

3DGS and New Trends in Rendering

-- the past, present, and a glimpse in to possible future

3D Gaussian Splatting for Real-Time Radiance Field Rendering

SIGGRAPH 2023

(ACM Transactions on Graphics)

Bernhard Kerbl^{* 1,2}

Georgios Kopanas^{* 1,2}

Thomas Leimkühler³

George Drettakis^{1,2}

^{*} Denotes equal contribution

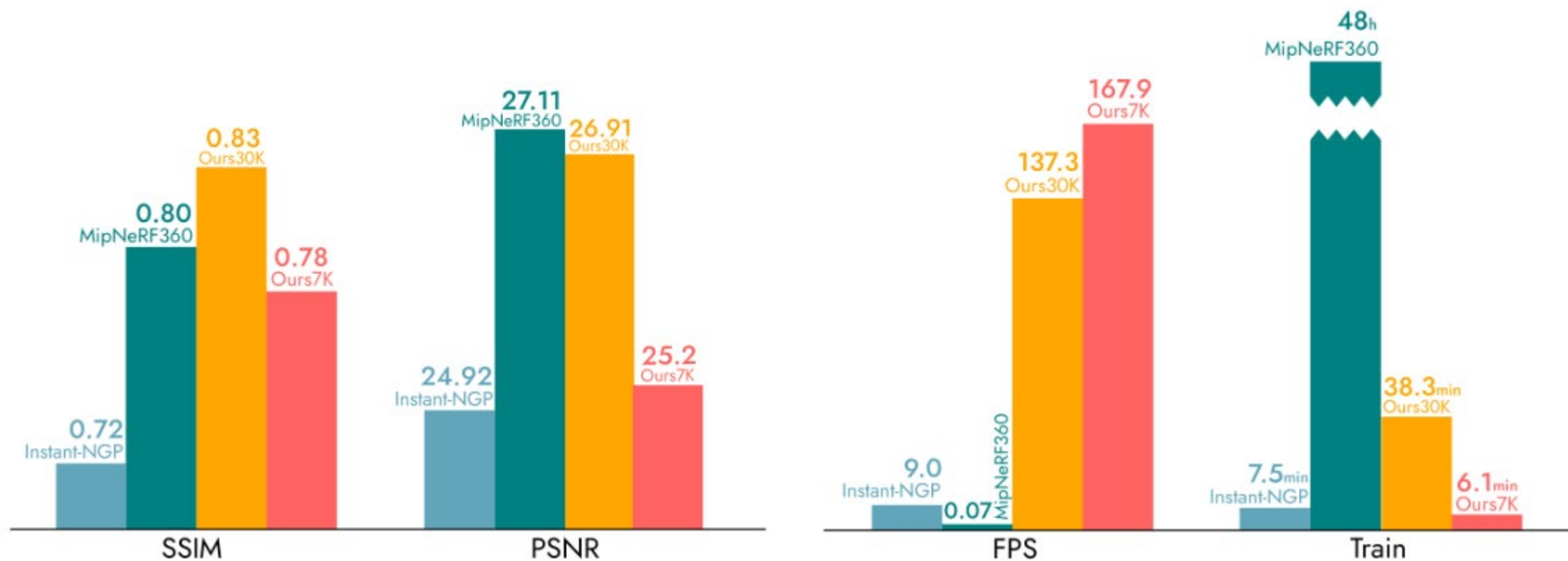
¹Inria

²Université Côte d'Azur

³MPI Informatik



3DGS is *fast* and the rendered images are *photorealistic*



Topics

- **The historic context of 3DGS**
- Present: [3DGS](#) and its recent progress
- Future discussions

The Historic Context of 3DGS

- **3D Representations: Mesh, Point Cloud, Implicit Surface/Neural Field, etc.**
- Volumetric Rendering
- Point Based Graphics

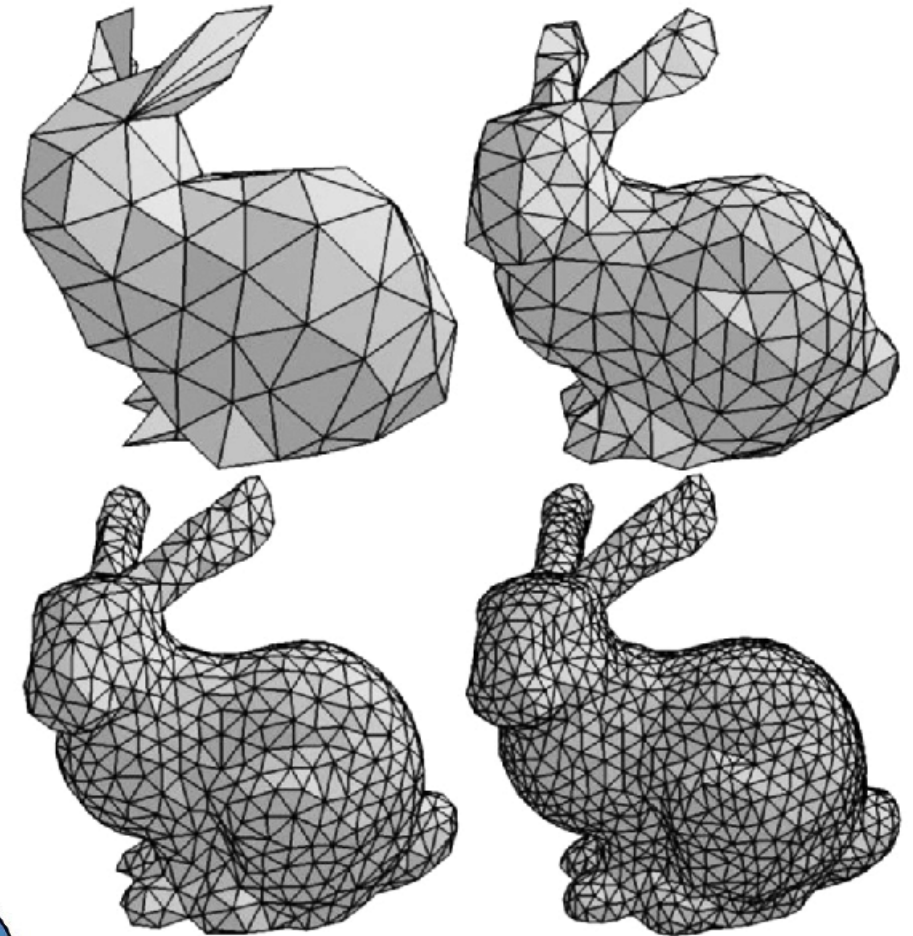
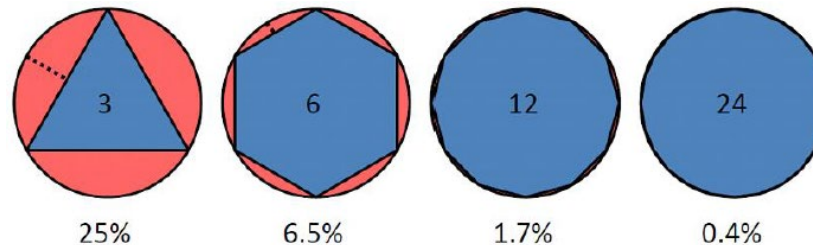
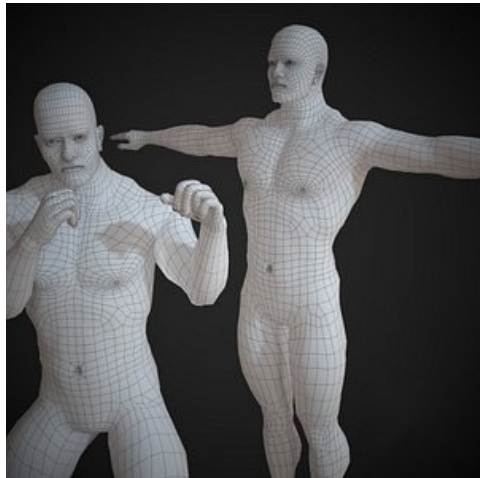
Some slides are borrowed from

- ECCV'22 Tutorial on Neural Volumetric Rendering for Computer Vision
- MIT 6.S980 – ML for Inverse Graphics – Vincent Sitzmann

3D Representations

Polygonal Meshes

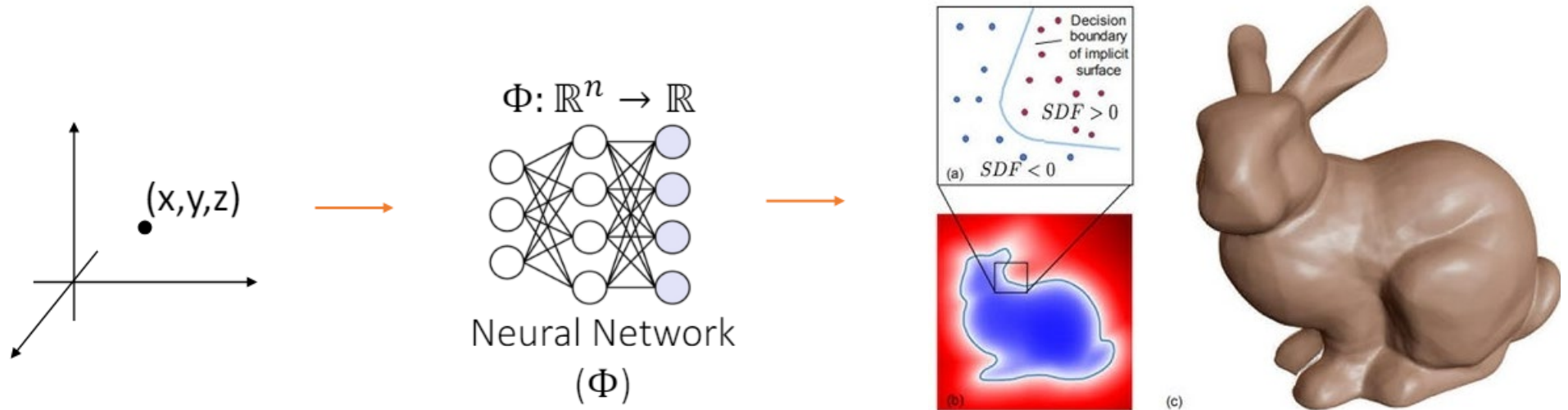
- Vertices + Faces (often triangles)
- Voronoi tessellation
- Piecewise linear
 - Attributes: colors, normals, textures



3D Representations

Implicit Surfaces and Neural Fields

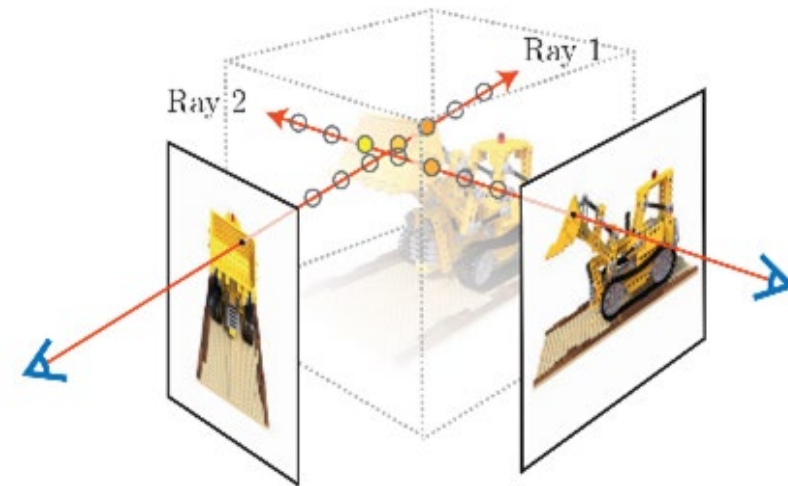
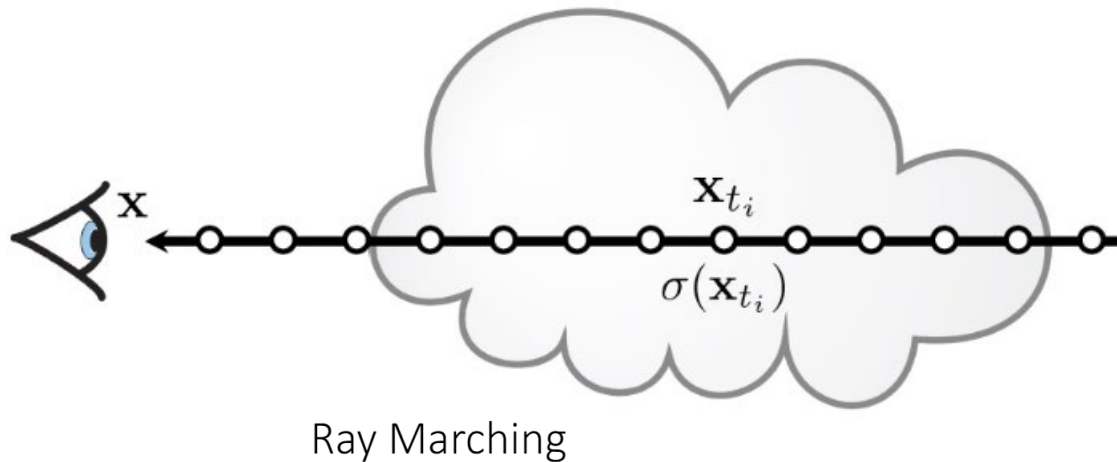
- Signed distance field (SDF)
- Note:
 - Explicit function: $y=f(x)$
 - Implicit function: $x^2+y^2-1=0$



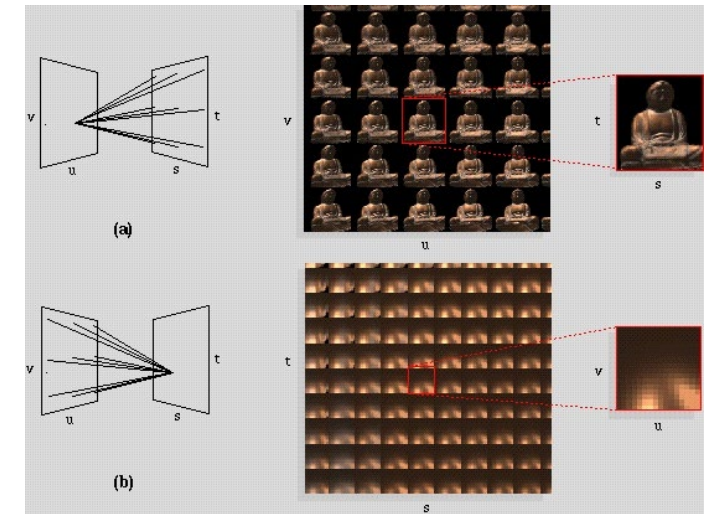
3D Representations

Implicit Surfaces and Neural Fields

- Neural Radiance Field or NeRF
- Objects as radiance/density fields (σ, \mathcal{C})
 - implicit fields (e.g. MLP)--> NeRF
 - explicit fields (e.g. 3D Gaussian Splatting)
- (Differentiable) Volumetric Rendering



Volume Rendering

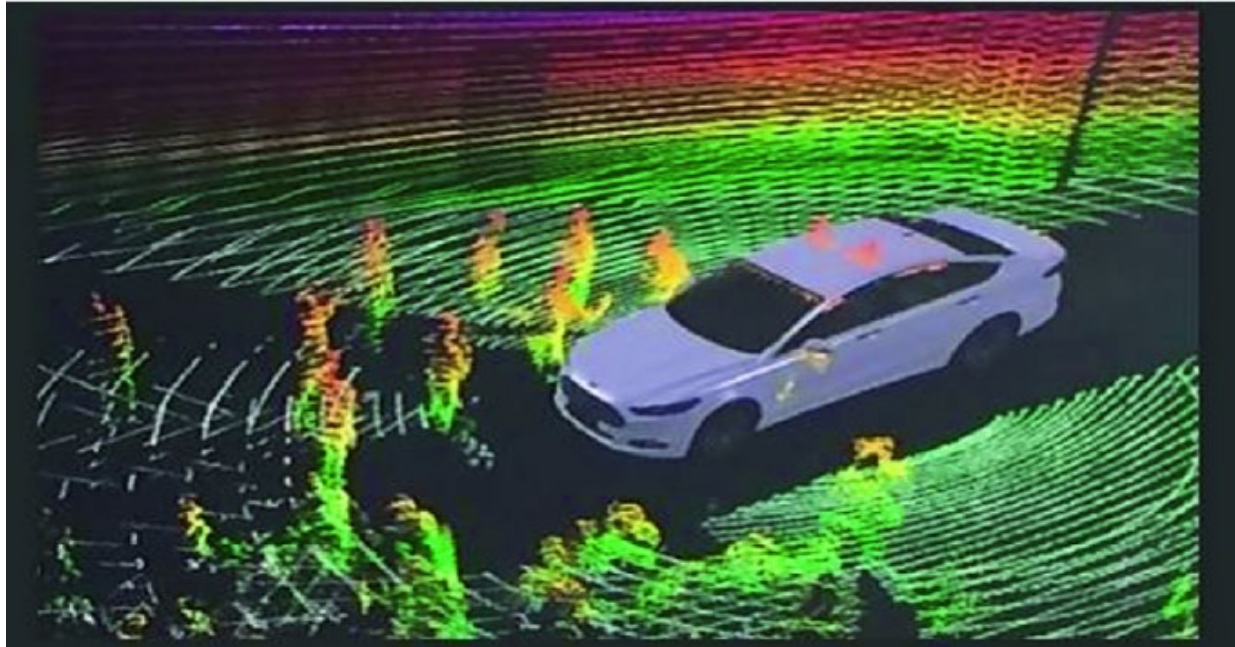
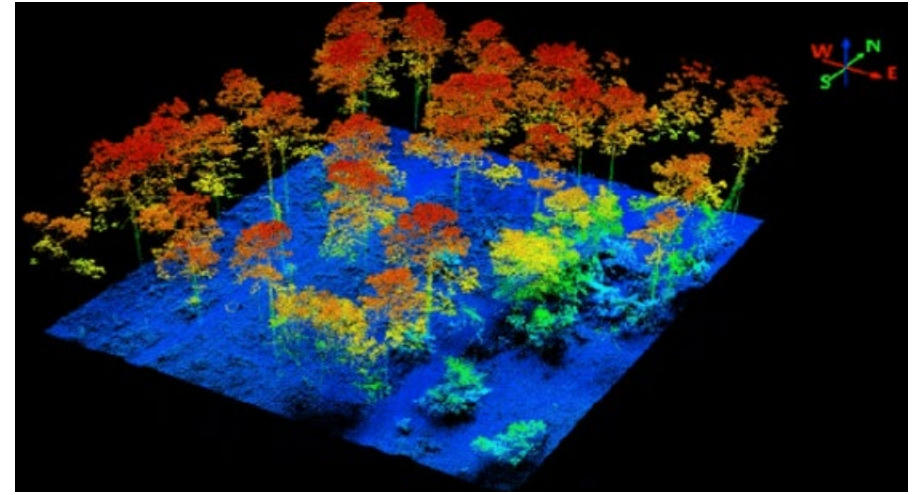


Light Field Representation

3D Representations

Point Clouds

- Set of 3D points with (x, y, z) coordinates
- Associated attributes
 - Color
 - Normal



3D Representations

Point Clouds

- 3D Gaussian Splatting



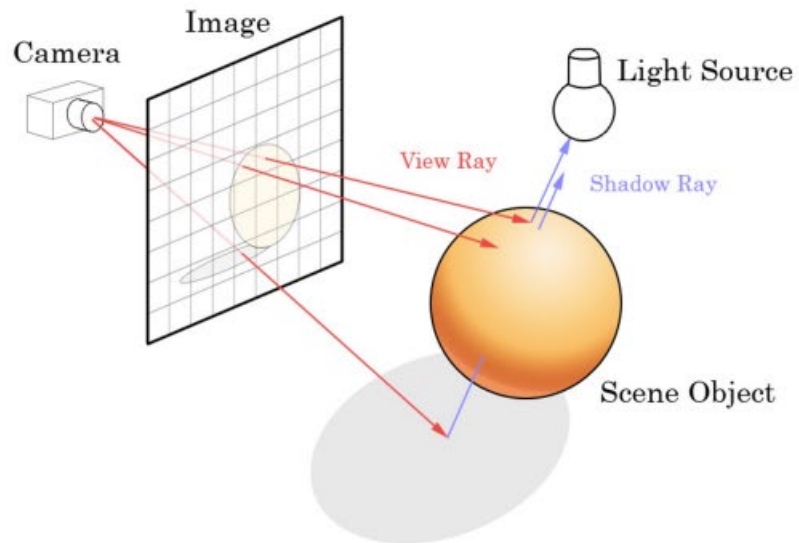
The Historic Context of 3DGS

- 3D Representations: Mesh, Point Cloud, Implicit Surface/Neural Field, etc.
- **Volumetric Rendering**
- Point Based Graphics

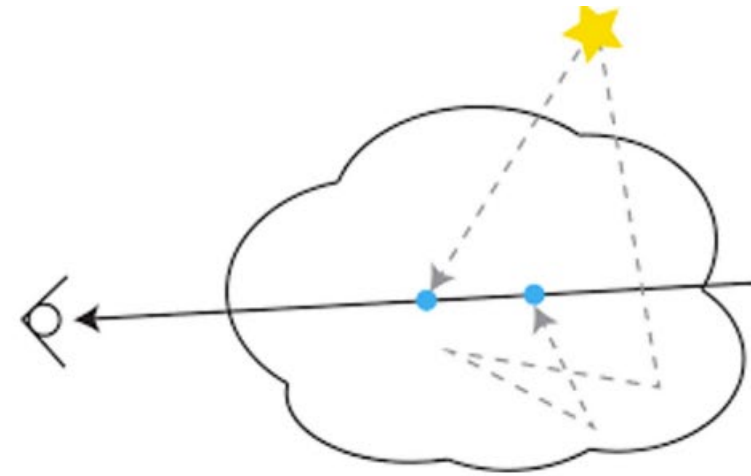
Some slides are borrowed from

- ECCV'22 Tutorial on Neural Volumetric Rendering for Computer Vision
- MIT 6.S980 – ML for Inverse Graphics – Vincent Sitzmann

Surface vs. volume rendering



Surface rendering



Volume rendering

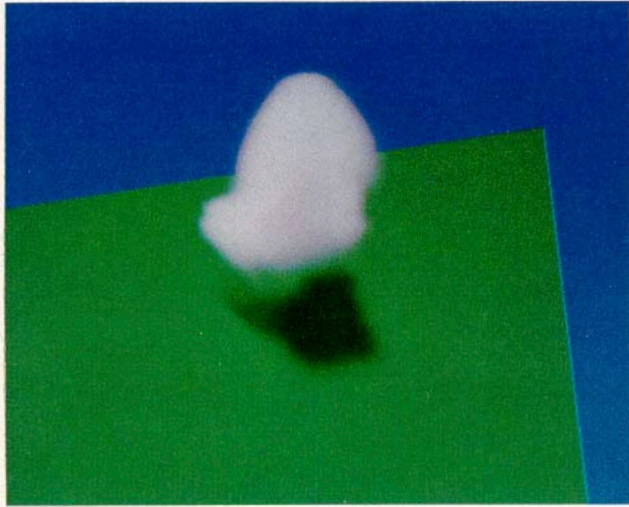
Want to know how ray interacts with scene

History of volume rendering

S.Chandrasekhar
RADIATIVE
TRANSFER

Theory of volume rendering

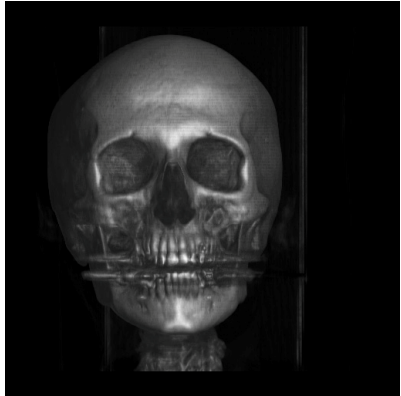
- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering



Ray tracing simulated cumulus cloud [Kajiya]

Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities

History of volume rendering



Medical data visualisation [Levoy]

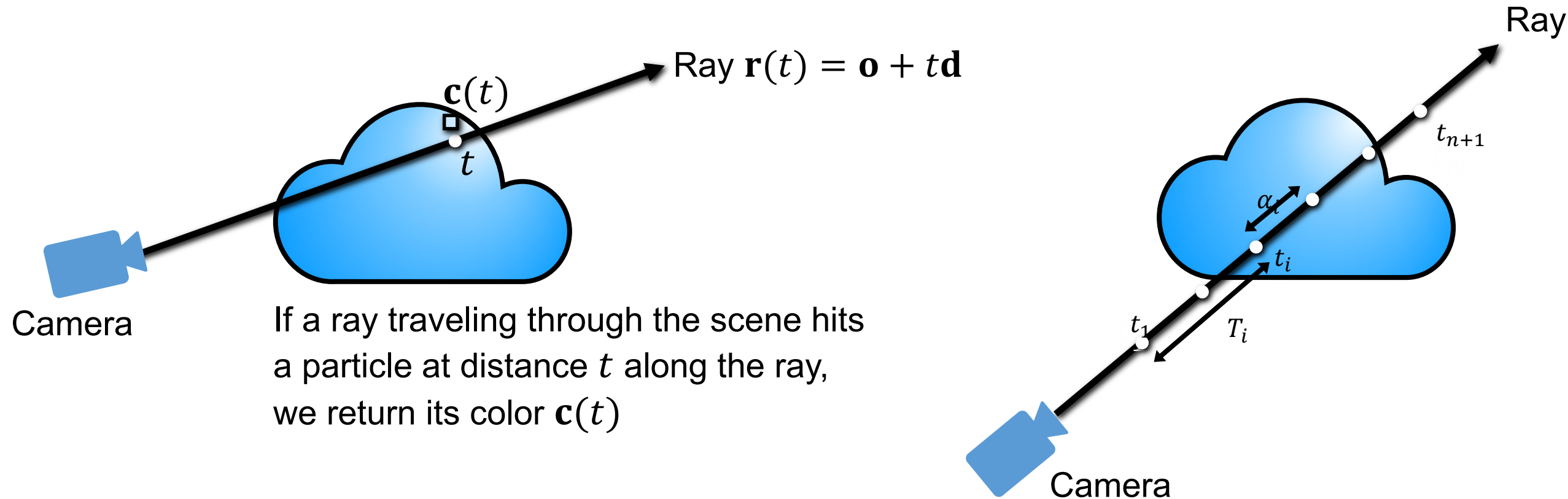
- ▶ Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- ▶ Volume rendering applied to visualise 3D medical scan data in 1990s

Volumetric formulation for NeRF

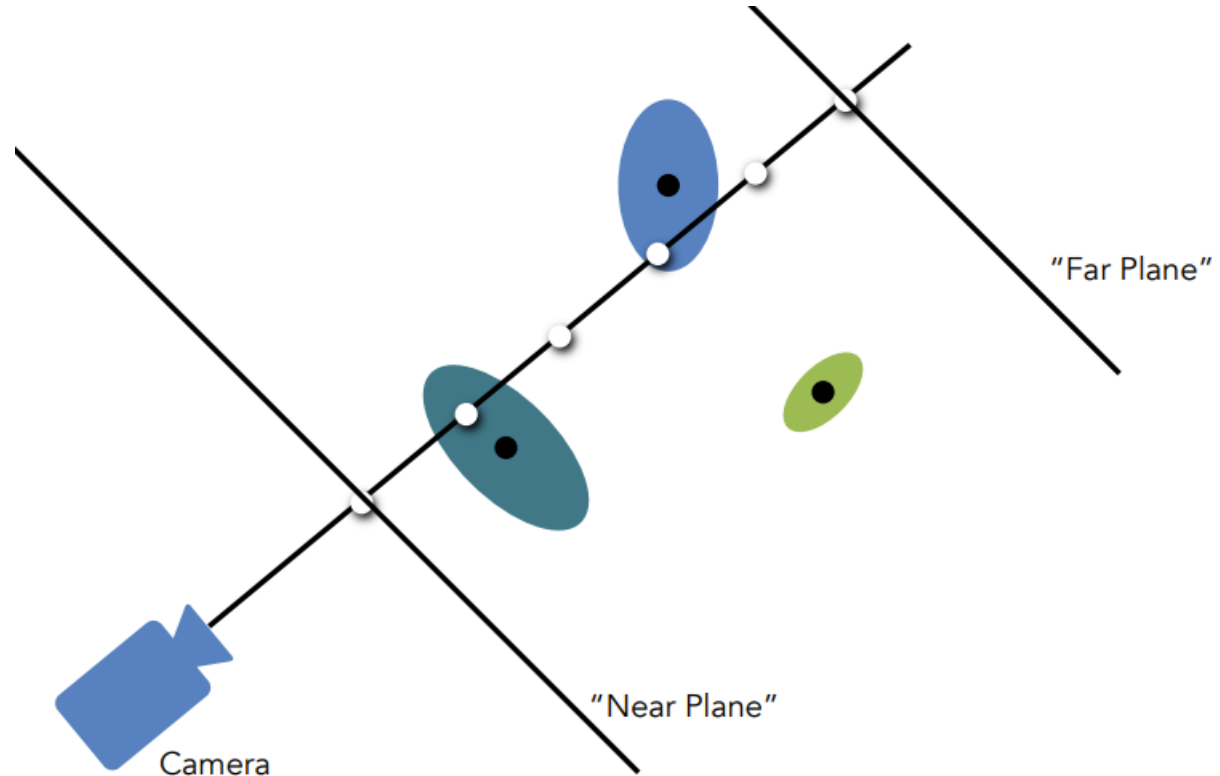


Scene is a cloud of tiny colored particles

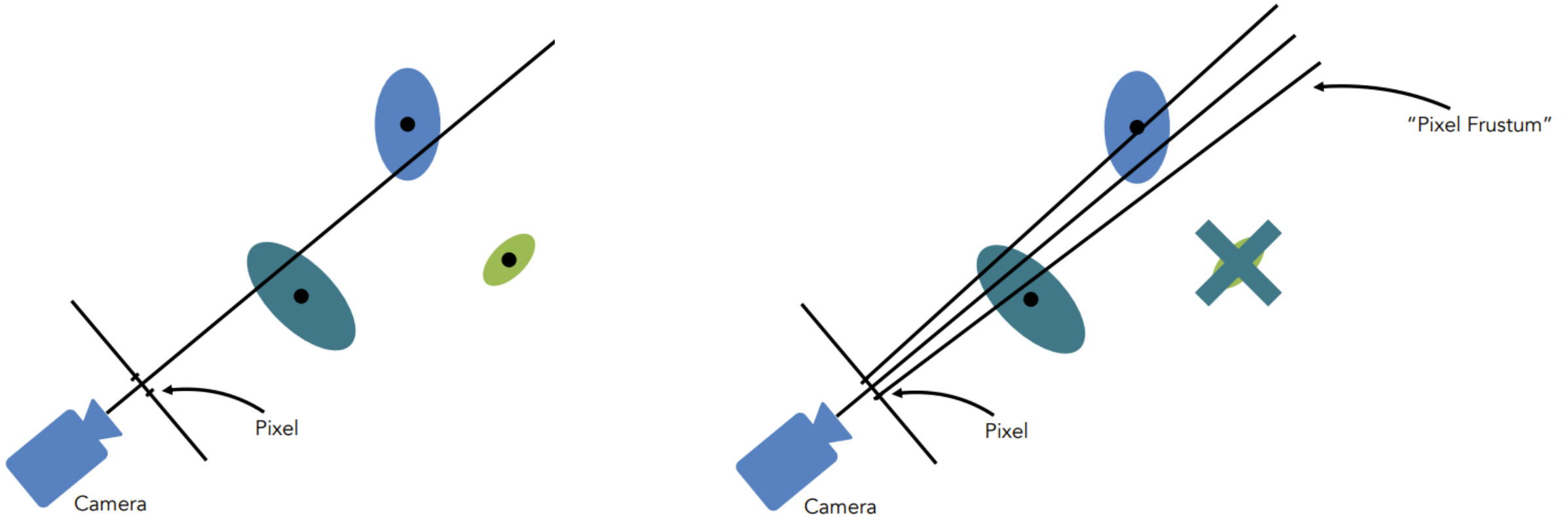
Volumetric formulation for NeRF



Volumetric formulation for 3DGS



Volumetric formulation for 3DGS: Discretization & Rasterization



Slide adopted from 6.S980 - ML for Inverse Graphics - Vincent Sitzmann

The specific volumetric rendering method used by 3DGS is the following one:
M. Zwicker, H. Pfister, J. van Baar, and M. Gross. 2001. EWA volume splatting. IEEE Conference on Visualization (VIS), 2001.

EWA Volume Splatting

Matthias Zwicker *

Hanspeter Pfister †

Jeroen van Baar†

Markus Gross*

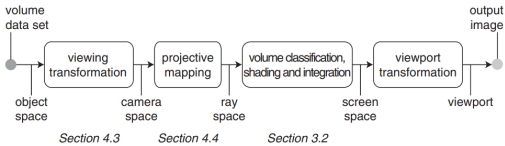


Figure 1: The forward mapping volume rendering pipeline.

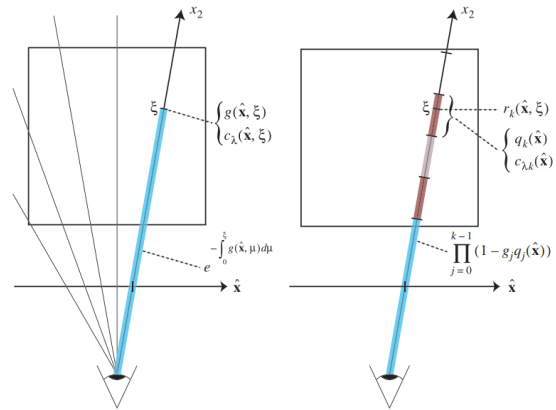


Figure 2: Volume rendering. Left: Illustrating the volume rendering equation in 2D. Right: Approximations in typical splatting algorithms.

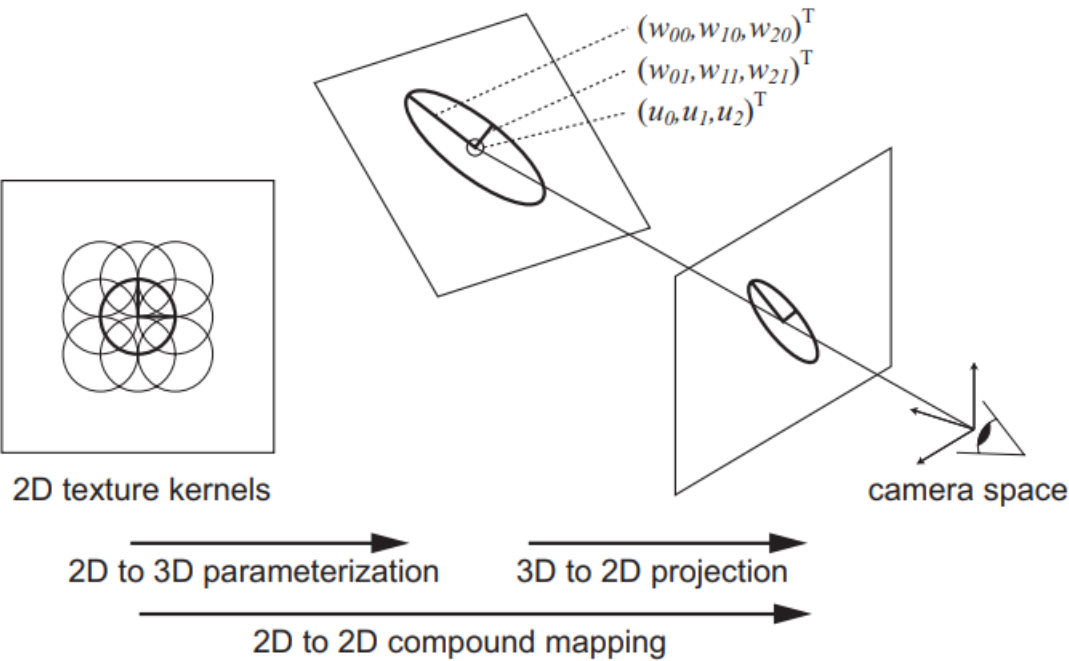


Figure 6: Rendering surface kernels.

The Historic Context of 3DGS

- 3D Representations: Mesh, Point Cloud, Implicit Surface/Neural Field, etc.
- Volumetric Rendering
- **Point Based Graphics**

Some slides are borrowed from

- ECCV'22 Tutorial on Neural Volumetric Rendering for Computer Vision
- MIT 6.S980 – ML for Inverse Graphics – Vincent Sitzmann

Point Set Surfaces, IEEE Visualization 2001

Point Set Surfaces

Marc Alexa TU Darmstadt Johannes Behr ZGDV Darmstadt Daniel Cohen-Or Tel Aviv University Shachar Fleishman Tel Aviv University David Levin Tel Aviv University Claudio T. Silva AT&T Labs

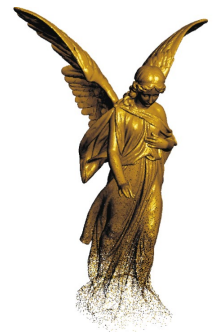


Figure 1: A point set representing a statue of an angel. The density of points and, thus, the accuracy of the shape representation are changing (intentionally) along the vertical direction.

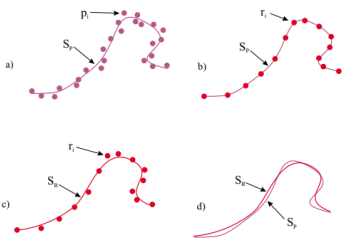


Figure 2: An illustration of the paradigm: The possibly noisy or redundant point set (purple points) defines a manifold (purple curve). This manifold is sampled with (red) representation points. The representation points define a different manifold (red curve). The spacing of representation points depends on the desired accuracy of the approximation.

Pulsar: Efficient sphere-based neural rendering, CVPR 2021

Pulsar: Efficient Sphere-based Neural Rendering

Christoph Lassner¹ Michael Zollhöfer¹
¹Facebook Reality Labs

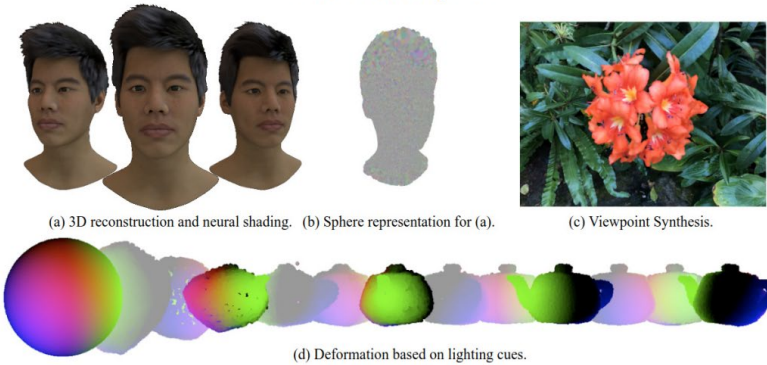


Figure 1: Pulsar is an efficient sphere-based differentiable rendering module that is orders of magnitude faster than competing techniques, modular, and easy-to-use. It can be employed to solve a large variety of applications, since it is tightly integrated with PyTorch. Using a sphere-based representation, it is possible to not only optimize for color and opacity, but also for positions and radii (a, b, c). Due to the modular design, lighting cues can also be easily integrated (d).

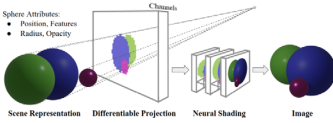


Figure 2: Visualization of the neural rendering pipeline. Pulsar enables a particularly fast differentiable projection step that scales to complex scene representations. The scene representation itself can be produced by a neural network. The channel information can be 'latent' and translated to RGB colors in a neural shading step.

Neural Point-Based Graphics, ECCV 2020, [video](#)

Neural Point-Based Graphics

Kara-Ali Aliev¹, Artem Sevastopolsky^{1,2}, Maria Kolos^{1,2}, Dmitry Ulyanov³, and Victor Lempitsky^{1,2}

¹ Samsung AI Center
² Skolkovo Institute of Science and Technology
³ In3D.io

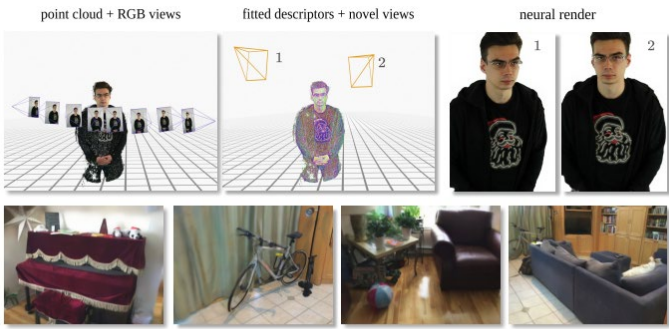
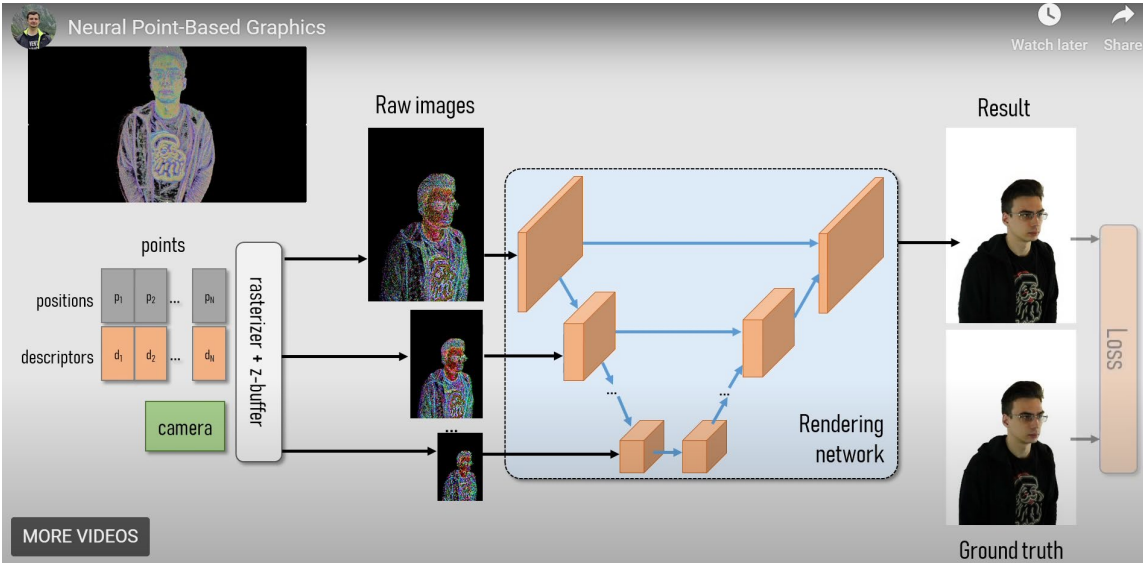


Fig. 1: Given a set of RGB views and a point cloud (top-left), our approach fits a neural descriptor to each point (top-middle), after which new views of a scene can be rendered (top-right). The method works for a variety of scenes including 3D portraits (top) and interiors (bottom).



Topics

- The historic context of 3DGS
- **Present: [3DGS](#) and its recent progress**
- Future discussions

Present: 3DGS, and how to make it stronger and better

- **3DGS**
- Directions for improvement

3D Gaussian Splatting (3DGS)

- **What is 3DGS**
 - **3D surface representation: a cloud of 3D Gaussian blobs**
 - Differentiable and rasterized volumetric renderer
 - Configuring the 3D points' locations by optimization
- A glimpse of 3DGS applications
- (Possible) issues with the original 3DGS work

3D Gaussian Splatting (3DGS)

A visual illustration of the 3D Gaussian blobs (source: [What is 3D Gaussian Splatting?](#))

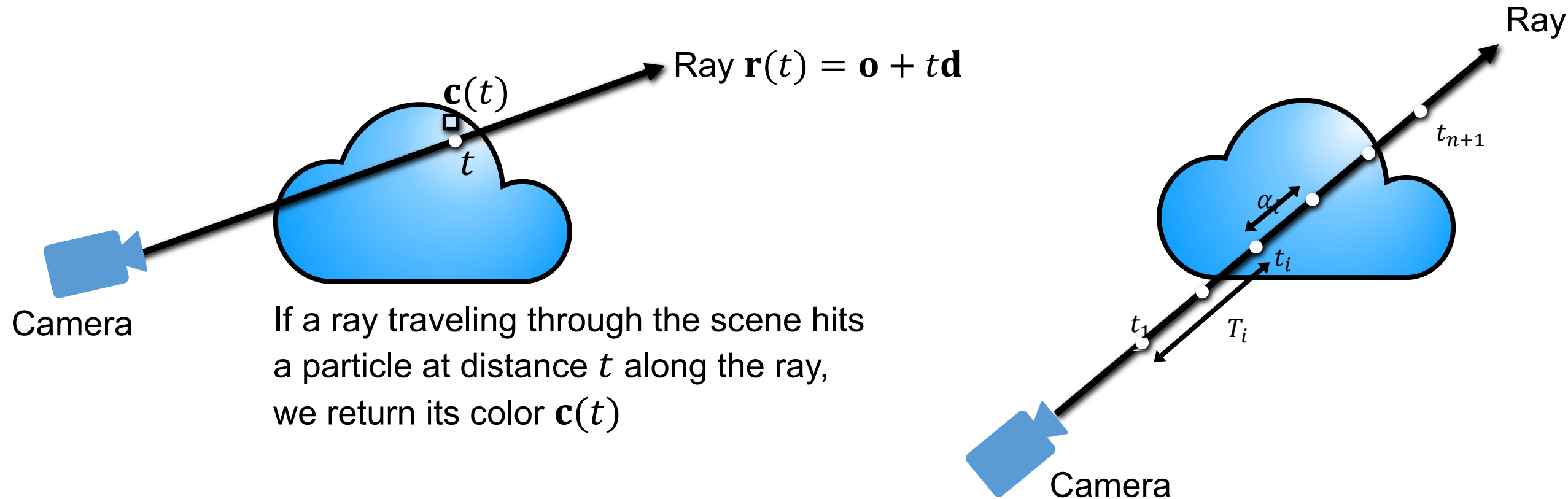


- 3D point cloud as basis,
- each 3D point wrapped with Gaussian blob,
- "water-tight",
- color based on spherical harmonics

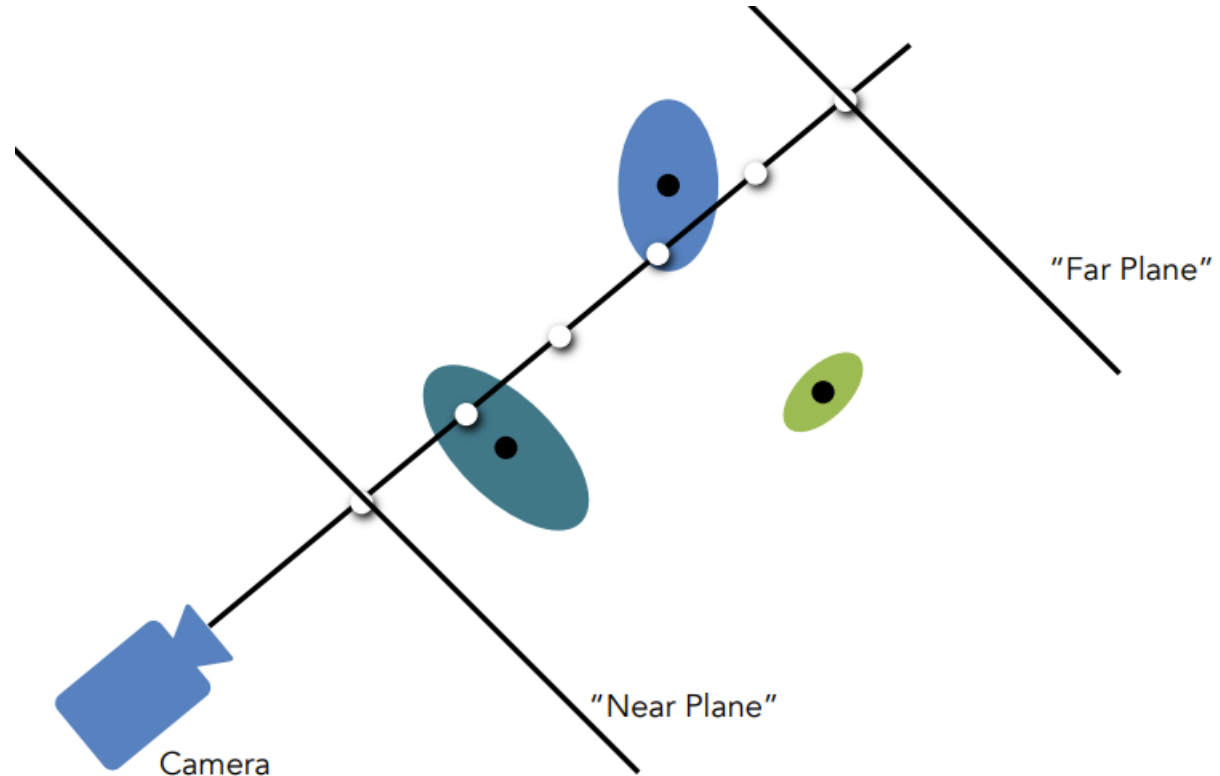
3D Gaussian Splatting (3DGS)

- What is 3DGS
 - 3D surface representation: a cloud of 3D Gaussian blobs
 - **Differentiable and rasterized volumetric renderer**
 - Configuring the 3D points' locations by optimization
- A glimpse of 3DGS applications
- (Possible) issues with the original 3DGS work

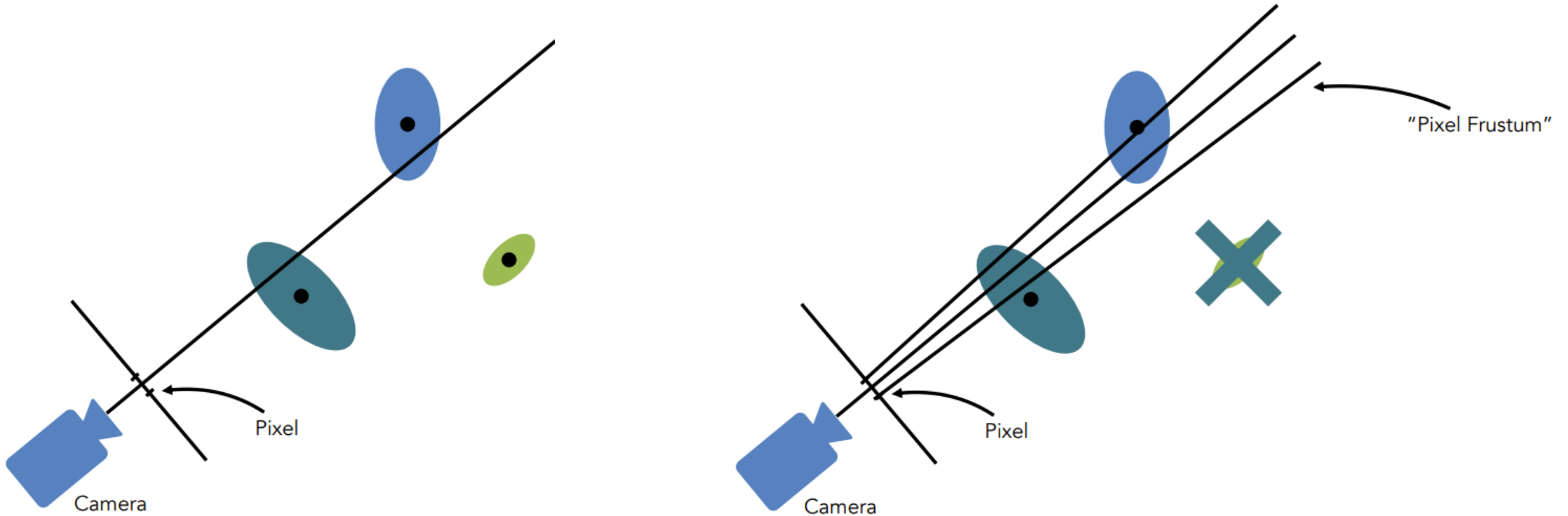
Recall: volumetric formulation for NeRF



Volumetric formulation for 3DGS



Volumetric formulation for 3DGS: Discretization & Rasterization



Slide adopted from 6.S980 - ML for Inverse Graphics - Vincent Sitzmann

The specific volumetric rendering method used by 3DGS is the following one:
M. Zwicker, H. Pfister, J. van Baar, and M. Gross. 2001. EWA volume splatting. IEEE Conference on Visualization (VIS), 2001.

EWA Volume Splatting

Matthias Zwicker *

Hanspeter Pfister †

Jeroen van Baar†

Markus Gross*

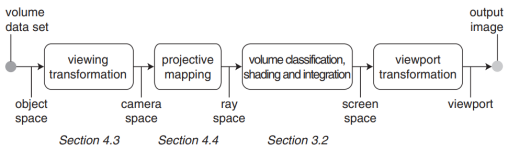


Figure 1: The forward mapping volume rendering pipeline.

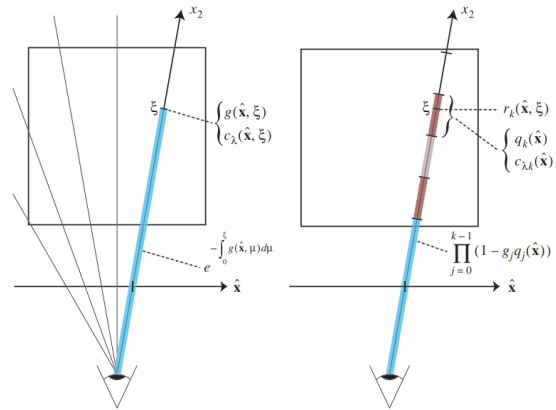


Figure 2: Volume rendering. Left: Illustrating the volume rendering equation in 2D. Right: Approximations in typical splatting algorithms.

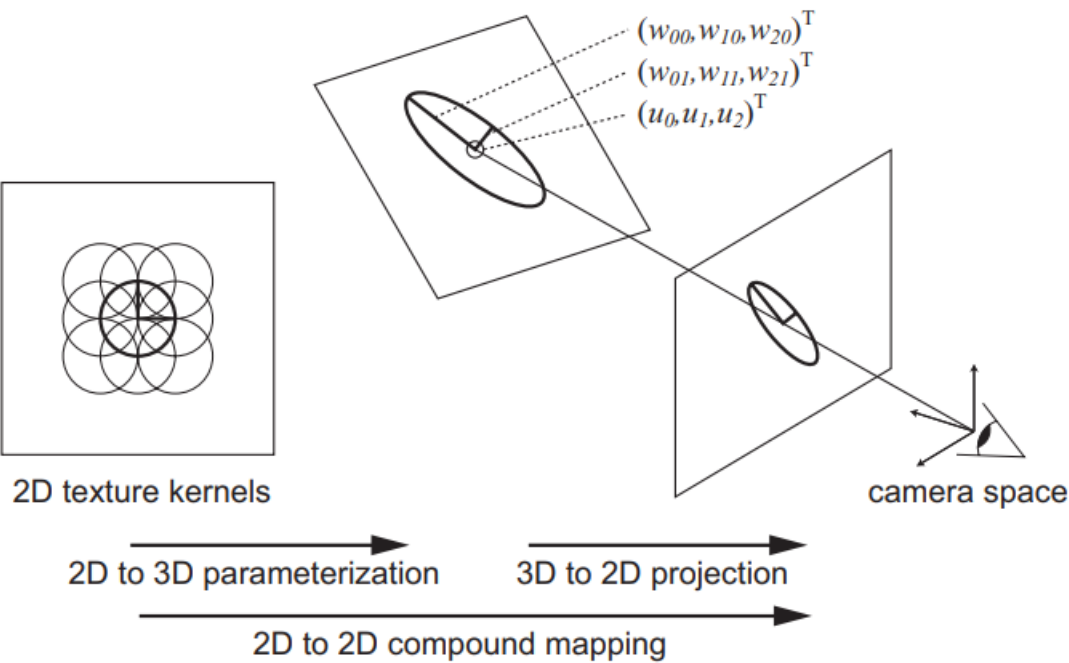


Figure 6: Rendering surface kernels.

3D Gaussian Splatting (3DGS)

- What is 3DGS
 - 3D surface representation: a cloud of 3D Gaussian blobs
 - Differentiable and rasterized volumetric renderer
 - **Configuring the 3D points' locations by optimization**
- A glimpse of 3DGS applications
- (Possible) issues with the original 3DGS work

3D Gaussian Splatting (3DGS)

Controlling the 3D points' locations by optimization

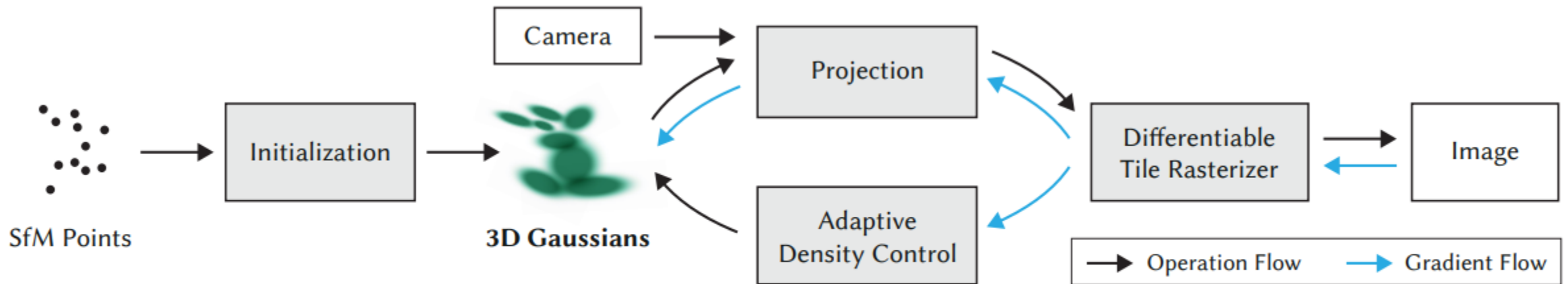
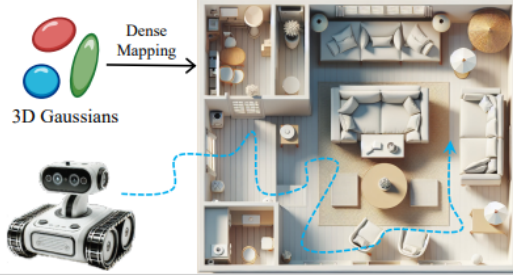
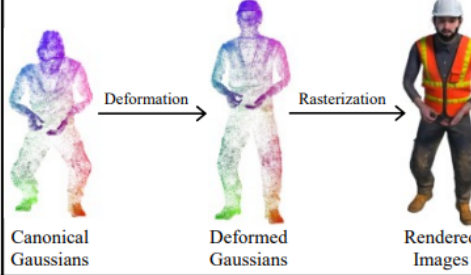



Fig. 2. Optimization starts with the sparse SfM point cloud and creates a set of 3D Gaussians. We then optimize and adaptively control the density of this set of Gaussians. During optimization we use our fast tile-based renderer, allowing competitive training times compared to SOTA fast radiance field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

3D Gaussian Splatting (3DGS)

- What is 3DGS
- **A glimpse of 3DGS applications**
 - Visual SLAM
 -
- (Possible) issues with the original 3DGS work

A glimpse of 3DGS applications

Robotics (Sec. 5.1)		Dynamic Scene Reconstruction (Sec. 5.2)		Generation & Editing (Sec. 5.3)	
 <p>Dense Mapping</p> <p>3D Gaussians</p>		 <p>Canonical Gaussians</p> <p>Deformation</p> <p>Deformed Gaussians</p> <p>Rasterization</p> <p>Rendered Images</p>		 <p>sorcerer's spellbook</p> <p>mushroom house</p>	
[111] , [112] , [113] , [114] , [115] , [116] , [117] , [118] , [119] , [120] , [121] , [122] , [123] , [124]		[125] , [94] , [95] , [93] , [126] , [127] , [128] , [129] , [106] , [130] , [131] , [132]		[133] , [134] , [135] , [136] , [137] , [138] , [139] , [140] , [141] , [142] , [90] , [143]	

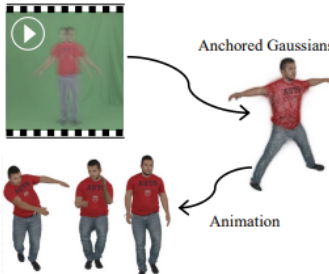
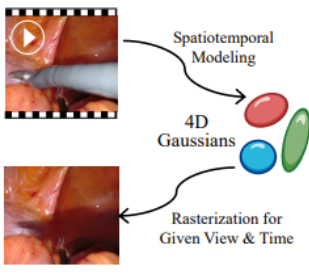
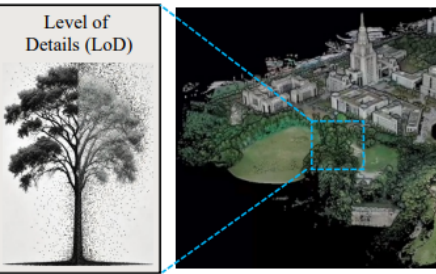
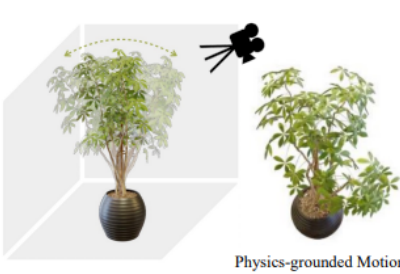
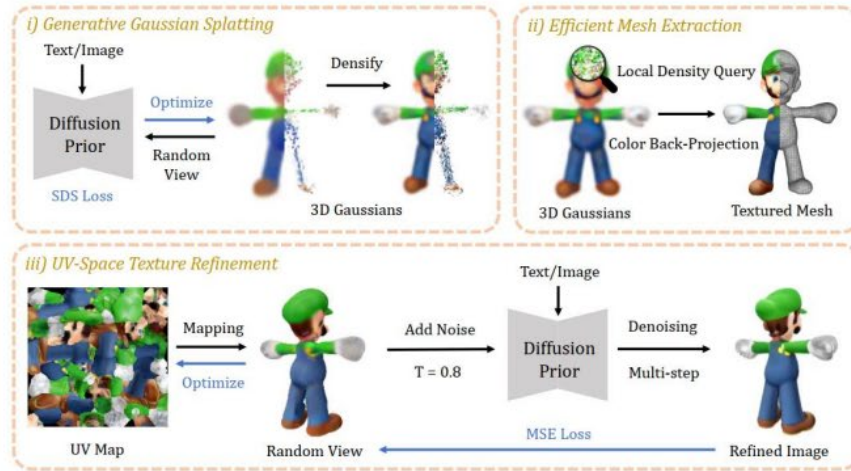
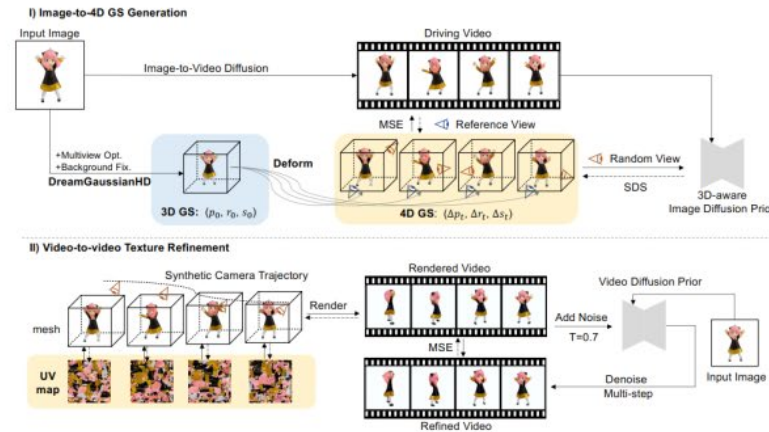
Avatar (Sec. 5.4)		Endoscopic Scene Reconstruction (Sec. 5.5)		Large-scale Scene Reconstruction (Sec. 5.6)		Physics (Sec. 5.7)	
 <p>Anchored Gaussians</p> <p>Animation</p>		 <p>Spatiotemporal Modeling</p> <p>4D Gaussians</p> <p>Rasterization for Given View & Time</p>		 <p>Level of Details (LoD)</p>		 <p>Physics-grounded Motion</p>	
[148] , [149] , [150] , [151] [144] , [145] , [146] , [147]		[155] , [156] , [157] [152] , [153] , [154]		[44] , [162] , [163] , [164] , [165] [158] , [159] , [160] , [161]		[170] , [21] , [171] , [172] , [173] [166] , [167] , [168] , [143] , [169]	

Fig. 6. Typical applications benefited from GS (Sec. 5). Some images are borrowed from [\[132\]](#), [\[135\]](#), [\[146\]](#), [\[154\]](#), [\[160\]](#), [\[166\]](#) and redrawn.

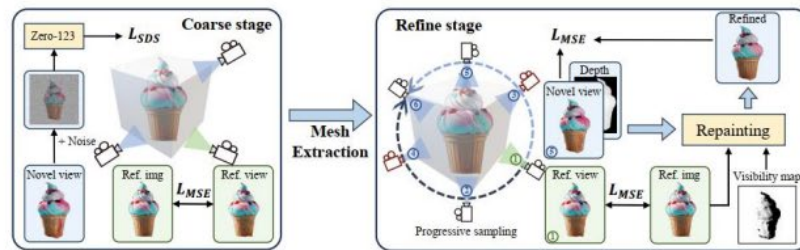
A glimpse of 3DGS applications



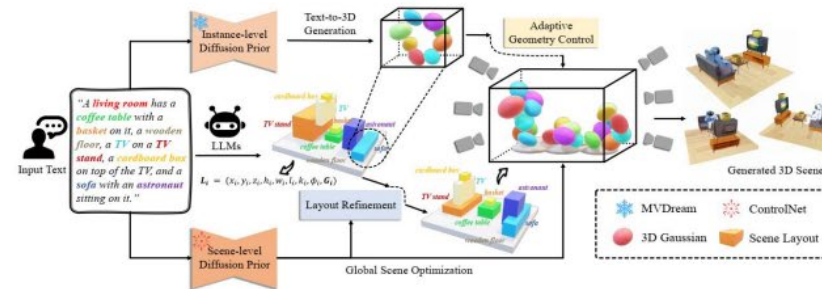
(a) Text to 3D Objects [95].



(b) 4D Generation [96].



(c) Image to 3D Object [97].



(d) Multi-Object Generation [98].

Fig. 3: Four typical tasks of 3DGC in AIGC Applications.

3D Gaussian Splatting (3DGS)

- What is 3DGS
- A glimpse of 3DGS applications
 - **Visual SLAM**
 -
- (Possible) issues with the original 3DGS work

Jianhao Zheng et al. WildGS-SLAM: Monocular Gaussian Splatting SLAM in Dynamic Environments, CVPR 2025

WildGS-SLAM

Monocular Gaussian Splatting SLAM in Dynamic Environments

CVPR 2025

Jianhao Zheng^{1*} Zihan Zhu^{2*} Valentin Bieri² Marc Pollefeys^{1,3} Songyou Peng² Iro Armeni¹

* Equal Contribution

¹Stanford University ²ETH Zürich ³Microsoft

[Paper](#) [Video](#) [Code](#)

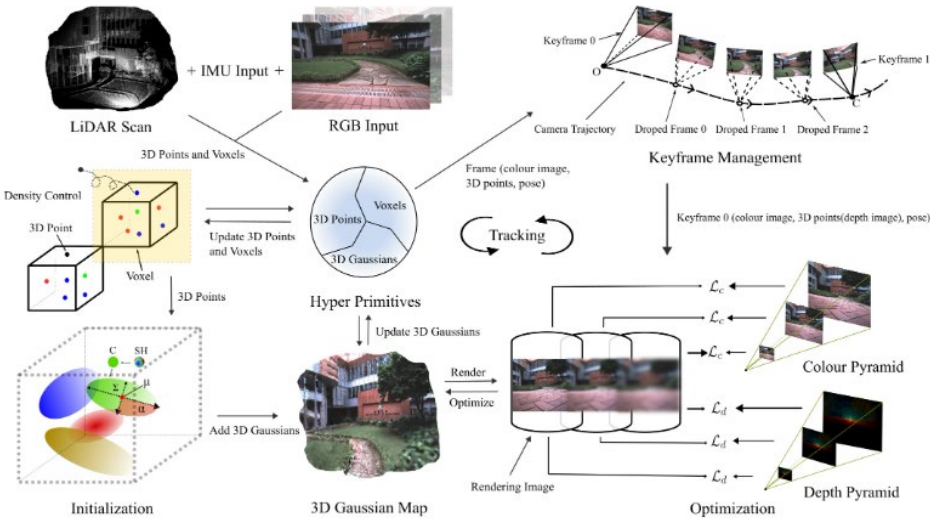
LVI-GS: Tightly-coupled LiDAR-Visual-Inertial SLAM using 3D Gaussian Splatting

Huibin Zhao^{1*}, Weipeng Guan^{1*}, Peng Lu¹

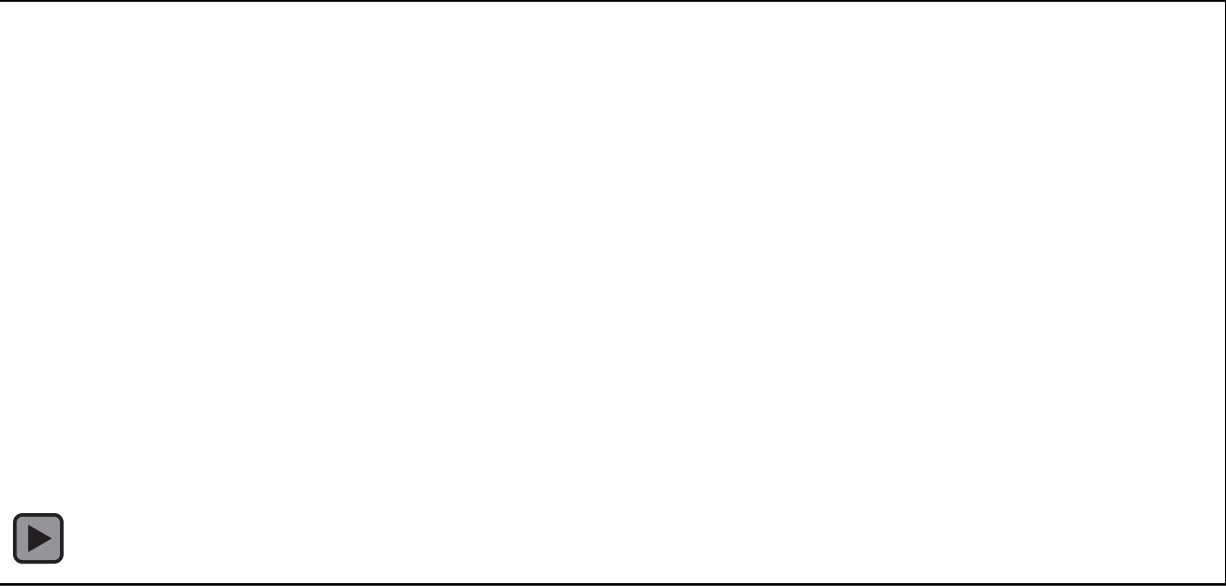
¹The University of Hong Kong

*Equal Contribution

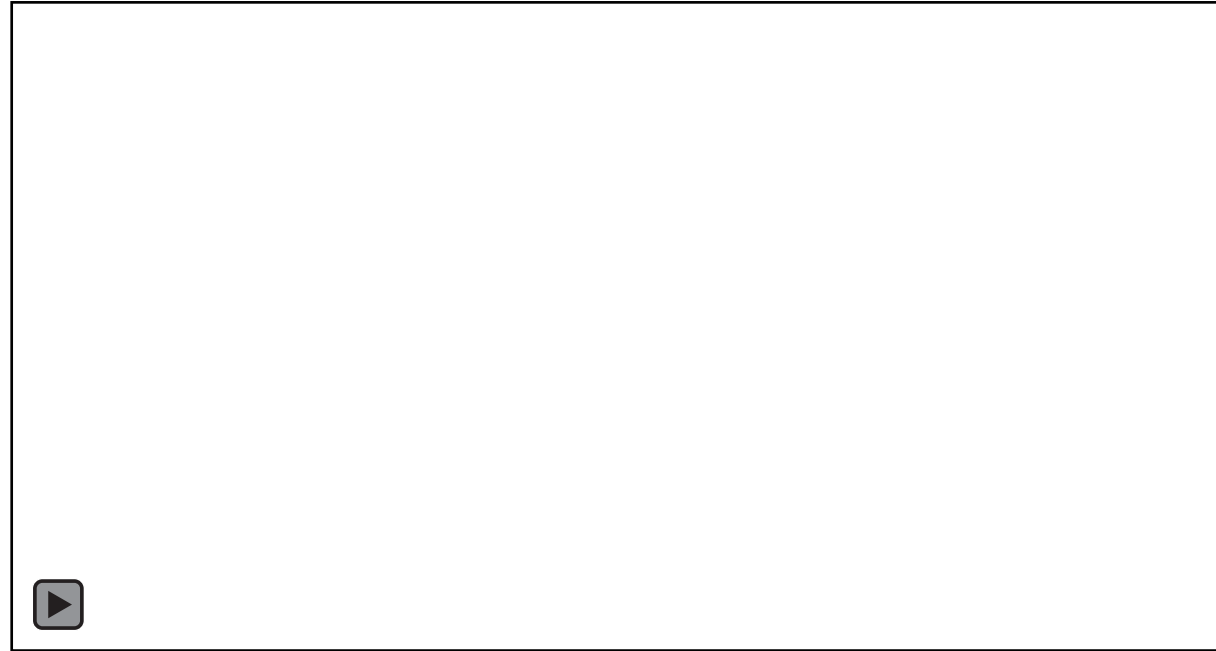
[Paper](#) [arXiv](#) [Code](#)



3DGS in building and interior designs:



3DGS + visual SLAM in Inria Campus:



3D Gaussian Splatting (3DGS)

- What is 3DGS
- A glimpse of 3DGS applications
- **(Possible) issues with the original 3DGS work**

3D Gaussian Splatting (3DGS)

- **(Possible) issues with 3DGS**
 - Significant memory consumption
 - Somewhat bumpy surface representation
 - Adapt the locations and shapes the cloud of 3D Gaussian blobs
 - Issues with shadows and relighting, in comparison to physics based rendering (ray tracing)
 - Static 3D scenes, no temporal capacity
 - No learning involved, in comparison to NeRF
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - Issues with shadows, relighting, and improving rendering quality
 - Temporal 3DGS
 - Learning based approaches

3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - **Less bumpy surface representation**
 - **2DGS / Gaussian Surfel**
 - Better basis as a cloud of 3D points
 - Issues with shadows and improving rendering quality
 - Temporal 3DGS
 - Learning based approaches
 - Attribute extension

2D Gaussian Splatting for Geometrically Accurate Radiance Fields, ACM Siggraph, 2024, [demo video](#)

2D Gaussian Splatting for Geometrically Accurate Radiance Fields

Binbin Huang
huangbb@shanghaitech.edu.cn
ShanghaiTech University
Shanghai, China

Zehao Yu
zehao.yu@uni-tuebingen.de
University of Tübingen
Tübingen AI Center
Tübingen, Germany

Anpei Chen
anpei.chen@uni-tuebingen.de
University of Tübingen
Tübingen AI Center
Tübingen, Germany

Andreas Geiger
a.geiger@uni-tuebingen.de
University of Tübingen
Tübingen AI Center
Tübingen, Germany

Shenghua Gao
gaoshh@shanghaitech.edu.cn
ShanghaiTech University
Shanghai, China

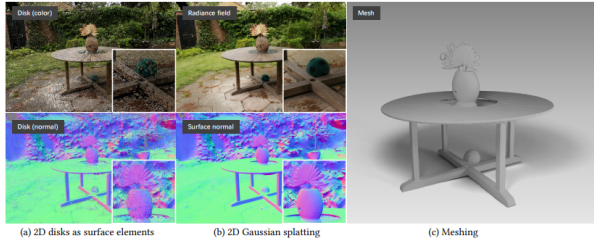


Figure 1: Our method, 2DGS, (a) optimizes a set of 2D oriented disks to represent and reconstruct a complex real-world scene from multi-view RGB images. These optimized 2D disks are tightly aligned to the surfaces. (b) With 2D Gaussian splatting, we allow real-time rendering of high quality novel view images with view consistent normals and depth maps. (c) Finally, our method provides detailed and noise-free triangle mesh reconstruction from the optimized 2D disks.

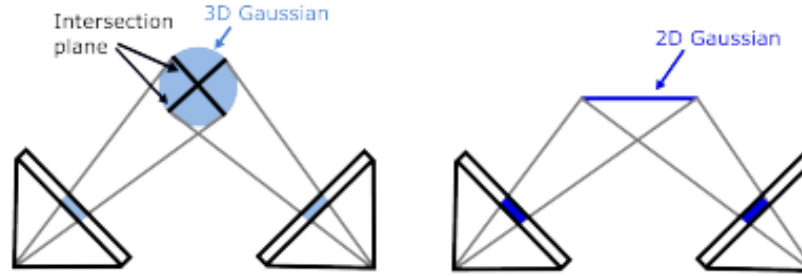


Figure 2: Comparison of 3DGS and 2DGS. 3DGS utilizes different intersection planes for value evaluation when viewing from different viewpoints, resulting in inconsistency. Our 2DGS provides multi-view consistent value evaluations.

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



Figure 1. We introduce a method that extracts accurate and editable meshes from 3D Gaussian Splatting representations within minutes on a single GPU. The meshes can be edited, animated, composited, etc. with very realistic Gaussian Splatting rendering, offering new possibilities for Computer Graphics. Note for example that we changed the posture of the robot between the captured scene on the bottom left and the composited scene on the right. The supplementary material provides more examples, including a video illustrating our results.

- precise and extremely fast mesh extraction from 3DGS
- a regularization term encourages the Gaussians to align well with the surface of the scene
 - exploits this alignment to extract a mesh from the Gaussians using Poisson reconstruction, which is fast, scalable, and preserves details, instead of the Marching Cubes

3D Gaussian Splatting (3DGS)


- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - **Better basis as a cloud of 3D points**
 - Issues with shadows and improving rendering quality
 - Temporal 3DGS
 - Learning based approaches
 - Attribute extension


3D Gaussian Splatting as Markov Chain Monte Carlo


Shakiba Kheradmand¹, Daniel Rebain¹, Gopal Sharma¹, Weiwei Sun¹,
Yang-Che Tseng¹, Hossam Isack², Abhishek Kar²
Andrea Tagliasacchi^{3, 4, 5} Kwang Moo Yi¹

¹University of British Columbia ²Google Research ³Google DeepMind

⁴Simon Fraser University ⁵University of Toronto

 Paper

 arXiv

 Video

 Code



3DGS vs. 3DGS-MCMC

Optimization done by 3DGS (SGD)

$$g \leftarrow g - \lambda_{lr} \cdot \nabla_g \mathbb{E}_{\mathbf{I} \sim \mathcal{I}} [\mathcal{L}_{total}(g; \mathbf{I})]$$



Optimization done by 3DGS-MCMC (SGLD)

$$g \leftarrow g - \lambda_{lr} \cdot \nabla_g \mathbb{E}_{\mathbf{I} \sim \mathcal{I}} [\mathcal{L}_{total}(g; \mathbf{I})] + \lambda_{noise} \cdot \epsilon$$

SGLD: Stochastic Gradient Langevin Dynamics



3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - **Issues with shadows and improving rendering quality**
 - **Neural Renderer**
 - Ray tracing
 - Temporal 3DGS
 - Learning based approaches
 - Attribute extension

Kara-Ali Aliev¹, Artem Sevastopolsky^{1,2}, Maria Kolos^{1,2}, Dmitry Ulyanov³,
Victor Lempitsky^{1,2}

¹Samsung AI Center, ²Skolkovo Institute of Science and Technology, ³In3D.io

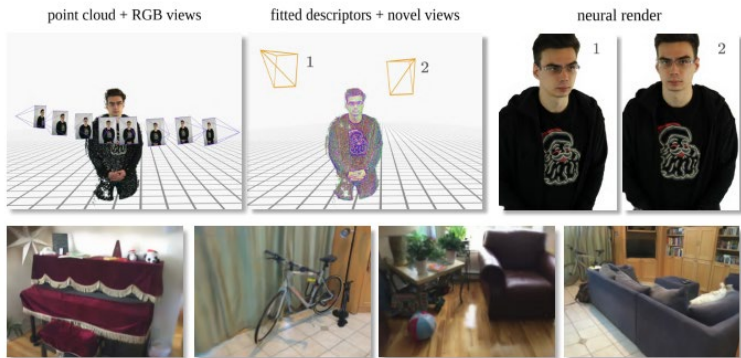


Fig. 1: Given a set of RGB views and a point cloud (top-left), our approach fits a neural descriptor to each point (top-middle), after which new views of a scene can be rendered (top-right). The method works for a variety of scenes including 3D portraits (top) and interiors (bottom).

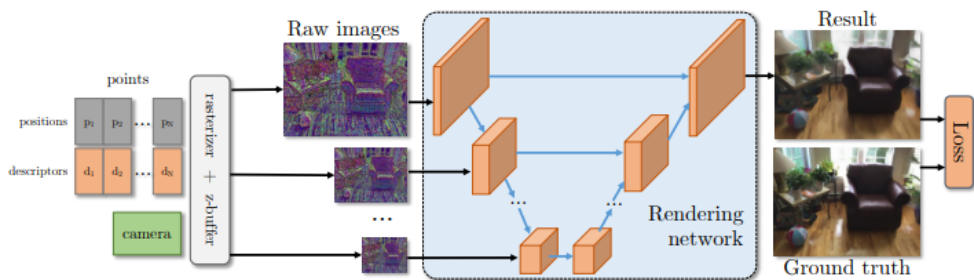


Fig. 2: An overview of our system. Given the point cloud \mathbf{P} with neural descriptors \mathbf{D} and camera parameters \mathbf{C} , we rasterize the points with z-buffer at several resolutions, using descriptors as pseudo-colors. We then pass the rasterizations through the U-net-like rendering network to obtain the resulting image. Our model is fit to new scene(s) by optimizing the parameters of the rendering network and the neural descriptors by backpropagating the perceptual loss function.

Point-NeRF: Point-based Neural Radiance Fields

Qiangeng Xu¹ [†] Zexiang Xu² Julien Philip² Sai Bi² Zhixin Shu²

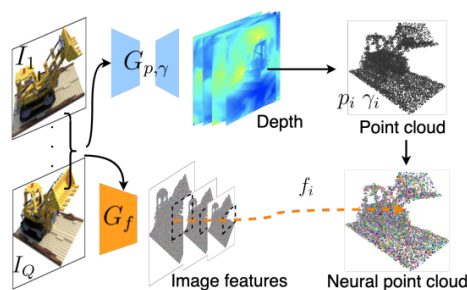
Kalyan Sunkavalli²

Ulrich Neumann¹

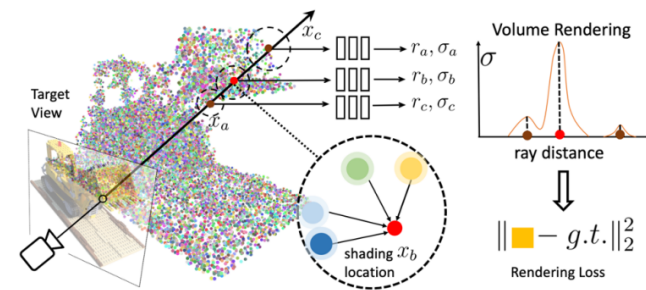
²Adobe Research

¹University of Southern California

{qiangenx, uneumann}@usc.edu {zexu, juphilip, sbi, zshu, sunkaval}@adobe.com



(a) Neural Point Generation.



(b) Point-NeRF Representation with Volume Rendering.

Figure 2. Overview of Point-NeRF. (a) From multi-view images, our model generates depth for each view by using a cost volume-based 3D CNNs $G_{p,\gamma}$ and extract 2D features from the input images by a 2D CNN G_f . After aggregating the depth map, we obtain a point-based radiance field in which each point has a spatial location p_i , a confidence γ_i and the unprojected image features f_i . (b) To synthesize a novel view, we conduct differentiable ray marching and compute shading only nearby the neural point cloud (e.g., x_a, x_b, x_c). At each shading location, Point-NeRF aggregates features from its K neural point neighbors and compute radiance r and volume density σ then accumulate r using σ . The entire process is end-to-end trainable and the point-based radiance field can be optimized with the rendering loss.

3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - Issues with shadows and improving rendering quality
 - Neural Renderer
 - **Ray tracing**
 - Temporal 3DGS
 - Learning based approaches
 - Attribute extension

3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes, Siggraph Asia 2024, [project website](#)

3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes

NICOLAS MOENNE-LOCCOZ*, NVIDIA, Canada

ASHKAN MIRZAEI*, NVIDIA, Canada and University of Toronto, Canada

OR PEREL, NVIDIA, Israel

RICCARDO DE LUTIO, NVIDIA, USA

JANICK MARTINEZ ESTURO, NVIDIA, Germany

GAVRIEL STATE, NVIDIA, Canada

SANJA FIDLER, NVIDIA, Canada, University of Toronto, Canada, and Vector Institute, Canada

NICHOLAS SHARP†, NVIDIA, USA

ZAN GOJCIC†, NVIDIA, Switzerland



Fig. 1. We propose a method for fast forward and inverse ray tracing of particle-based scene representations such as Gaussians. The main idea is to construct encapsulating primitives around each particle, and insert them into a BVH to be rendered by a ray tracer specially adapted to the high density of overlapping particles. Efficient ray tracing opens the door to many advanced techniques, including secondary ray effects like mirrors, refractions and shadows, as well as highly-distorted cameras with rolling shutter effects and even stochastic sampling of rays. **Project page:** [GaussianTracer.github.io](#)

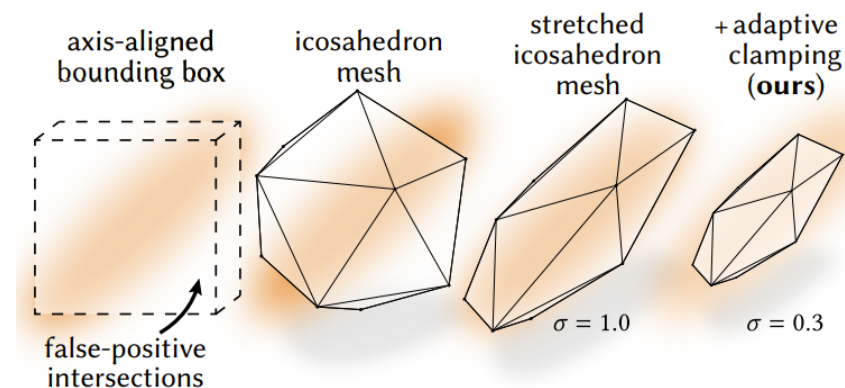


Fig. 4. **Proxy Geometries:** Examples of BVH primitives considered.

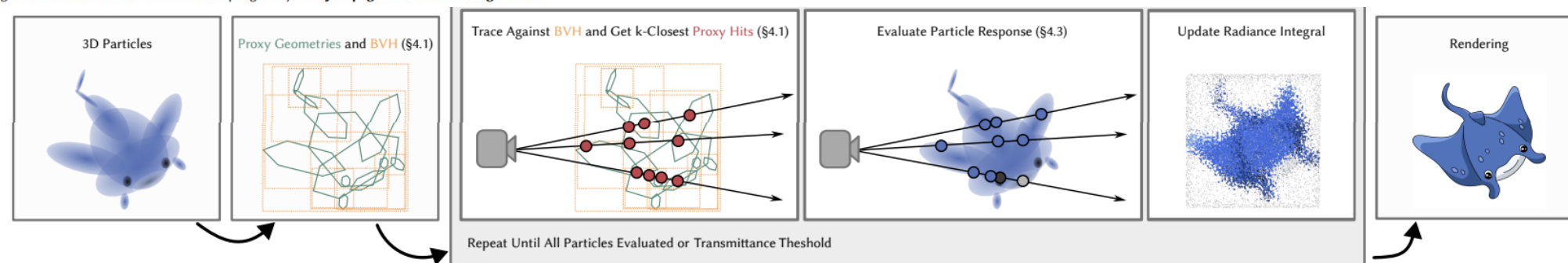


Fig. 3. **Overview of the Accelerated Tracing Algorithm:** Given a set of 3D particles, we first build the corresponding bounding primitives and insert them into a BVH. To compute the incoming radiance along each ray, we trace rays against the BVH to get the next k particles. We then compute the intersected particles' response and accumulate the radiance according to Equation 6. The process repeats until all particles have been evaluated or the transmittance meets a predefined threshold and the final rendering is returned.

Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing, ECCV 2024

Relightable 3D Gaussian: Real-time Point Cloud Relighting with BRDF Decomposition and Ray Tracing

Jian Gao^{1*}, Chun Gu^{2*}, Youtian Lin¹, Hao Zhu¹, Xun Cao¹, Li Zhang²✉, Yao Yao¹✉

¹ Nanjing University, ² Fudan University

*Equally contributed.

Paper

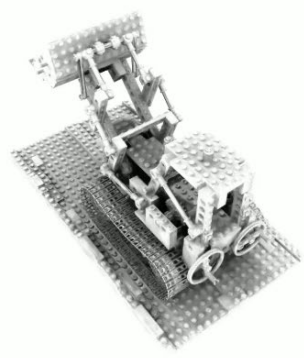
Code



Points



PBR with Decomposed BRDF



Ambient Occlusion from Point-based Ray Tracing

Rendering

Relighting1



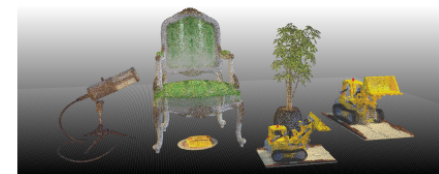
Relighting2



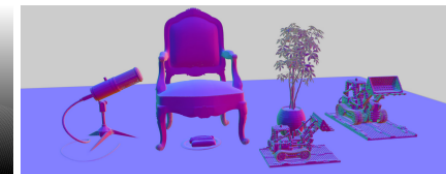
Garden



Kitchen



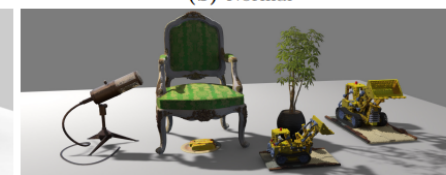
(a) Point Cloud



(b) Normal



(c) Ambient Occlusion



(d) Physically Based Rendering

Fig. 1: Visual results of our pipeline on a multi-object composition scene. In our pipeline, we represent a scene as *Relightable 3D Gaussians*. From multi-view images, we recover the geometry and materials of individual objects with *inverse rendering* techniques (see Sec. 3). Then, objects are easily composited into a new scene, thanks to our explicit representation. After that, we solve the complex occlusions through *point based ray tracing* (see Sec. 4) and re-light the new scene. Ultimately, we achieve high-fidelity relighting with remarkably **realistic shadow**.

Fig. 6: Qualitative results of relighting on real-world scenes.

3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - Issues with shadows and improving rendering quality
 - **Temporal 3DGS**
 - Learning based approaches
 - Attribute extension

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering, CVPR 2024

4D Gaussian Splatting for Real-Time Dynamic Scene Rendering

Guanjun Wu^{1*}, Taoran Yi^{2*}, Jiemin Fang^{3†}, Lingxi Xie³, Xiaopeng Zhang³,
Wei Wei¹, Wenyu Liu², Qi Tian³, Xinggang Wang^{2†}

¹School of CS, Huazhong University of Science and Technology

²School of EIC, Huazhong University of Science and Technology ³Huawei Inc.

{guajuwu, taoranyi, weiw, liuwu, xgwang}@hust.edu.cn

{jaminfong, 198808xc, zxphistory}@gmail.com tian.qi1@huawei.com

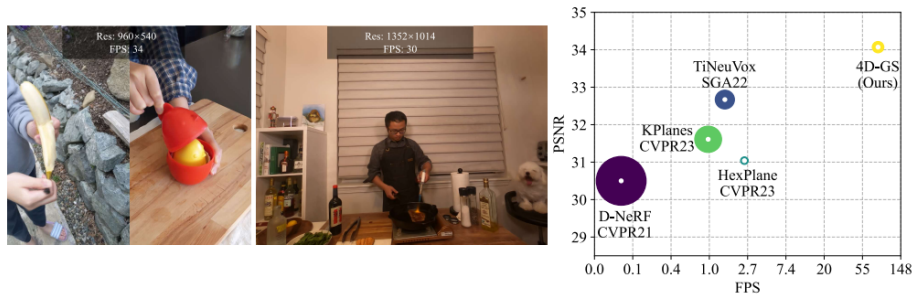


Figure 1. Our method achieves real-time rendering[†] for dynamic scenes at high image resolutions while maintaining high rendering quality. The right figure is tested on synthetic datasets, where the radius of the dot corresponds to the training time. “Res”: resolution.

[†]The rendering speed not only depends on the image resolution but also the number of 3D Gaussians and the scale of deformation fields which are determined by the complexity of the scene.

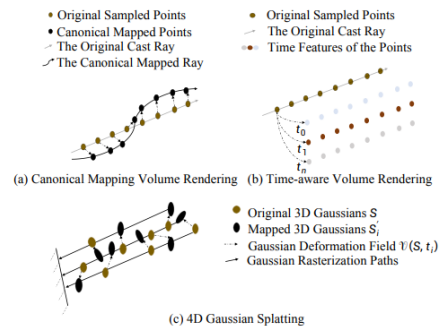


Figure 2. Illustration of different dynamic scene rendering methods. (a) Points are sampled in the casted ray during volume rendering. The point deformation fields proposed in [6, 28] map the points into a canonical space. (b) Time-aware volume rendering computes the features of each point directly and does not change the rendering path. (c) The Gaussian deformation field converts original 3D Gaussians into another group of 3D Gaussians with a certain timestamp.

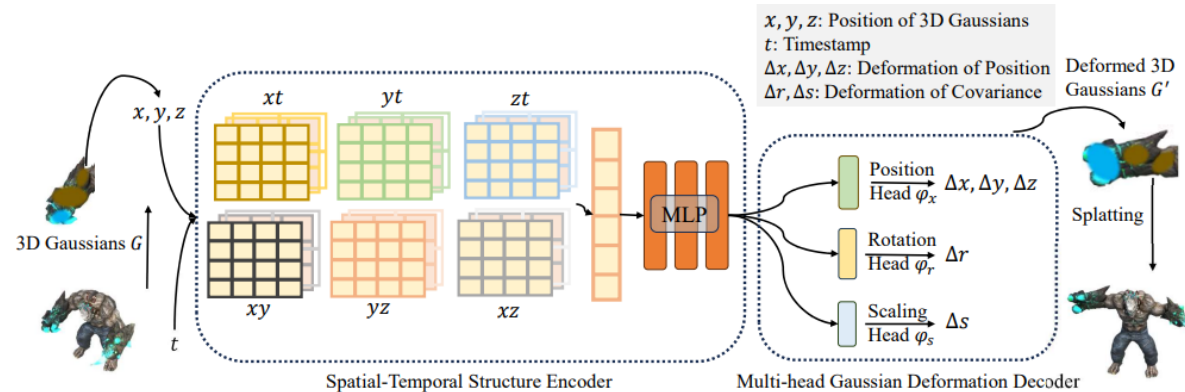


Figure 3. The overall pipeline of our model. Given a group of 3D Gaussians \mathcal{G} , we extract the center coordinate of each 3D Gaussian \mathcal{X} and timestamp t to compute the voxel feature by querying multi-resolution voxel planes. Then a tiny multi-head Gaussian deformation decoder is used to decode the feature and get the deformed 3D Gaussians \mathcal{G}' at timestamp t . The deformed Gaussians are then splatted to the rendered image.

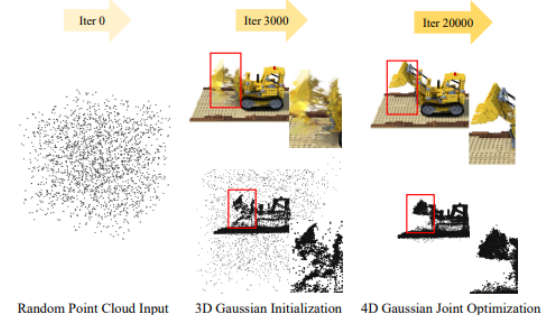


Figure 4. Illustration of the optimization process. With static 3D Gaussian initialization, our model can learn high-quality 3D Gaussians of the motion part.

3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - Issues with shadows and improving rendering quality
 - Temporal 3DGS
 - **Learning based approaches**
 - **Learning Gaussian blobs**
 - Attribute extension
 - 3D Large language field, and the connection to large language models (LLMs), VLMs (e.g. SAM)

L3DG: Latent 3D Gaussian Diffusion, Siggraph Asia 2024

L3DG: Latent 3D Gaussian Diffusion

BARBARA ROESSLE, Technical University of Munich, Germany
NORMAN MÜLLER, Meta Reality Labs Zurich, Switzerland
LORENZO PORZI, Meta Reality Labs Zurich, Switzerland
SAMUEL ROTA BULÓ, Meta Reality Labs Zurich, Switzerland
PETER KONTSCHIEDER, Meta Reality Labs Zurich, Switzerland
ANGELA DAI, Technical University of Munich, Germany
MATTHIAS NIESSNER, Technical University of Munich, Germany



Fig. 1. L3DG learns a compressed latent space of 3D Gaussian representations and efficiently synthesizes novel scenes via diffusion in latent space. This approach makes L3DG scalable to room-size scenes, which are generated from pure noise leading to geometrically realistic scenes of 3D Gaussians that can be rendered in real-time. Above results are from our model trained on 3D-FRONT; we visualize the 3D Gaussian ellipsoids and show renderings.

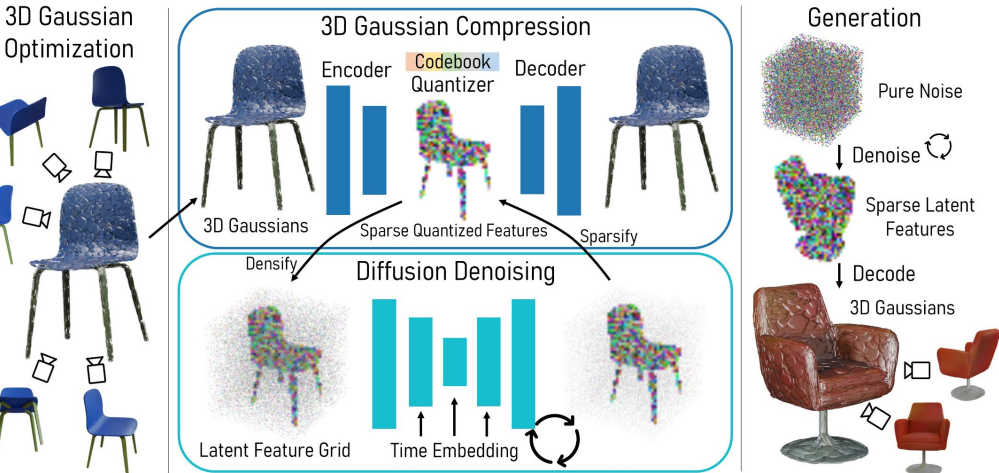


Fig. 2. L3DG method overview: our 3D Gaussian compression model learns to compress 3D Gaussians into sparse quantized features using sparse convolutions and vector-quantization at the bottleneck (VQ-VAE). This allows our 3D diffusion model to efficiently operate on the compressed latent space. At test time, novel scenes are generated by denoising in latent space, which can be sparsified and decoded to high quality 3D Gaussians.



Fig. 3. Comparison on Poshops [23] et al. [2019]. Our method generates more detail than the baselines, such as thin structures, and fine-texture artifacts.



GAUSSIAN MASKED AUTOENCODERS

Jathushan Rajasegaran^{1,2}, Xinlei Chen¹, Rulilong Li²,
Christoph Feichtenhofer¹, Jitendra Malik^{1,2}, Shiry Ginosar³
Meta, FAIR¹, UC Berkeley², Toyota Technological Institute at Chicago³

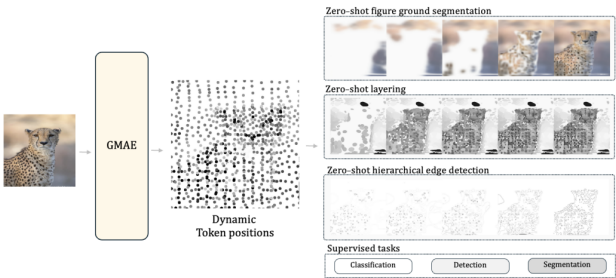


Figure 1: **Gaussian Masked Autoencoders (GMAE)** maintains high performance in supervised representation learning tasks such as classification, detection, and segmentation, but more importantly enables zero-shot capabilities. GMAE introduces a learned mid-level intermediate representation of 3D Gaussians that we train using pixel-based image reconstruction losses rather than direct supervision by rendering the Gaussians into pixel space. Through this reconstruction loss, the Gaussian collection learns to distribute non-uniformly in space and scale, dynamically following the input image’s information density and high-frequency details. Having the degree of freedom in depth allows the model to learn the layering of objects and scenes, which enables figure-ground separation, layering, and edge detection without any training.

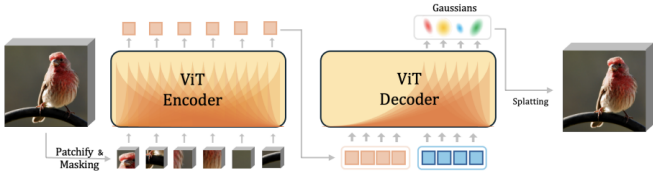


Figure 2: **Masked Autoencoding via Gaussian Splatting:** The ViT Encoder processes masked input image patches to produce **latent embeddings**. The ViT Decoder then predicts explicit Gaussian parameters based on **query tokens**, including color, opacity, center, scale, and orientation. These Gaussians are then rendered via differentiable volume splatting (Kerbl et al., 2023) to reconstruct the original image. We pre-train our models fully end-to-end with self-supervision.

MaskGaussian: Adaptive 3D Gaussian Representation from Probabilistic Masks

Yifei Liu^{1,2} Zhihang Zhong^{1,†} Yifan Zhan^{1,3} Sheng Xu² Xiao Sun^{1,†}

¹Shanghai AI Laboratory ²Beihang University ³The University of Tokyo

Key idea: 3D Gaussian points are probabilistic entities.

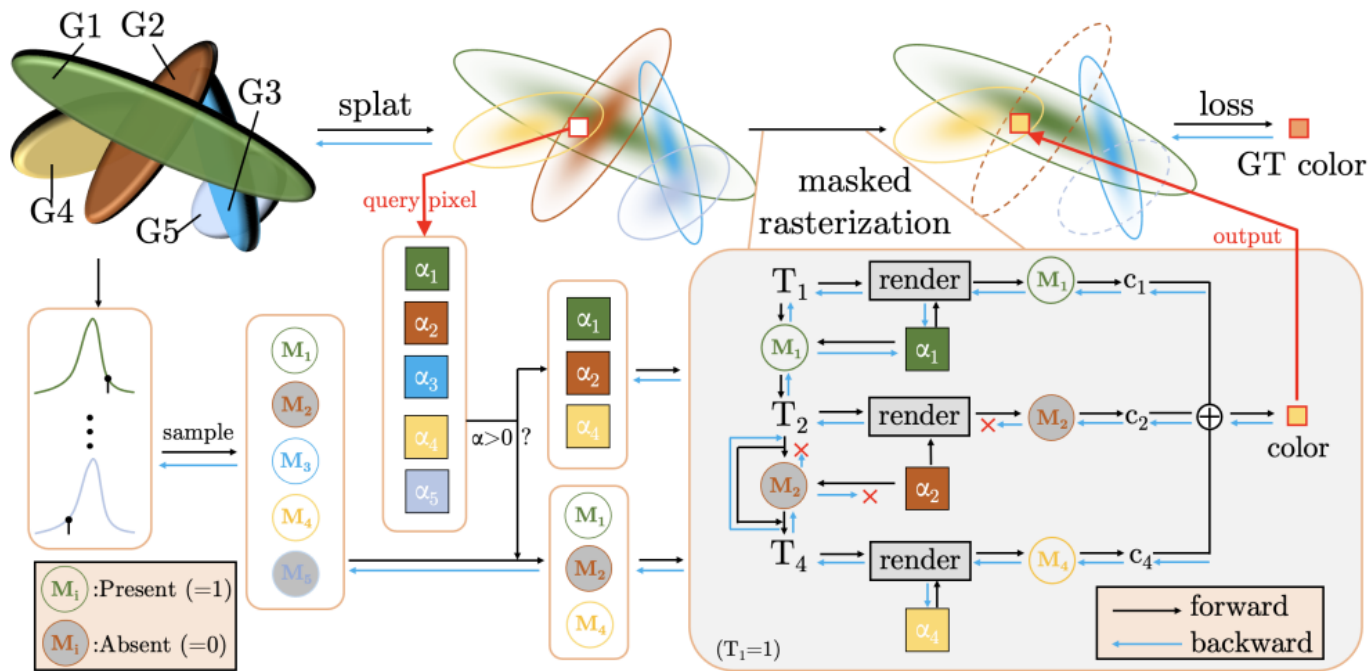


Figure 1. **Overview of MaskGaussian.** We illustrate our pipeline with five Gaussians, G_1 through G_5 , where G_2 and G_5 are not sampled and masked. First, all Gaussians are splatted in the standard manner, and differentiable masks are sampled from their existence distributions. For each query pixel, a splat G_i has α_i computed from normal attributes (center, scale, rotation). Splats with zero α_i are filtered out, and the remaining splats and their masks are passed into the masked-rasterization. We apply the masks in two places: the transmittance evolution for T_i and the color rendering for c_i , as detailed in Eq 3 and Eq. 4. A masked splat G_i (e.g., $i=2$ in this figure) does not receive a gradient for α_i , and thus does not update its normal attributes, but it receives a gradient for mask m_i and updates its existence probability.

GaussTR: Foundation Model-Aligned Gaussian Transformer for Self-Supervised 3D Spatial Understanding

Haoyi Jiang^{1*} Liu Liu² Tianheng Cheng¹ Xinjie Wang² Tianwei Lin²

Zhizhong Su² Wenyu Liu¹ Xinggang Wang^{1†}

¹Huazhong University of Science & Technology ²Horizon Robotics

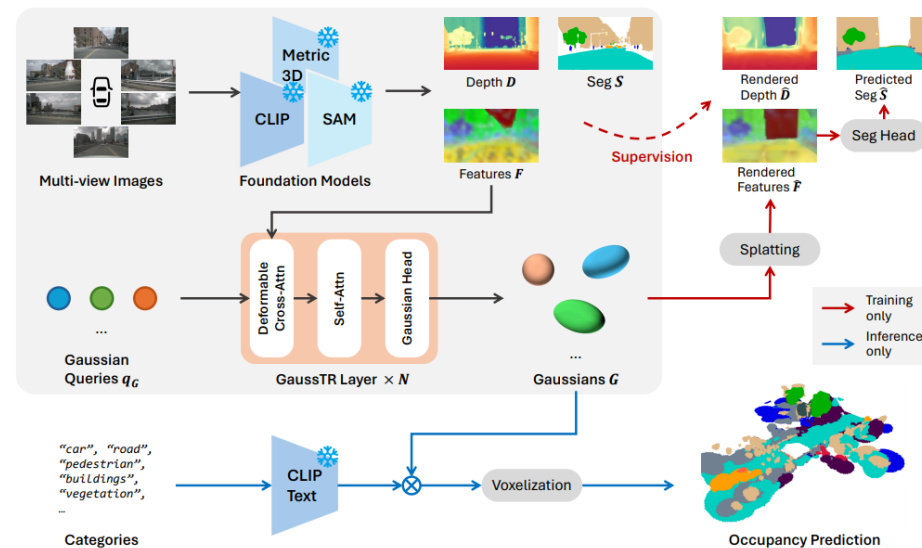


Figure 2. **Architectural overview of the GaussTR framework.** The GaussTR framework initiates with extracting features and depth estimation using various foundation models, including CLIP [10, 37] and Metric3D [19]. Subsequently, GaussTR predicts a sparse set of Gaussian queries to represent the scene through a series of Transformer layers. During the training phase, the predicted Gaussians are splatted to source views and aligned with original 2D features for supervision. For inference, the Gaussians are converted into logits by measuring their similarity with category vectors, followed by voxelization to produce the final volumetric prediction.

ZERO-1-TO-G: TAMING PRETRAINED 2D DIFFUSION MODEL FOR DIRECT 3D GENERATION

Xuyi Meng¹, Chen Wang¹, Jiahui Lei¹, Kostas Daniilidis¹, Jiatao Gu², Lingjie Liu¹

¹University of Pennsylvania ²Apple

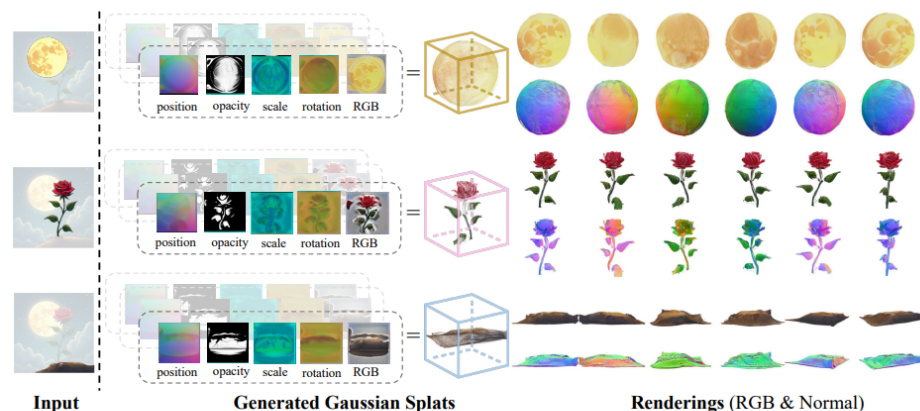


Figure 1: **Zero-1-to-G** tackles direct Gaussian splat generation from single images. By using pretrained 2D diffusion models, we are able to generalize to in-the-wild objects.

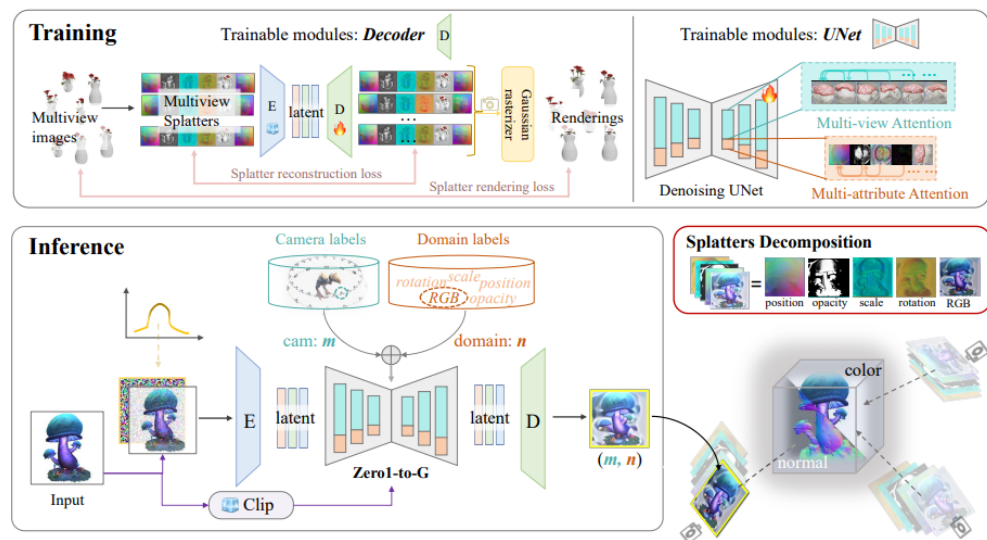


Figure 2: The pipeline of Zero-1-to-G. During training, we fine-tune both the VAE decoder and the denoising UNet of Stable Diffusion. Decoder fine-tuning is required for high-quality splatter image rendering because the renderings of splatter images are sensitive to pixel value changes; the attention mechanism of denoising UNet is extended to model the multi-view and multi-attribute correlation. At inference time, given a single view input of the target object, each component in the splatter image is generated by conditioning the camera view and attribute switcher. The generated set of splatter image components can be directly composed into Gaussian splats. Note that here we only show 3 views of splatter images for better visualization, but we use 6 views in our experiment.

Structured 3D Latents for Scalable and Versatile 3D Generation

Jianfeng Xiang^{1,3*} Zelong Lv^{2,3*} Sicheng Xu³ Yu Deng³ Ruicheng Wang^{2,3*}
Bowen Zhang^{2,3*} Dong Chen³ Xin Tong³ Jiaolong Yang^{3†}
¹Tsinghua University ²USTC ³Microsoft Research
<https://trellis3d.github.io>

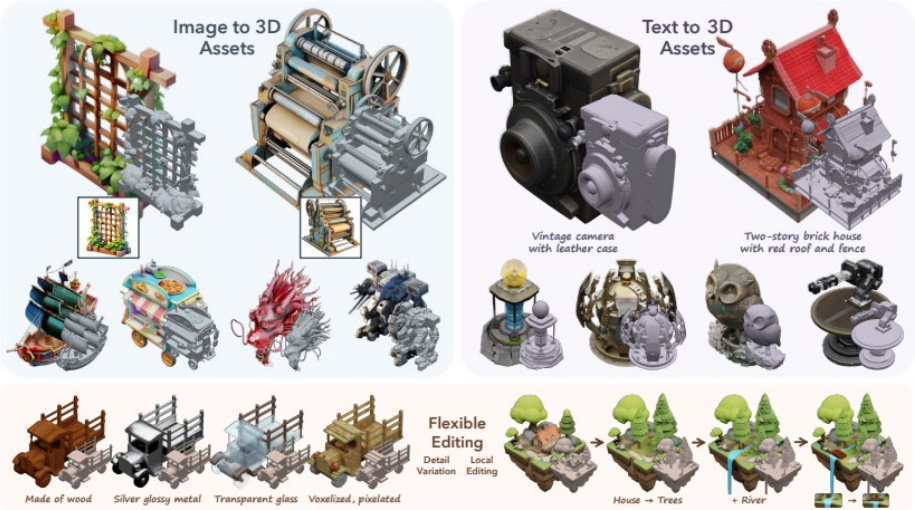


Figure 1. High-quality 3D assets generated by our method in various formats from text or image prompts (using GPT-4o and DALL-E 3). Our method enables versatile generation in about 10 seconds, offering vivid appearances with 3D Gaussians or Radiance Fields and detailed geometries with meshes. It also supports flexible 3D editing. *Best viewed with zoom-in.*

3D Assets Encoding & Decoding

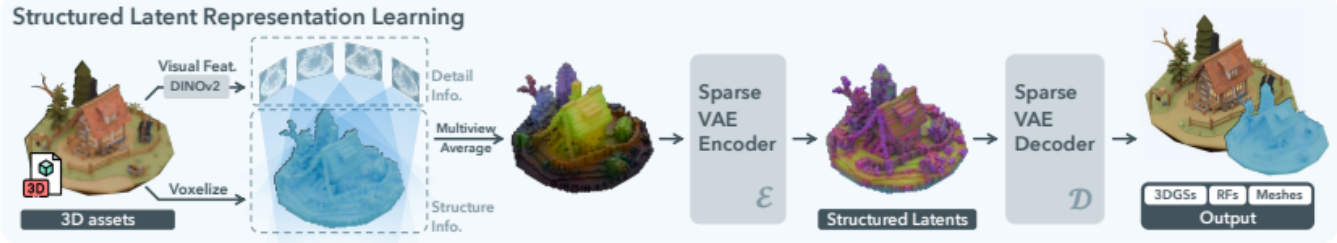


Figure 2. Overview of our method. **Encoding & Decoding:** We adopt a structured latent representation (SLAT) for 3D assets encoding, which defines local latents on a sparse 3D grid to represent both geometry and appearance information. It is encoded from the 3D assets by fusing and processing dense multiview visual features extracted from a DINOv2 encoder, and can be decoded into versatile output representations with different decoders. **Generation:** Two specialized rectified flow transformers are utilized to generate SLAT, one for the sparse structure and the other for local latents attached to it.

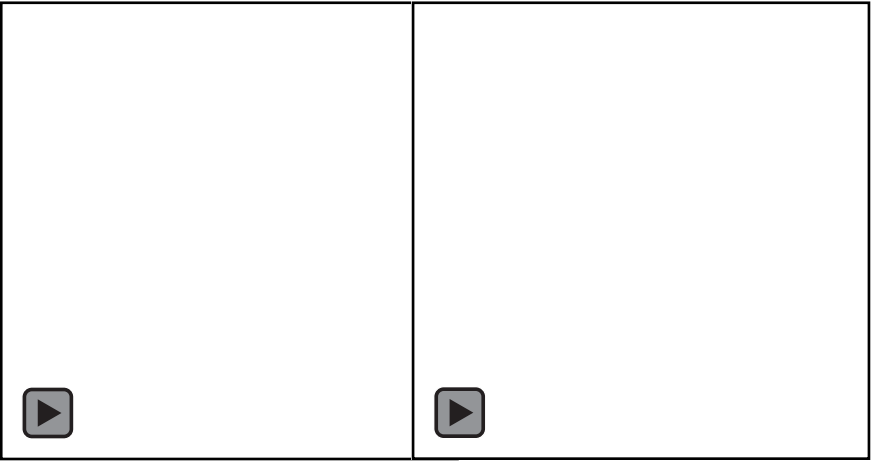


Figure 20. A vibrant streetview constructed with assets generated by TRELLIS. (Text and image prompts are linked with yellow lines)

3D Gaussian Splatting (3DGS)

- What is 3DGS
- Directions for improvement
 - Faster and less memory consumption
 - Less bumpy surface representation
 - Better basis as a cloud of 3D points
 - Issues with shadows and improving rendering quality
 - Temporal 3DGS
 - Learning based approaches
 - **Attribute extension**
 - **3D Large language field, and the connection to large language models (LLMs), VLMs (e.g. SAM)**

LangSplat: 3D Language Gaussian Splatting, CVPR 2024

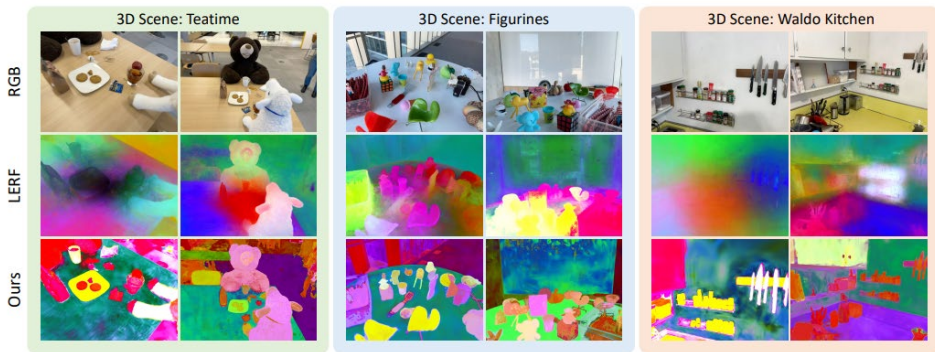
LangSplat: 3D Language Gaussian Splatting

Minghan Qin^{1,*}, Wanhua Li^{2,*}, Jiawei Zhou^{1,*}, Haoqian Wang¹, Hanspeter Pfister²

¹Tsinghua University ²Harvard University

qmh21@mails.tsinghua.edu.cn, wanhua@seas.harvard.edu, zhoujw22@mails.tsinghua.edu.cn

wanghaoqian@tsinghua.edu.cn, pfister@seas.harvard.edu



Rendered RGB Video



Visualization of Learned Language Feature¹

¹Different colors represent different language features.

Figure 1. Visualization of learned 3D language features of the previous SOTA method LERF and our LangSplat. While LERF generates imprecise and vague 3D features, our LangSplat accurately captures object boundaries and provides precise 3D language fields. While being effective, our LangSplat is also $199\times$ faster than LERF at the resolution of 1440×1080 .

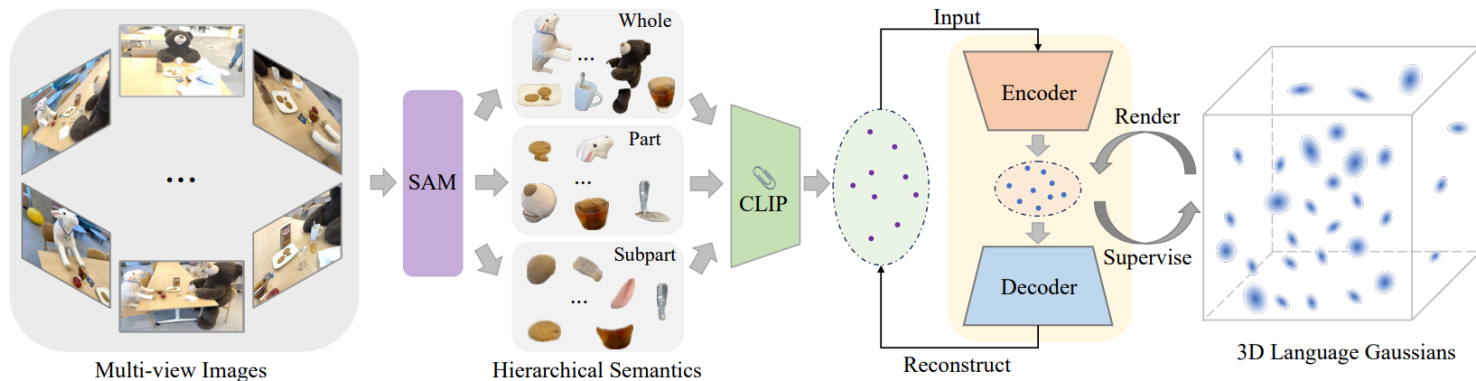


Figure 2. The framework of our LangSplat. Our LangSplat leverages SAM to learn hierarchical semantics to address the point ambiguity issue. Then segment masks are sent to the CLIP image encoder to extract the corresponding CLIP embeddings. We learn an autoencoder with these obtained CLIP embeddings. Our 3D language Gaussian learn language features on the scene-specific latent space to reduce the memory cost. During querying, the rendered language embeddings are sent to the decoder to recover the features on the CLIP space.

Topics

- The historic context of 3DGS
- Present: [3DGS](#) and its recent progress
- **Future discussions**

Some Thoughts on the future of (AI-assisted) Rendering

- Is 3DGS the final/winning rendering pipeline?
- How AI/deep learning will contribute in rendering?
- What will be the next “stable” GPU?
 - Polygonal mesh based 3D surface representation and its rendering pipeline
 - PBR, i.e. ray tracing
 - AI compute: convolution, transformer, gradient-based optimization