densification.
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

Wrap-Up

- Gaussian Splatting is fast, efficient, accurate and practical.
- But it doesn't mean that it comes without limitations.

How this efficiency will boot-strap **new ideas**, **applications** and **solutions** to **fundamental problems** of Radiance Fields?

Wrap-Up

- Gaussian Splatting is fast, efficient, accurate and practical.
- But it doesn't mean that it comes without limitations.

How this efficiency will boot-strap **new ideas**, **applications** and **solutions** to **fundamental problems** of Radiance Fields?

Thank you!