

Surface Reconstruction and 3D Gaussian Splatting

Overview of the research

Antoine Guédon

Imagine Team

LIGM, Ecole des Ponts (ENPC), Univ Gustave Eiffel, CNRS, France



École des Ponts
ParisTech



LABORATOIRE
D'INFORMATIQUE
GASPARD MONGE

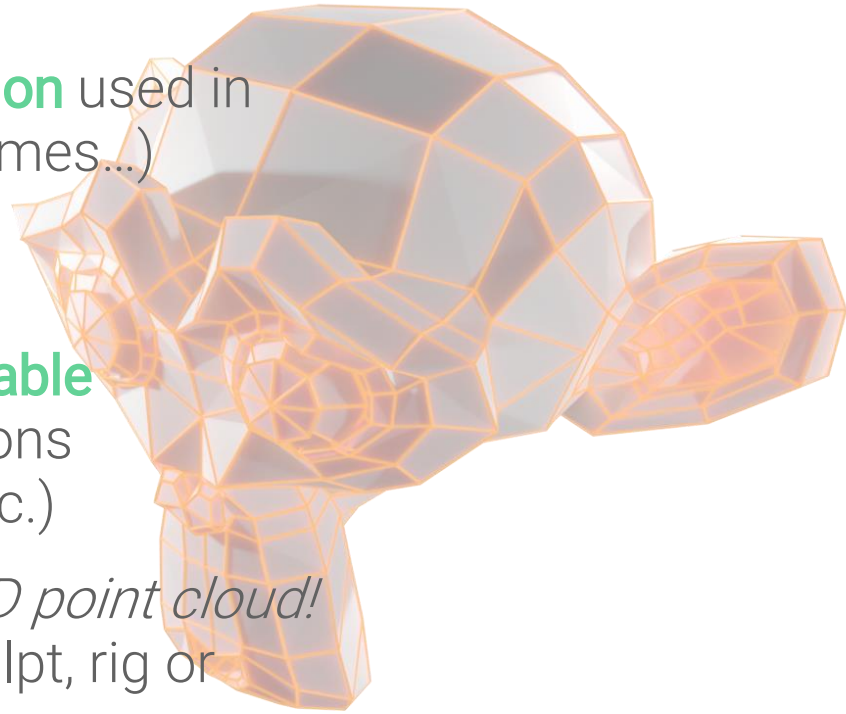
Introduction

Meshes

The **mesh** is still the **default representation** used in computer graphics (animation, video games...) because:

- It is memory efficient
- It is easily **deformable/editable/riggable**
- It allows for **physics-based** interactions (**collisions, relighting** on surfaces, etc.)

But 3DGS already provides an explicit 3D point cloud!
Trust me, it's a nightmare to directly sculpt, rig or animate a point cloud. I tried.



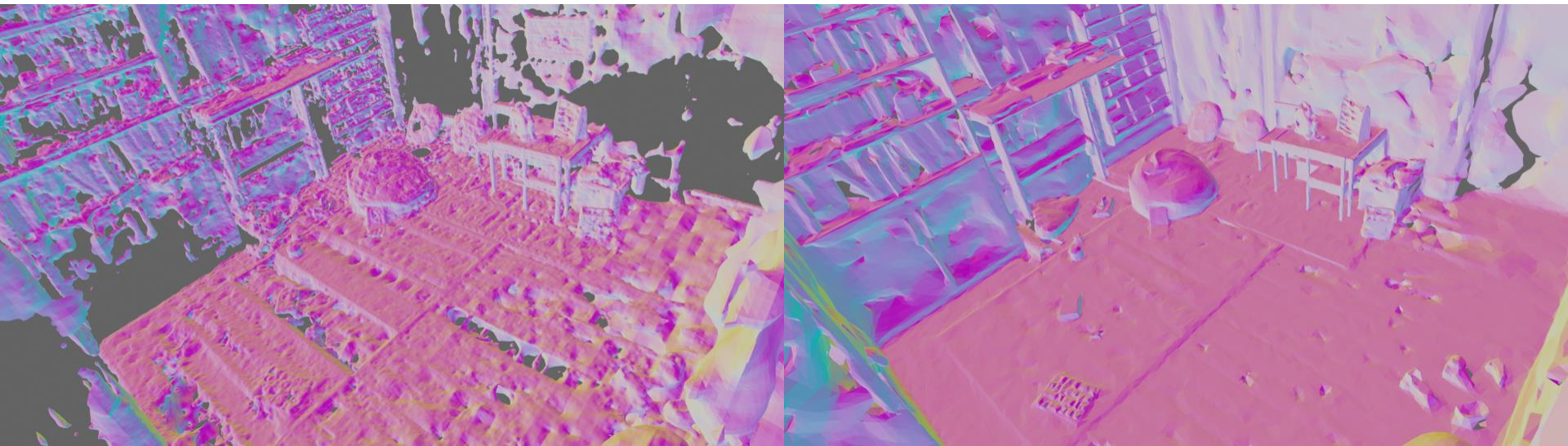
Meshes from Radiance Fields

- **NeuS**, and **Instant-NeuS**
- **BakedSDF** (Yariv., Barron et al., SIGGRAPH 2023)
- **Neuralangelo** (CVPR2023) is also a good example, and adapts InstantNGP to the SDF framework
- **Binary Opacity Grids** (arXiv 2024)

But still, NeRF-based meshing approaches can be very slow (several GPU-hours, even days) or inaccurate.

Questions

1. Can we leverage Gaussian Splatting to reconstruct triangles meshes?
2. Can we use a mesh as an underlying structure for a Gaussian Splatting representation (for easier editing, animation, *etc.*)?

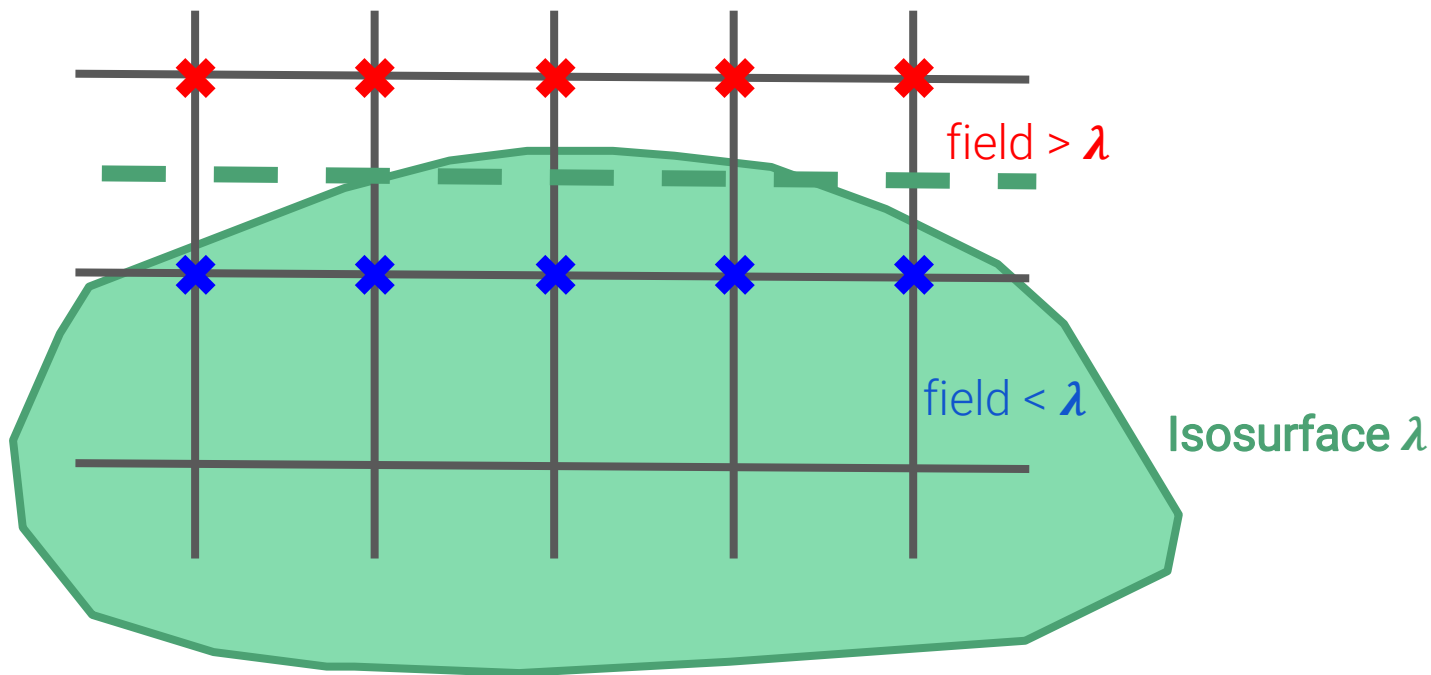


Instant-NeuS

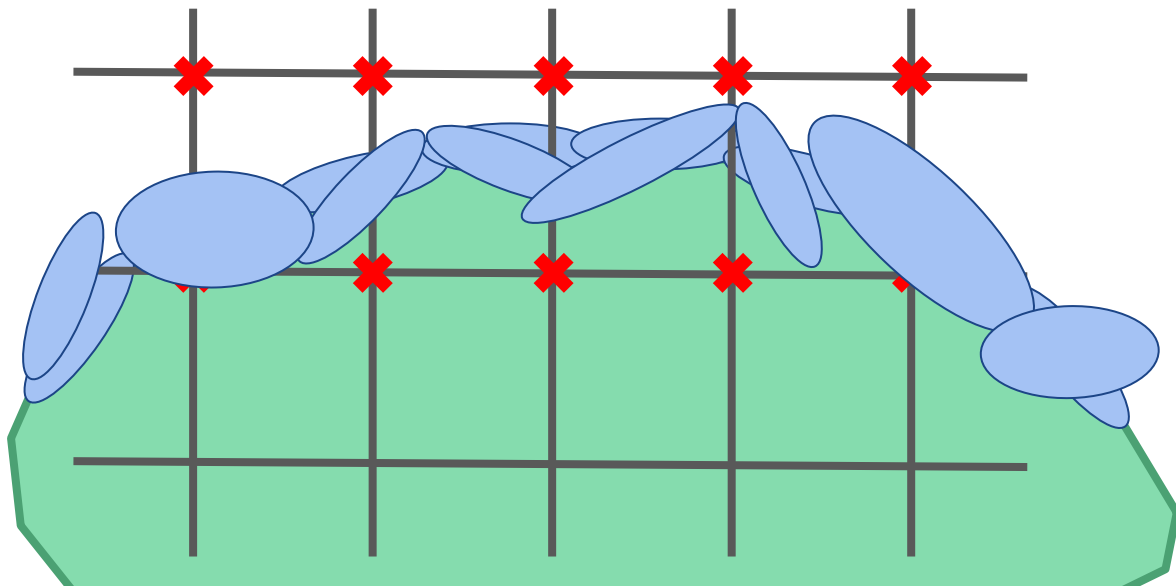
SuGaR

Radiance Field to Mesh: Standard Practice

Standard Practice: Marching Algorithm on Implicit Field

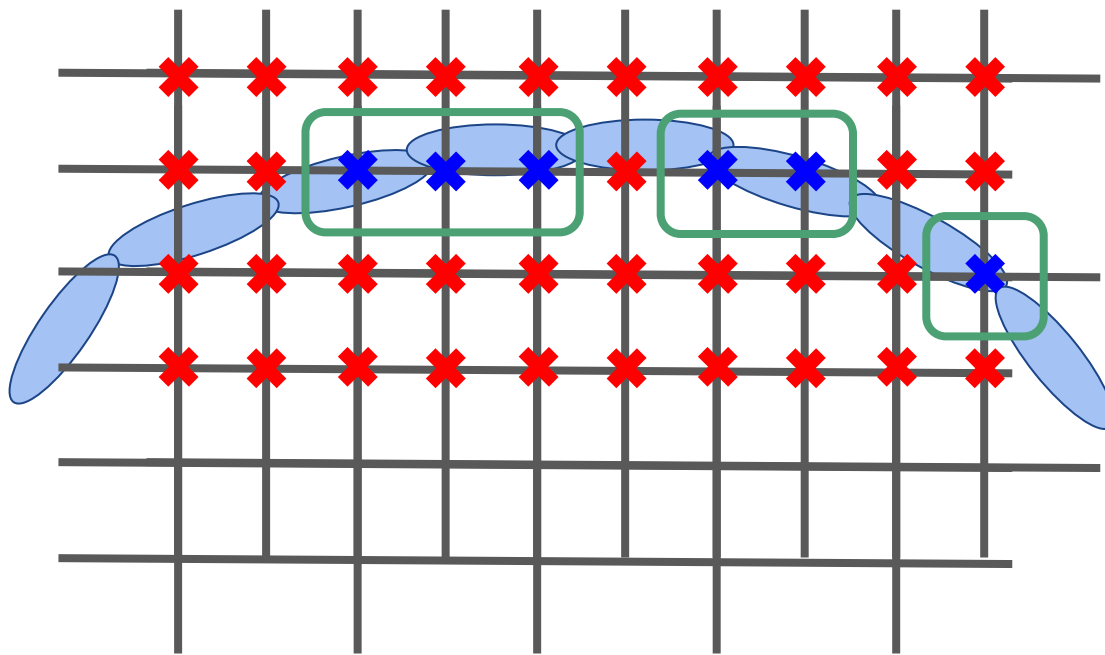


Standard Practice: Marching Algorithm on Implicit Field

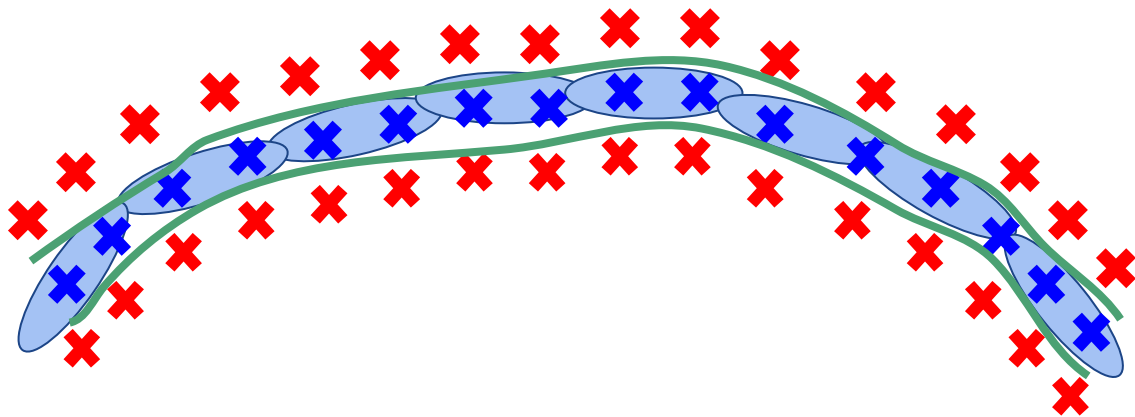


$$d(p) = \sum_g \alpha_g \exp \left(-\frac{1}{2} (p - \mu_g)^T \Sigma_g^{-1} (p - \mu_g) \right)$$

Standard Practice: Marching Algorithm on Implicit Field



Standard Practice: Marching Algorithm on Implicit Field



From Gaussians to mesh: Quick Overview

Marching Cubes and 3DGS: DreamGaussian

DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation,
Jiaxiang Tang *et al.*

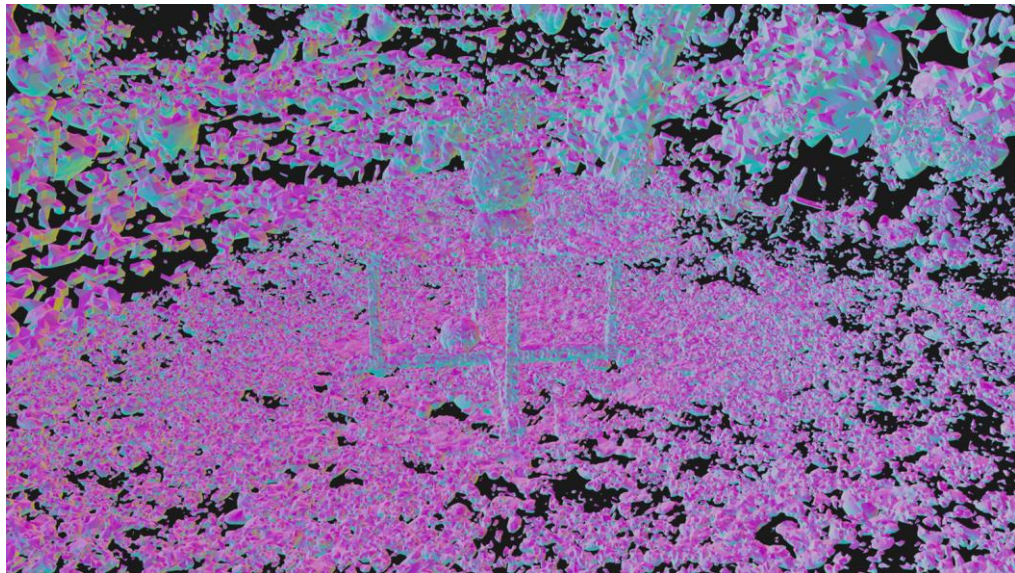
The paper adapts **Zero-123** from NeRF to 3D Gaussian Splatting: The goal is to generate 3D content from a single image. Authors use a marching algorithm.



Marching Cubes does not work in practice, in real scenes

Only **works with a few thousands** Gaussians (5k).

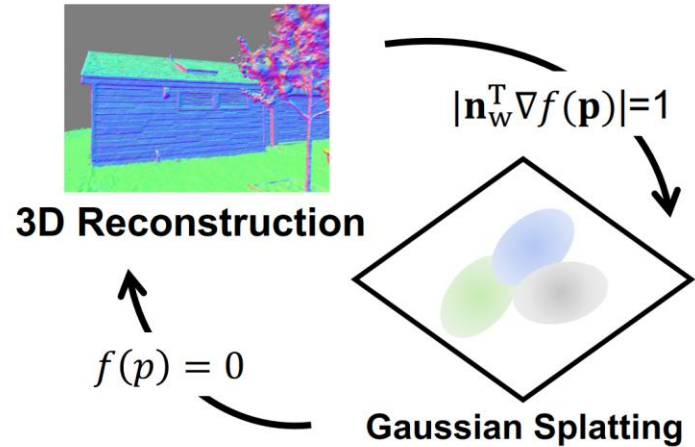
In practice, 3DGS produces **millions** of Gaussians!



NeRFs augmented with 3D Gaussians

NeuSG: Neural Implicit Surface Reconstruction with 3D Gaussian Splatting Guidance,
Chen *et al.*, 2023

- Jointly optimizing NeuS and 3DGS
- 3DGS as an additional regularization tool for NeuS
- Conceptually, very different
- Less “light-weight” than just using 3DGS (optimization takes 16 hours)



SuGaR:

Surface-Aligned **Ga**ussian Splatting for Efficient 3D Mesh
Reconstruction and High-Quality Mesh Rendering

To appear at CVPR 2024

Antoine Guédon and Vincent Lepetit

Imagine Team

LIGM, Ecole des Ponts (ENPC), Univ Gustave Eiffel, CNRS, France



École des Ponts
ParisTech



LABORATOIRE
D'INFORMATIQUE
GASPARD MONGE

SuGaR: Three main contributions

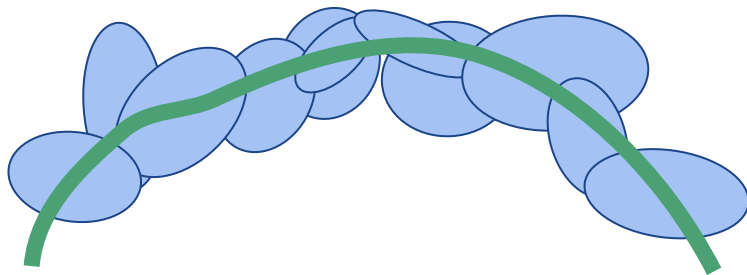
1. A **regularization term** that encourages 3D Gaussians to **align with the surface**
2. A **scalable mesh extraction** method **tailored** for 3D Gaussians
3. A **refinement method** that **binds** new 3D Gaussians to the triangles of the mesh, resulting in a **hybrid representation**

Aligning Gaussians with the surface

Aligning Gaussians with the true surface: *Density constraint*

Question: What would the density function look like, in an ideal scenario where Gaussians are well-aligned with the surface?

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

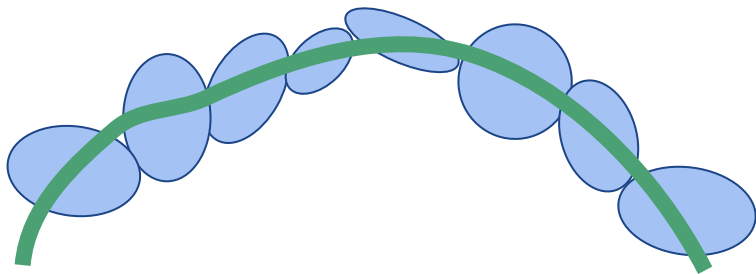


Aligning Gaussians with the true surface: *Density constraint*

Question: What would the density function look like, in an ideal scenario where Gaussians are well-aligned on the surface?

#1: Gaussians should have limited overlap, and be well-spread on the surface.

$$d(p) = \alpha_{g^*} \exp \left(-\frac{1}{2} (p - \mu_{g^*})^T \Sigma_{g^*}^{-1} (p - \mu_{g^*}) \right)$$

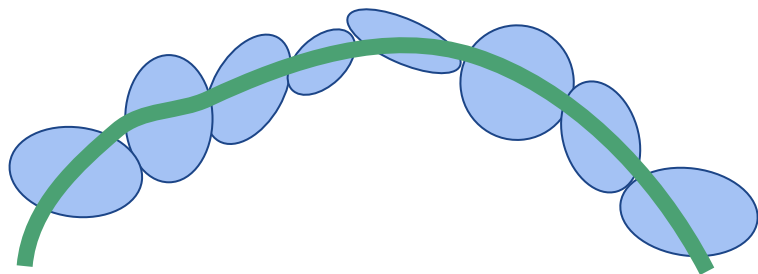


Aligning Gaussians with the true surface: *Density constraint*

Question: What would the density function look like, in an ideal scenario where Gaussians are well-aligned on the surface?

#2: Gaussians should be opaque or fully transparent (otherwise, isosurface are meaningless).

$$d(p) = \exp \left(-\frac{1}{2} (p - \mu_{g^*})^T \Sigma_{g^*}^{-1} (p - \mu_{g^*}) \right)$$



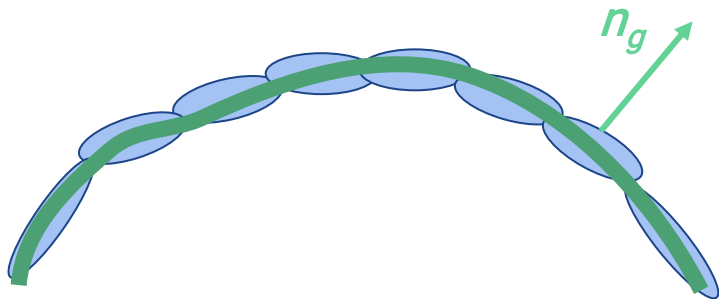
Aligning Gaussians with the true surface: *Density constraint*

Question: What would the density function look like, in an ideal scenario where Gaussians are well-aligned on the surface?

#3: Gaussians should be as flat as possible (one of the three scaling factors should be close to zero).

$$\bar{d}(p) = \exp\left(-\frac{1}{2s_g^2} \langle p - \mu_{g^*}, n_{g^*} \rangle^2\right)$$

because $(p - \mu_g)^T \Sigma_g^{-1} (p - \mu_g) \approx \frac{1}{s_g^2} \langle p - \mu_g, n_g \rangle^2$



Aligning Gaussians with the true surface: *Density constraint*

“Ideal” density:

$$\bar{d}(p) = \exp\left(-\frac{1}{2s_{g^*}^2} \langle p - \mu_{g^*}, n_{g^*} \rangle^2\right)$$

Regularization term:

$$\mathcal{R} = |d(p) - \bar{d}(p)|$$

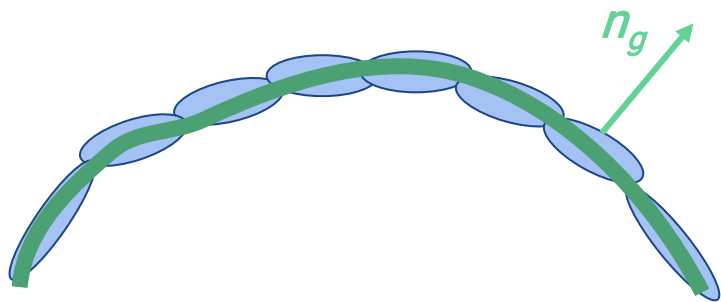
where p are points sampled near the centers of the Gaussians, following the Gaussian distributions.

Goal: Encourage the density to converge toward the ideal scenario.
Enforces Gaussians to align with the surface in a non destructive way.

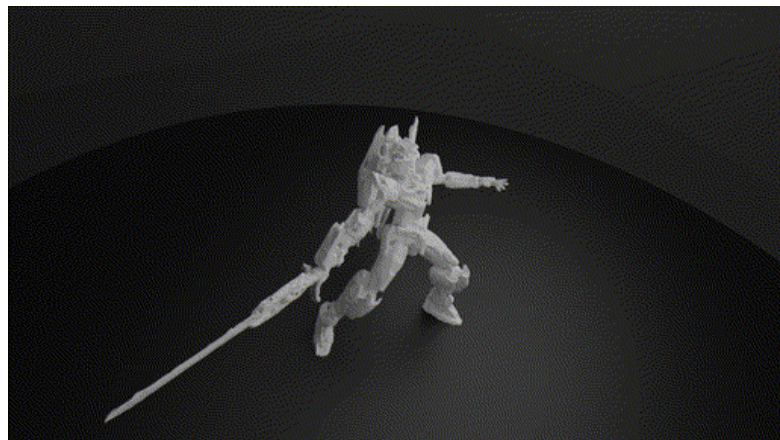
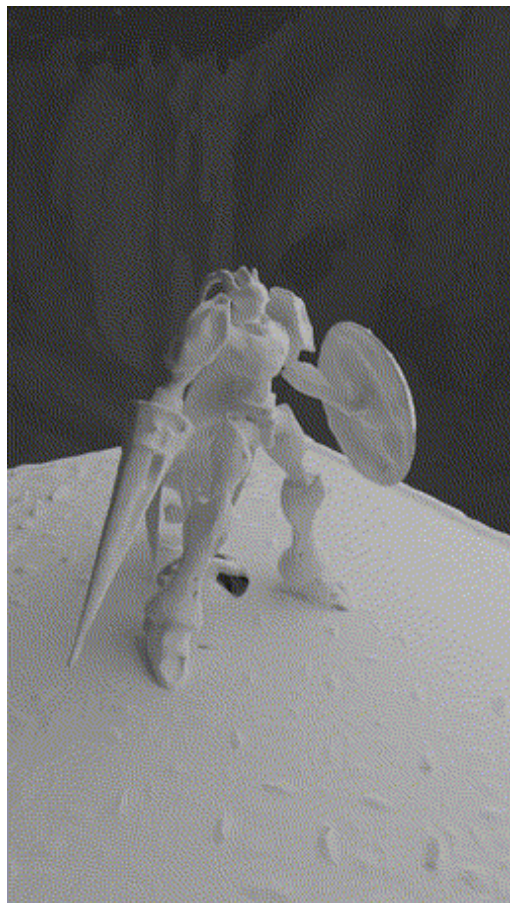
Aligning Gaussians with the true surface: *Density constraint*

Additional regularization term for faster alignment:

$$\mathcal{R}_{\text{Norm}} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left\| \frac{\nabla d(p)}{\|\nabla d(p)\|_2} - n_{g^*} \right\|_2^2$$

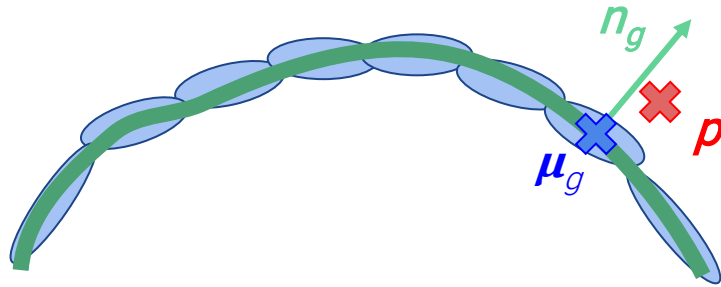


The density loss works pretty well in practice (foreground)



For even more regularization: *SDF constraint*

Ideal density: $\bar{d}(p) = \exp\left(-\frac{1}{2s_{g^*}^2} \langle p - \mu_{g^*}, n_{g^*} \rangle^2\right)$

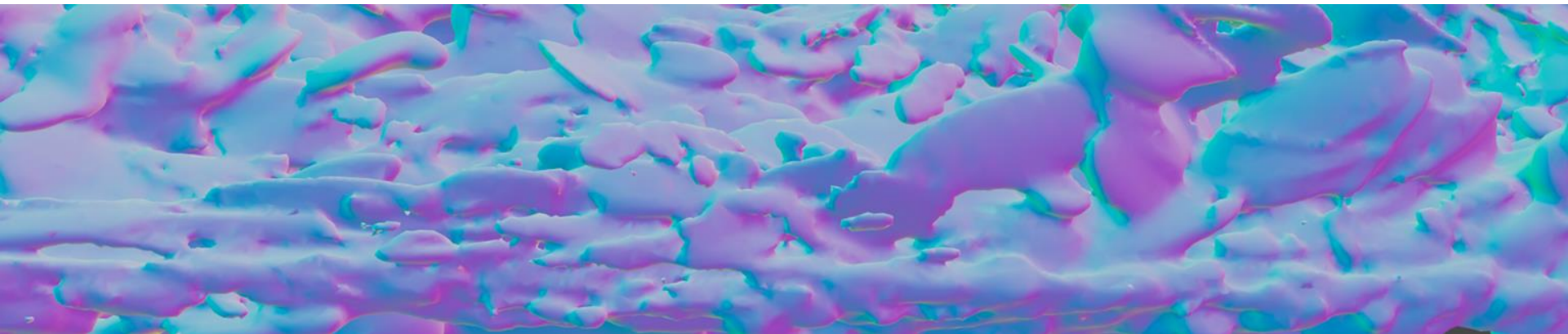


Distance between p and the surface: $|\langle p - \mu_{g^*}, n_{g^*} \rangle|$

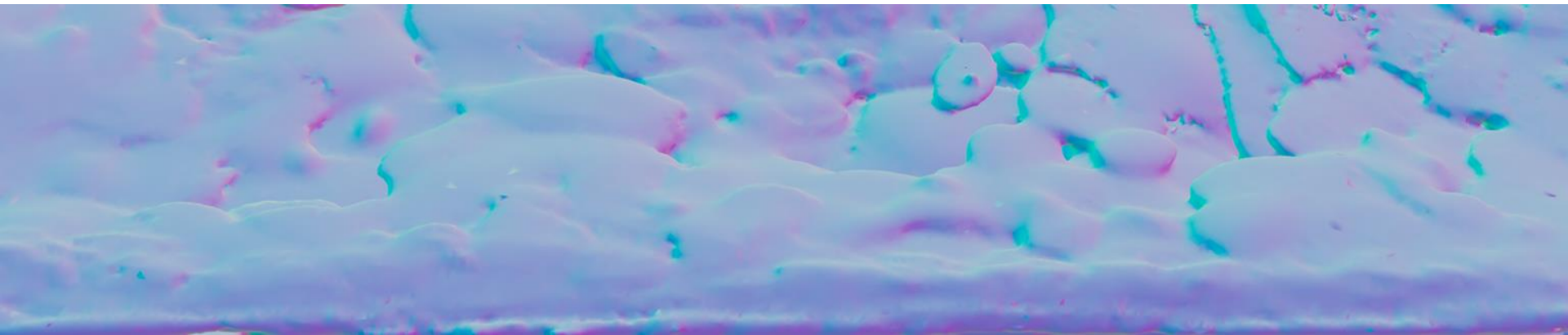
Conclusion: We introduce an “*ideal*” SDF

$$f(p) = \pm s_{g^*} \sqrt{-2 \log(d(p))}$$

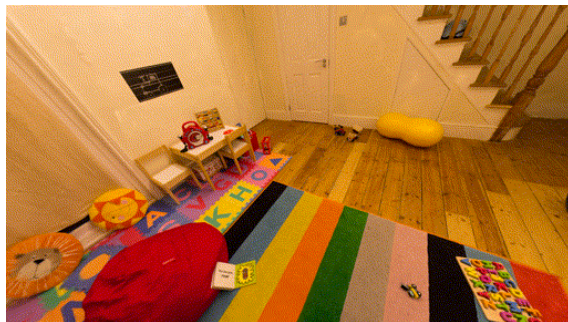
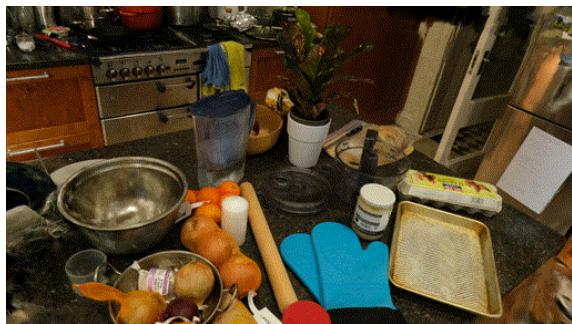
Alignment: Example of a plane surface (No regularization)



Alignment: Example of a plane surface (With regularization)

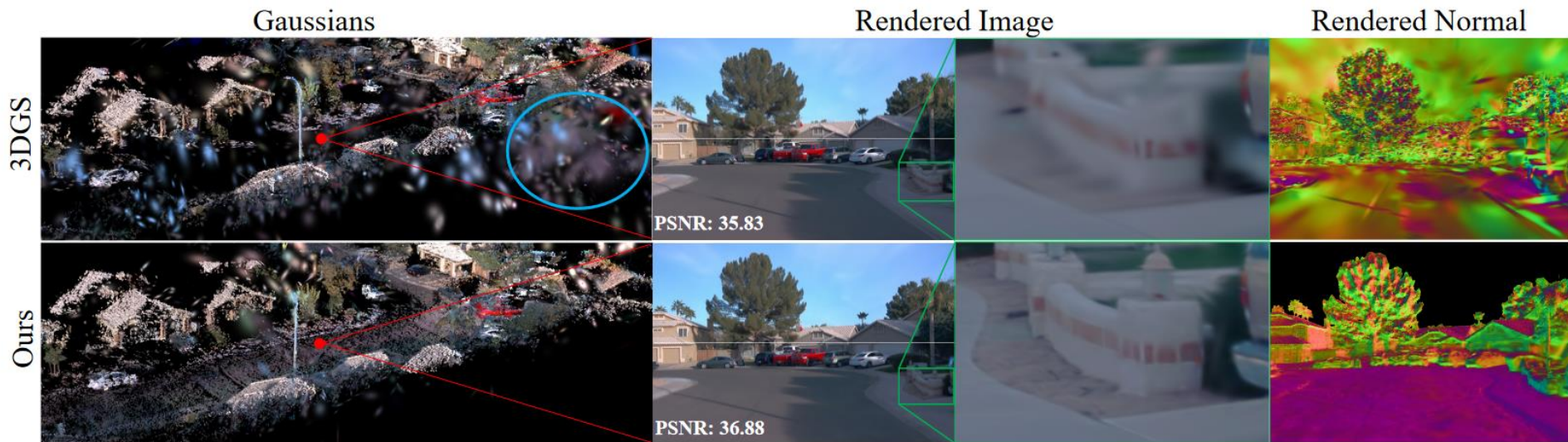


Alignment in scenes with background geometry



Other approaches to align Gaussians with the surface

GaussianPro: 3D Gaussian Splatting with Progressive Propagation, Cheng *et al.*



Mesh Extraction

Isosurface reconstruction

3DGS uses an **explicit 3D representation**. Why should we use a method tailored for implicit fields, like Marching Cubes?

Let's use a point cloud-based method!

Our choice: The **Poisson Surface Reconstruction**, which needs **3D points** sampled on the surface of the scene, as well as the corresponding **surface normals**.

Isosurface reconstruction as a Poisson problem

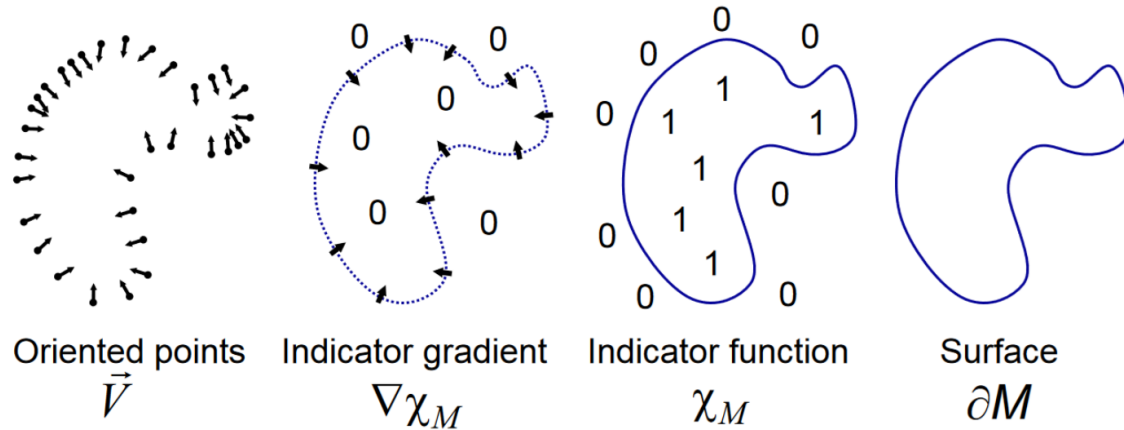


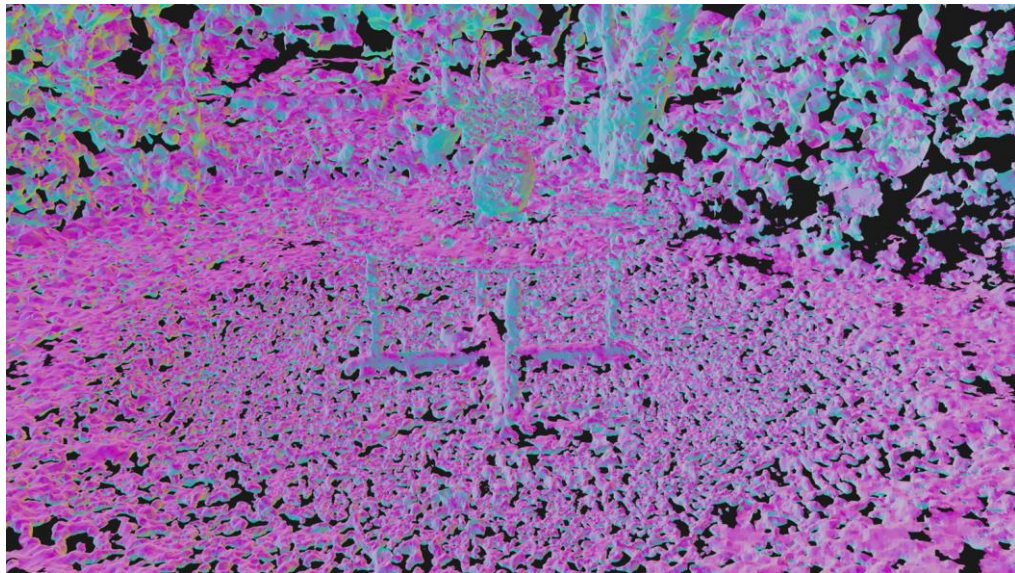
Figure 1: *Intuitive illustration of Poisson reconstruction in 2D.*

Conceptually, we are looking for **another implicit field χ** than the density d , that has approximately the same isosurface and normals, but with different boundary values:

The field is equal to 1 inside objects, and equal to 0 outside objects. Much better for marching!

Poisson reconstruction on Gaussian centers

The result is full of holes, because Gaussians can be far from their neighbors depending on their size!

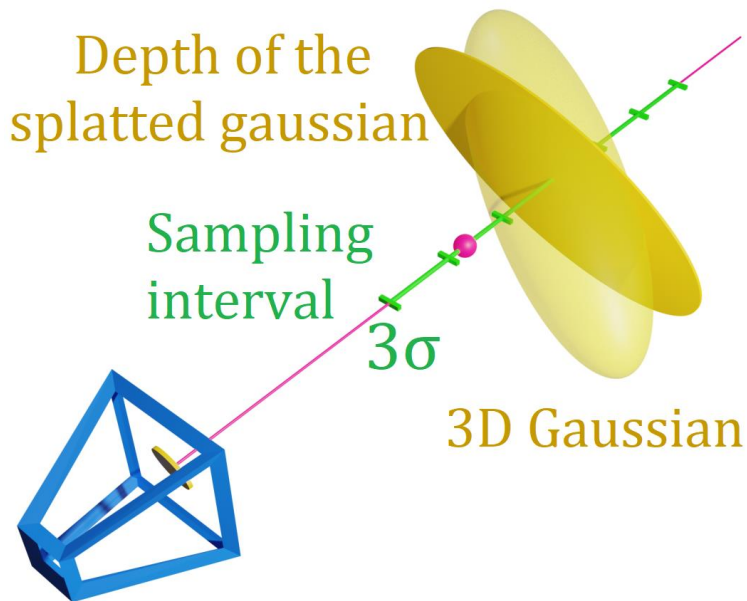


Sampling points on the surface

- Using the **centers of the Gaussians** does not work.
- Sampling **points from the Gaussian distributions** does not work, even if we target a specific isosurface (we end up with points located *“inside”* the objects).
- We need to sample points **only on the visible parts** of the isosurface of the density.

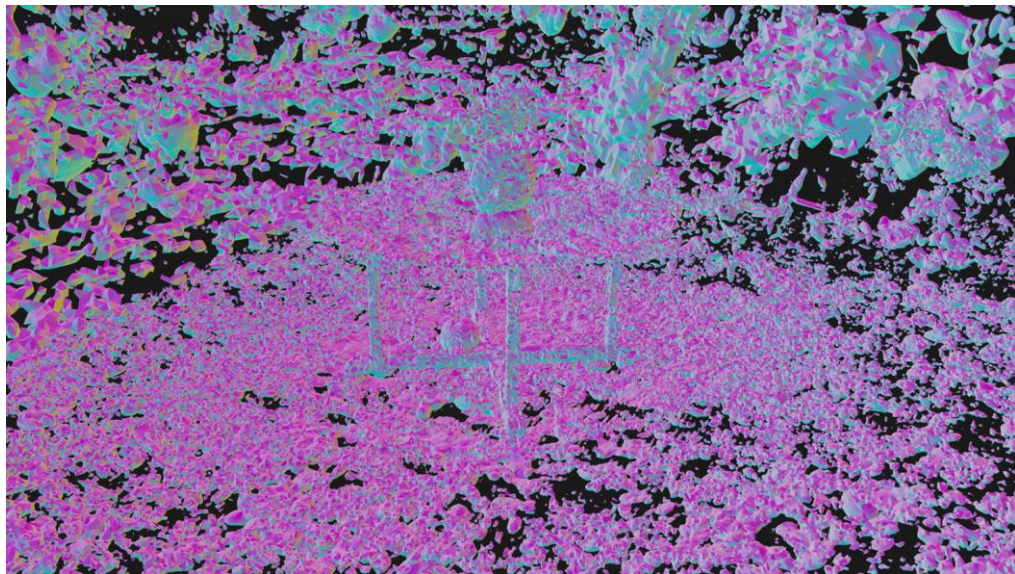
Sampling points on the surface

- We render depth maps from training viewpoints
- We backproject pixels into 3D points p using the depth map
- For a p sampled from a splatted Gaussian g , we look for the closest isosurface point in a range $3\sigma_g$ (confidence interval for the 99.7 level)
- We compute the normal at p as the normalized gradient of the density d



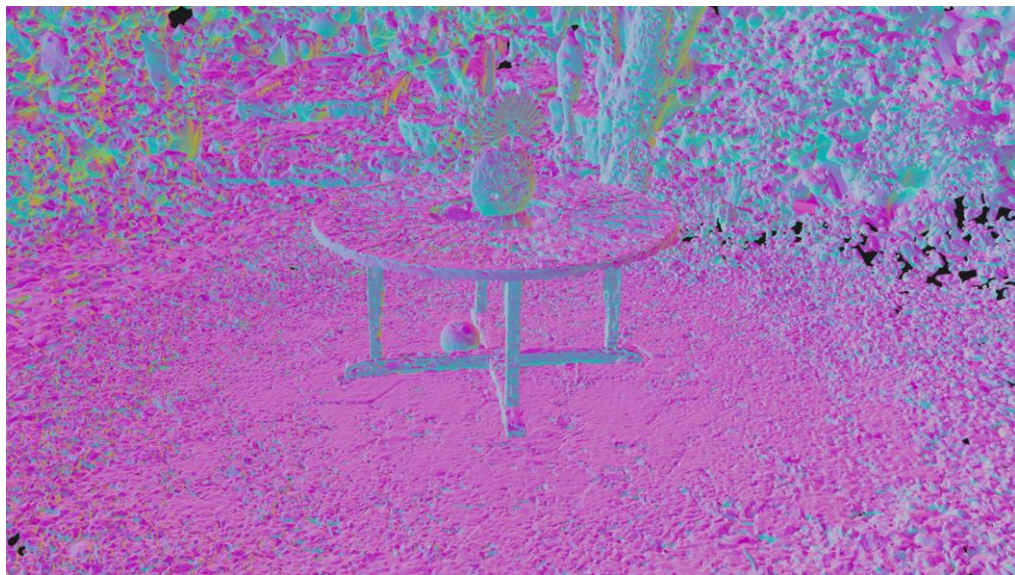
Mesh extraction: comparison

Marching Cubes, no regularization (vanilla 3DGS):



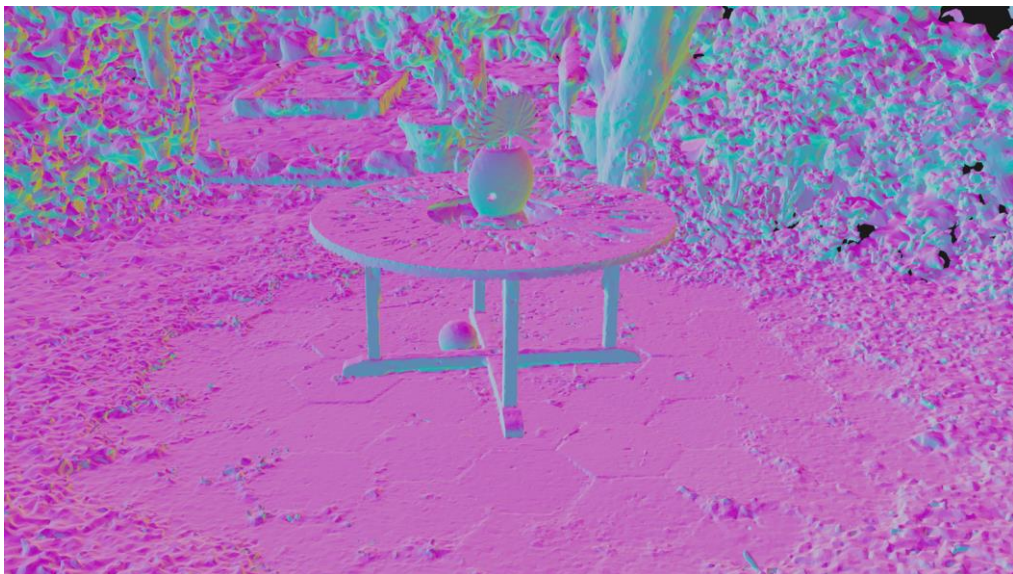
Mesh extraction: comparison

Our extraction method, no regularization (vanilla 3DGS):



Mesh extraction: comparison

Our extraction method, our regularization

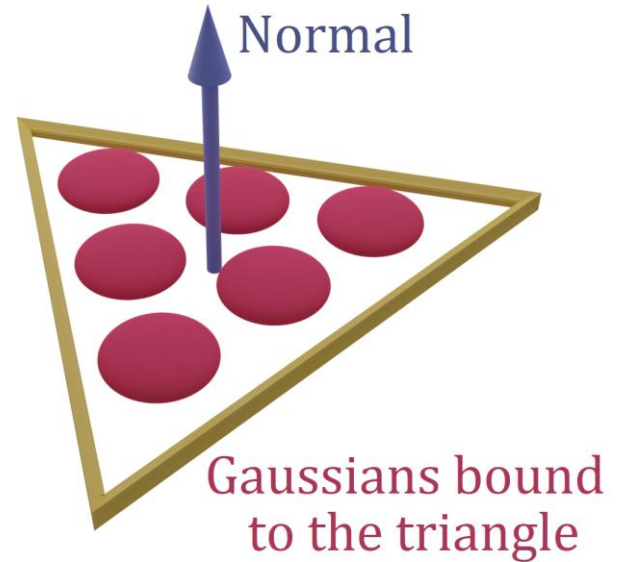


Hybrid Representation: Mesh + Gaussians

Binding Gaussians to triangles

We sample **flat Gaussians** in the triangles of the mesh, and change the parameterization of the Gaussians so that they **automatically adjust** when **deforming, rotating, scaling**, or **animating** the mesh.

- We use **barycentric coordinates** in the triangles for the means
- The rotation matrix is written in the coordinate space of the triangle. We only learn a 2D **rotation in the triangle's plane**, encoded with a **complex number $x+iy$**
- The scaling factor **along the normal is very small**
- We automatically adjust rotations and scaling with simple rules when deforming the triangle



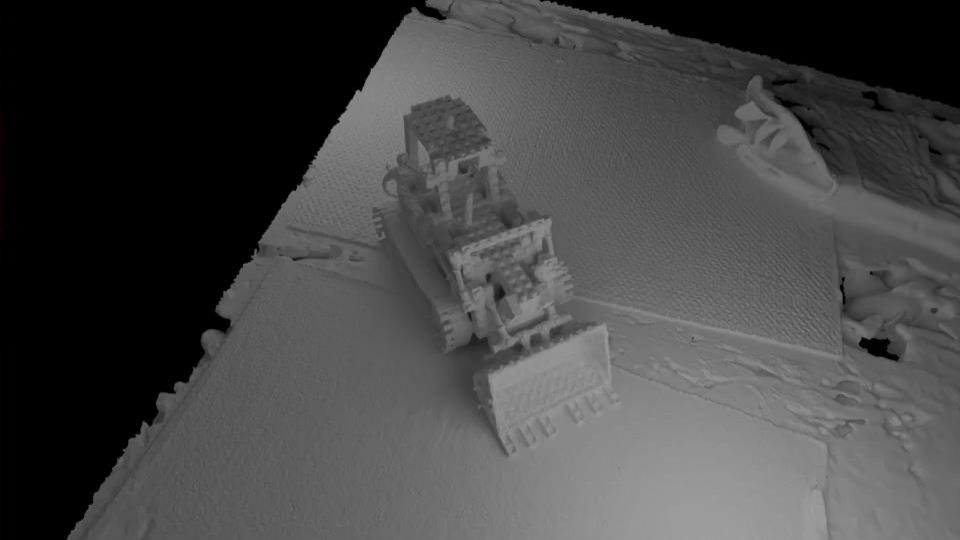
Similar approach

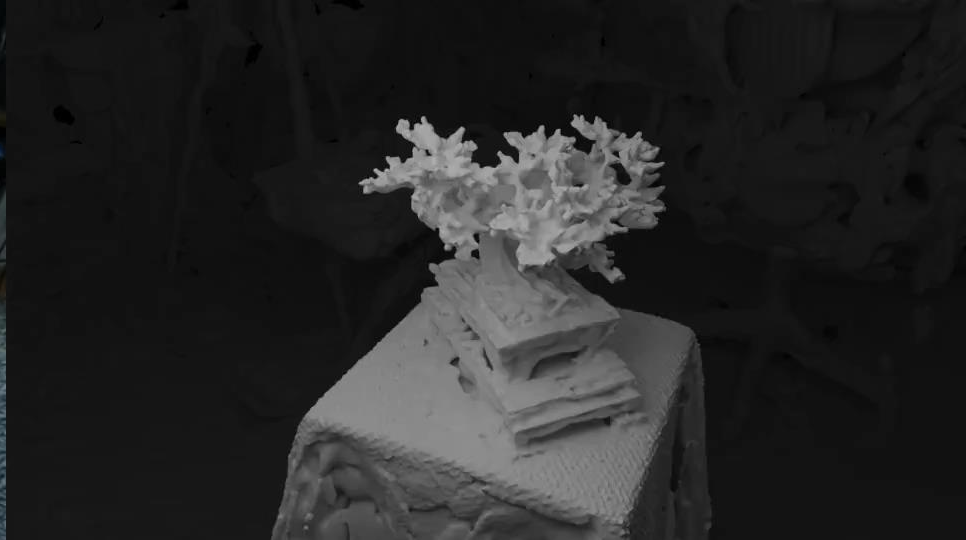
Mesh-based Gaussian Splatting for Real-time Large-scale Deformation, Gao *et al.*

- This approach considers the **mesh as an input**, and focuses on **parameterizing Gaussians on the surface of the mesh**.
- The parameterization is very similar to SuGaR (each Gaussian is bound to a triangle of the mesh, in a similar way) but authors push it further (more sophisticated)!
- Their parameterization works well for **adapting Gaussians' parameters to large-scale deformations** of the mesh.
- Super interesting!

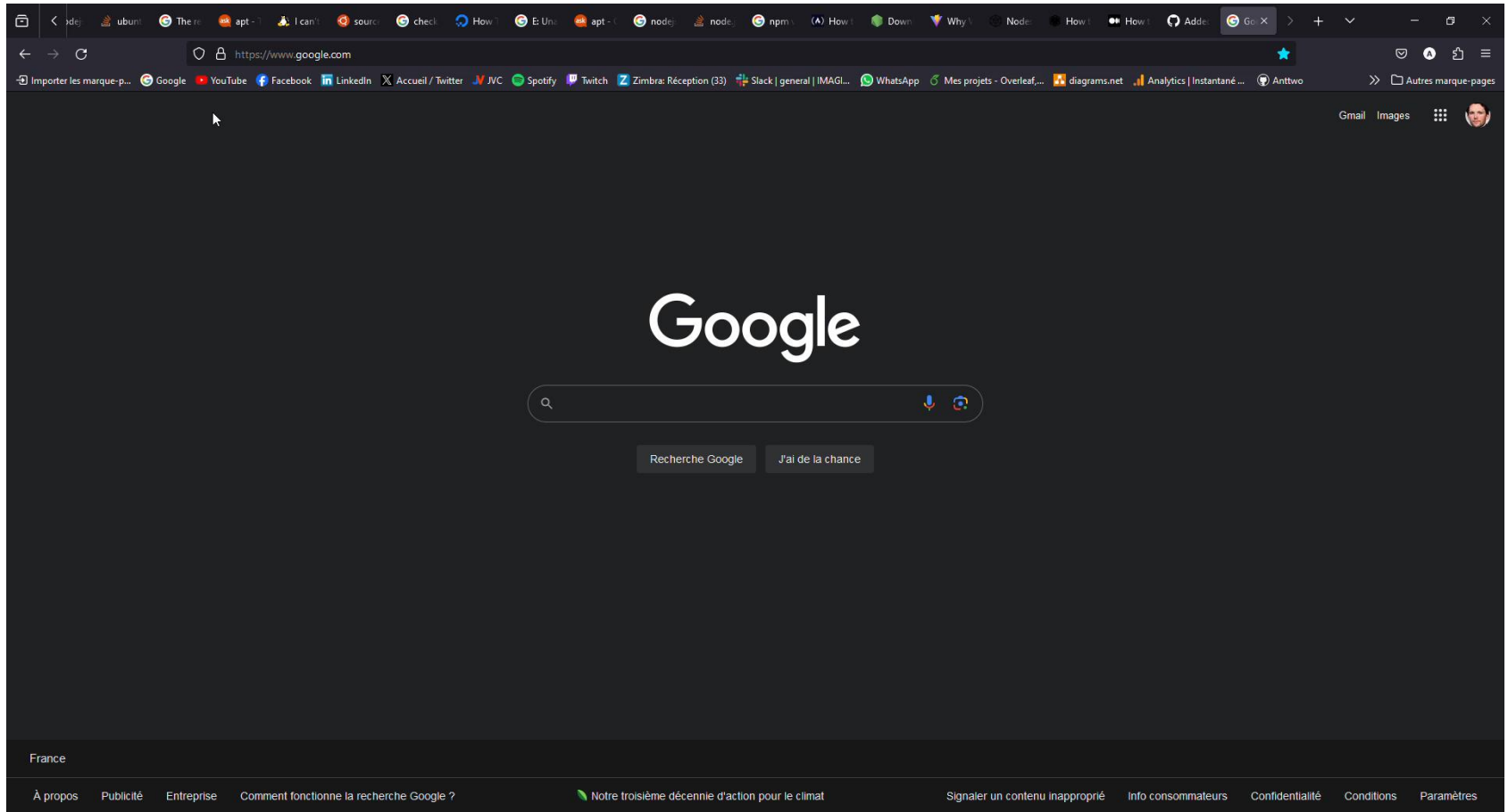


Results

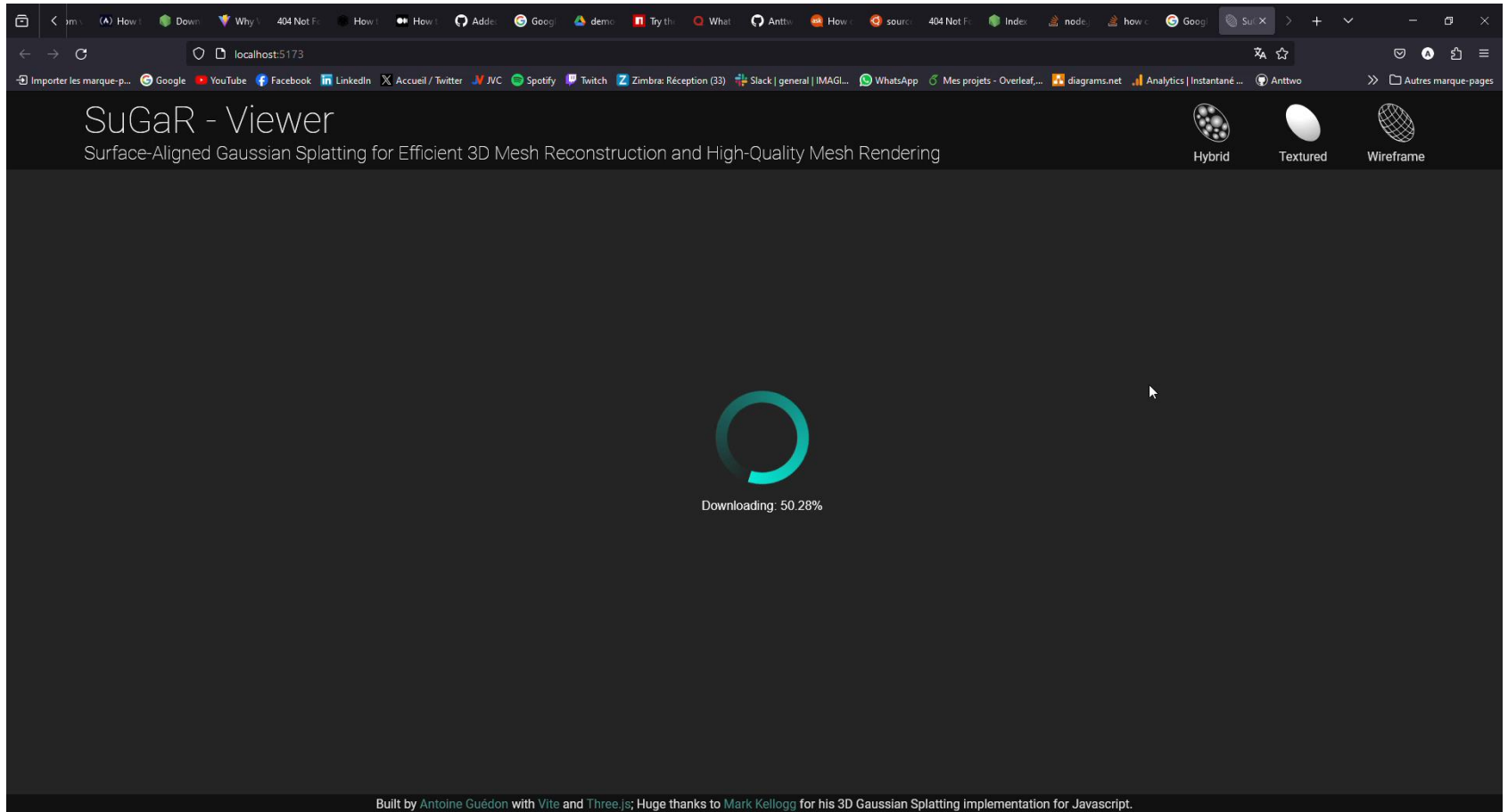




Dedicated viewer: Bicycle



Dedicated viewer: Playroom



localhost:5173

SuGaR - Viewer

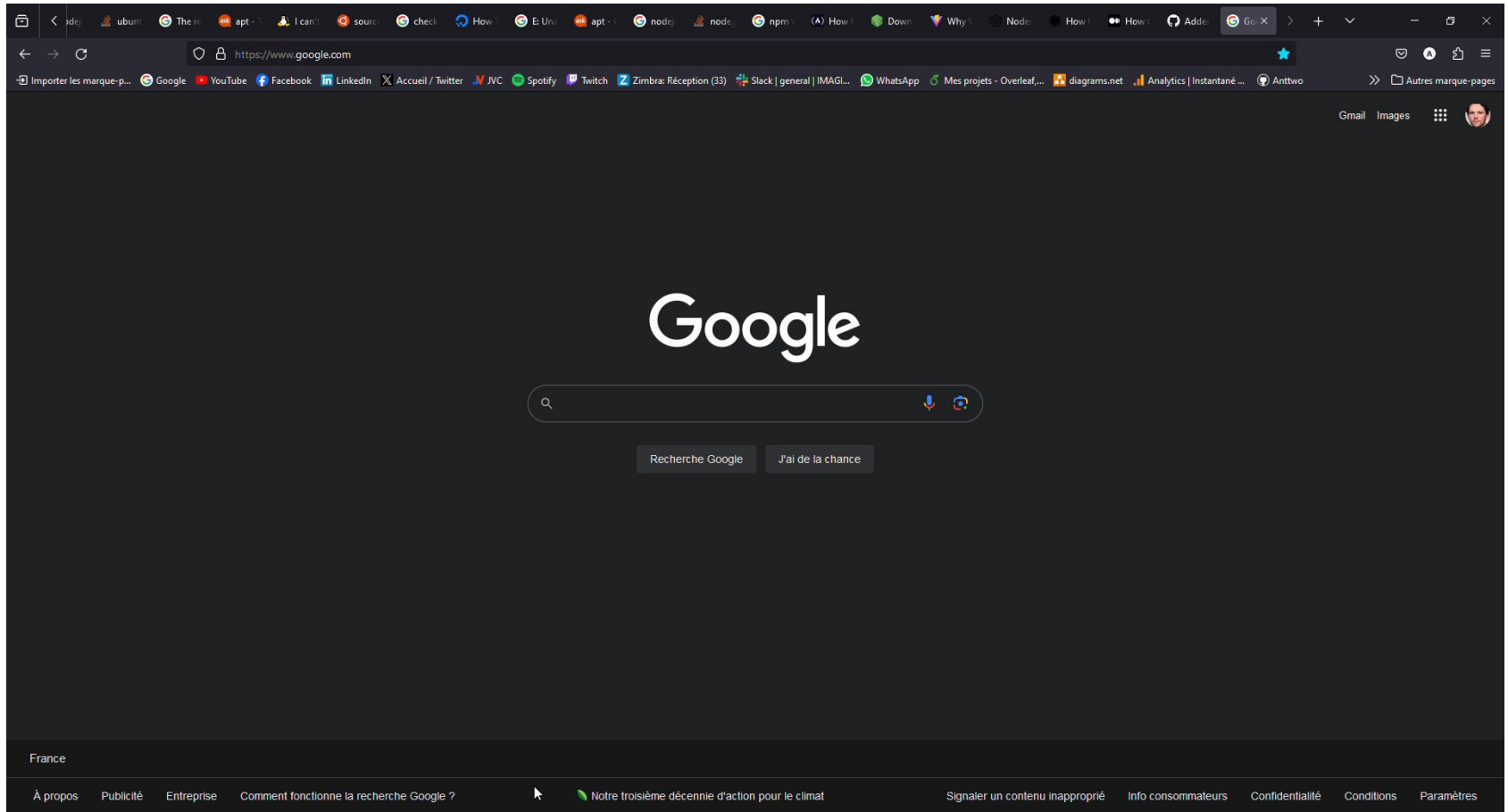
Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering

Hybrid Textured Wireframe

Downloading: 50.28%

Built by Antoine Guédon with Vite and Three.js; Huge thanks to Mark Kellogg for his 3D Gaussian Splatting implementation for Javascript.

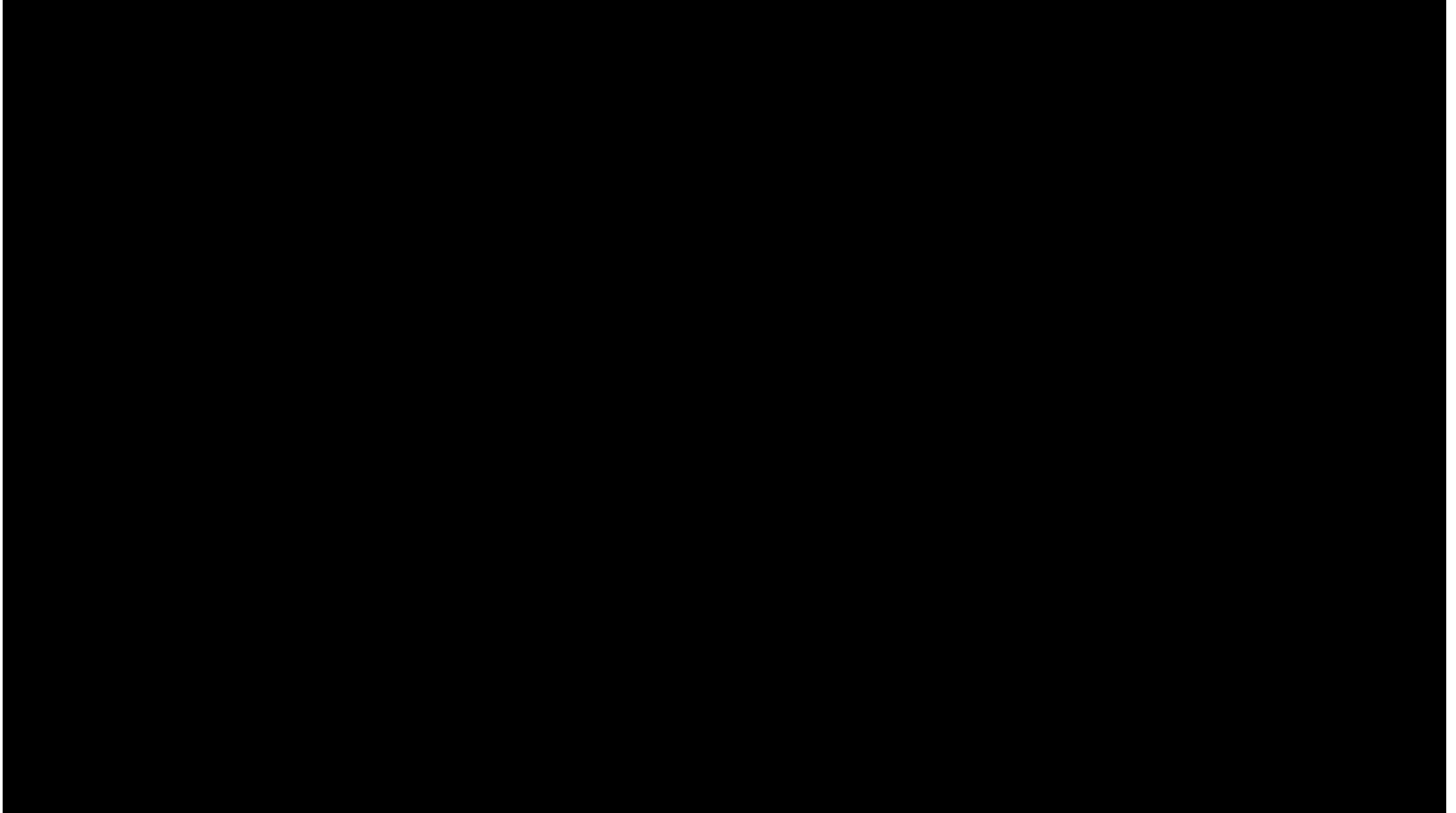
Dedicated viewer: Kitchen



Composition



Composition



Composition + Animation



Composition and animation with traditional softwares like Blender

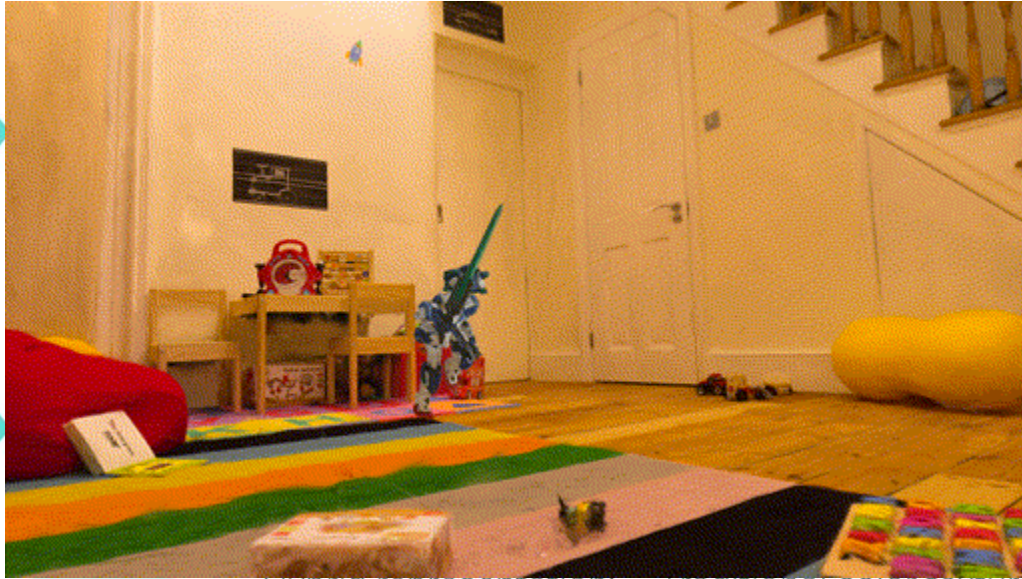


Input images or video

Hybrid representation:
Mesh + Gaussians

Traditional color texture
for the mesh

Composition + Animation



Hybrid representation: Mesh + Gaussians
Traditional color texture for the mesh

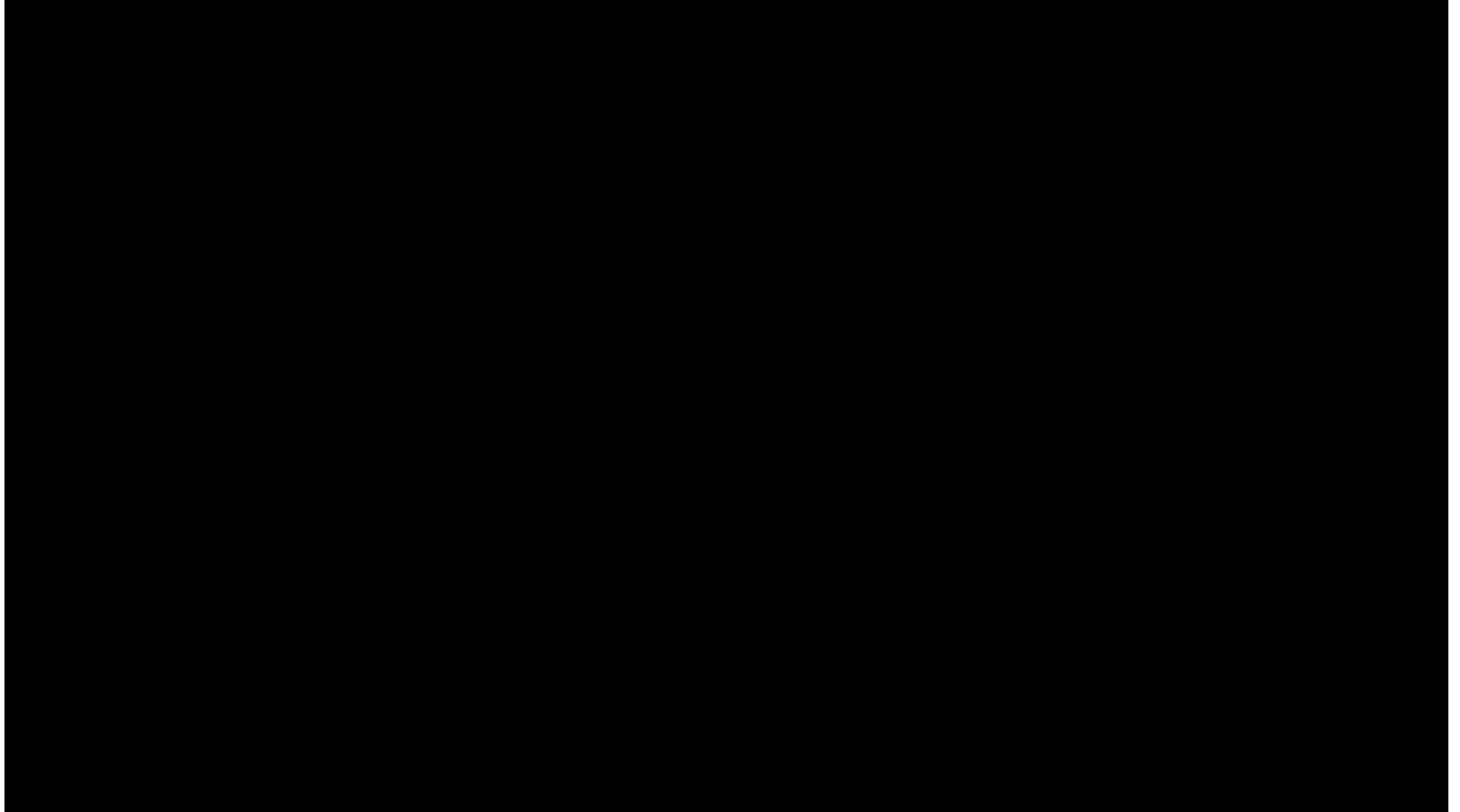


Input images or video



Composition and animation with traditional softwares like Blender

Composition + Animation



Limitations

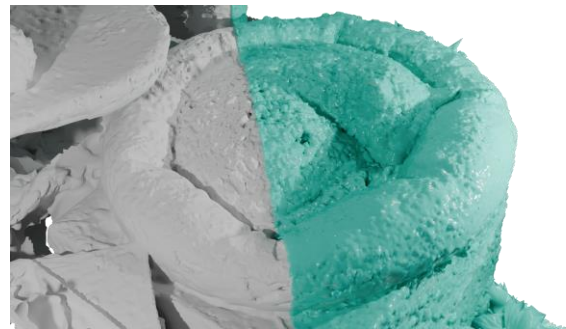
- Gaussians “cheat” on the geometry to recreate specularity. SuGaR’s regularization mitigates this issue, but highly specular surfaces can still generate bumps or cavities in the mesh.
- SuGaR reaches lower rendering performance than vanilla 3DGS. Indeed, when reconstructing the world as a surface, it becomes much harder to recover volumetric effects and fuzzy materials, like hair or grass.

Gaussian Frosting

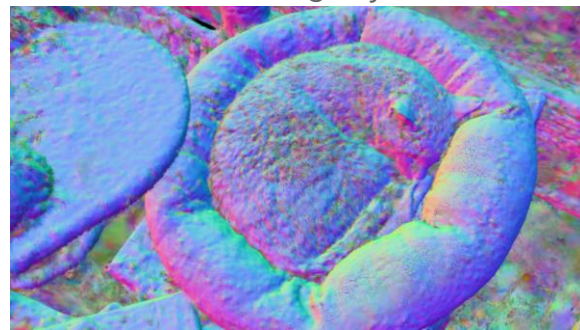
Gaussian Frosting: Editable Complex Radiance Fields



Rendering



Frosting layer



Normals

Gaussian Frosting: Editable Complex Radiance Fields



Gaussian Frosting: Editable Complex Radiance Fields

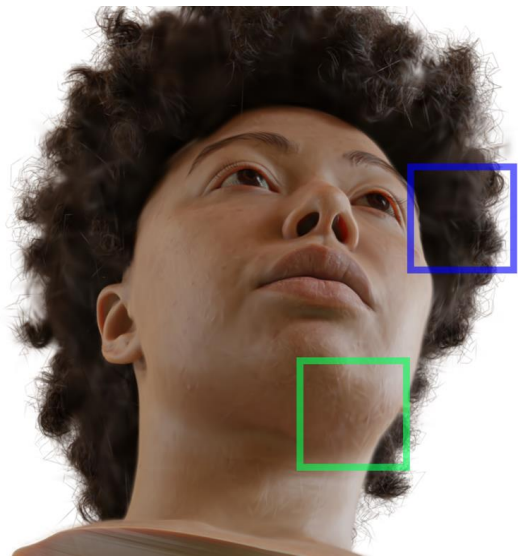




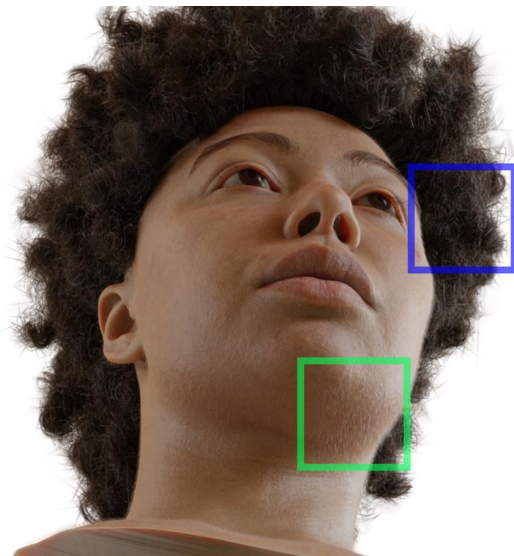
SuGaR

Frosting

Gaussian Frosting: Editable Complex Radiance Fields



3DGS



Frosting



SuGaR's code is available here:
<https://github.com/Anttwo/SuGaR>

If you like the project, please consider leaving a star!
We will update the repo very soon with some big changes!

Thank you so much! Here's my cat for conclusion:

