Gaussian What ⁉

Rendering Gaussian Splats on the Web

Fivos
DOGANIS

Seminal Paper

[3D Gaussian Splatting for Real-Time Radiance Field Rendering]

INRIA, SIGGRAPH 2023

3D Gaussian Splatting for Real-Time Radiance Field Rendering

BERNHARD KERBL*, Inria, Université Côte d'Azur, France
GEORGIOS KOPANAS*, Inria, Université Côte d'Azur, France
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany
GEORGE DRETTAKIS, Inria, Université Côte d'Azur, France

Fig. 1. Our method achieves real-time rendering of radiance fields with quality that equals the previous method with the best quality [Barron et al. 2022], while only requiring optimization times competitive with the fastest previous methods [Fridovich-Keil and Yu et al. 2022; Müller et al. 2022]. Key to this performance is a novel 3D Gaussian scene representation coupled with a real-time differentiable renderer, which offers significant speedup to both scene optimization and novel view synthesis. Note that for comparable training times to InstantNGP [Müller et al. 2022], we achieve similar quality to theirs; while this is the maximum quality they reach, by training for 51min we achieve state-of-the-art quality, even slightly better than Mip-NeRF360 [Barron et al. 2022].

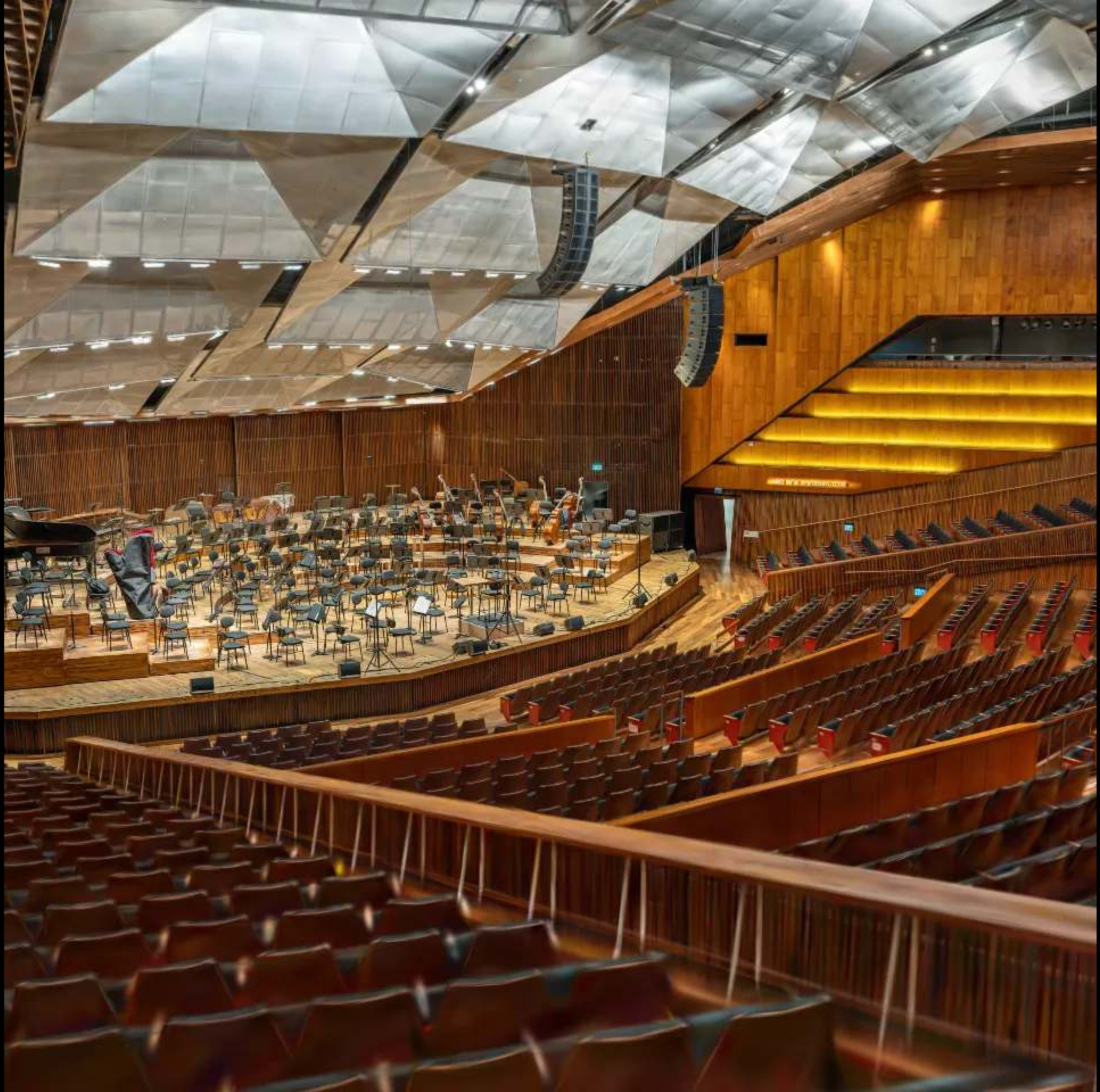
Radiance Field methods have recently revolutionized novel-view synthesis of scenes captured with multiple photos or videos. However, achieving high visual quality still requires neural networks that are costly to train and ren-

Additional Key Words and Phrases: novel view synthesis, radiance fields, 3D gaussians, real-time rendering

ACM Reference Format:

Today


QUECHUA

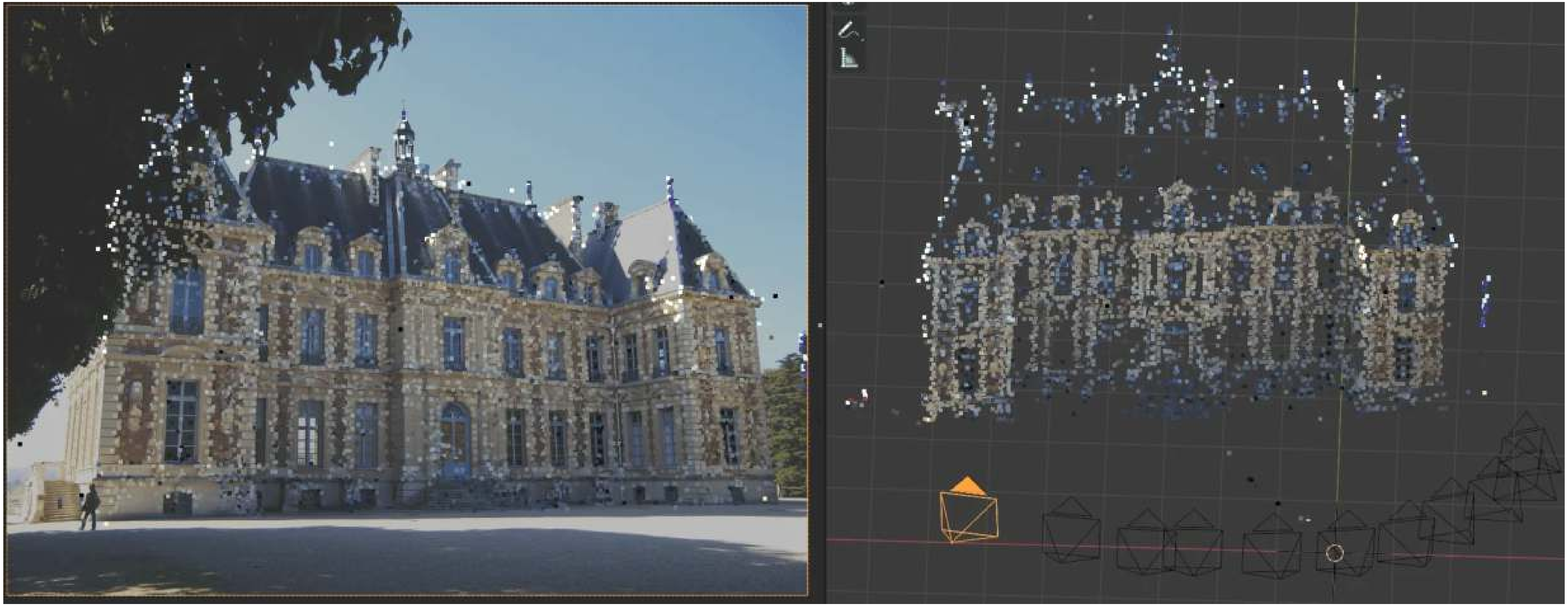


© Adam HAVKIN : Lowy Concert Hall

3D Points

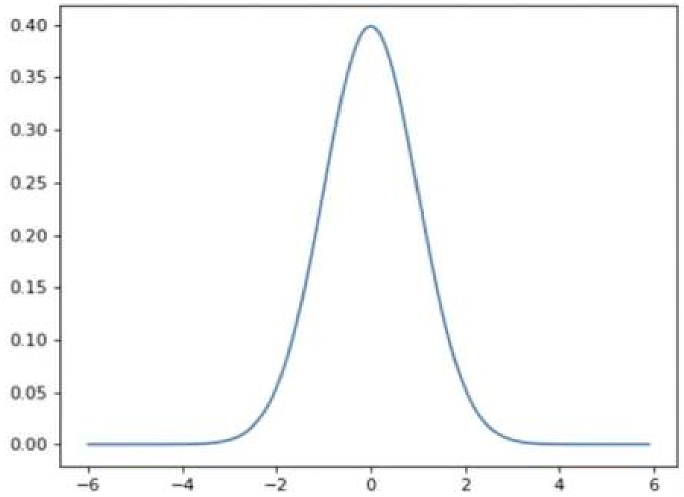
cover picture by Luma AI, taken from here

Sparse Point Cloud produced by SfM

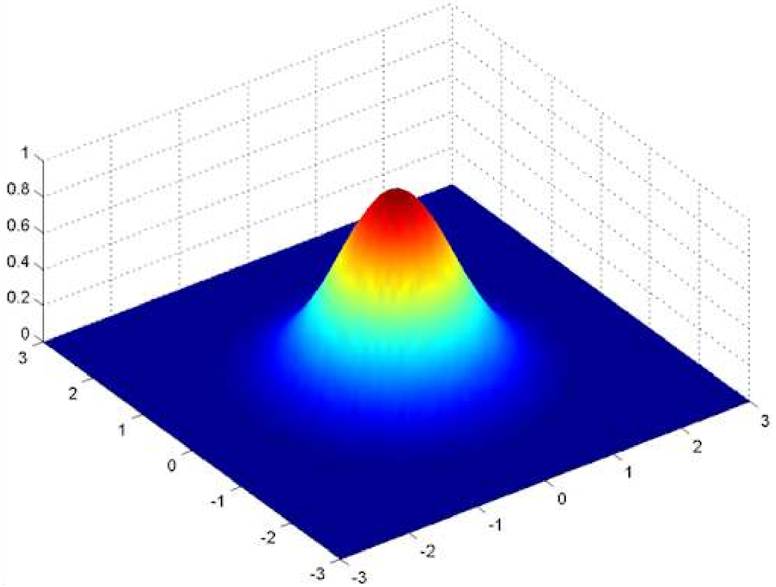


3D Points ➡️ Gaussian Splats

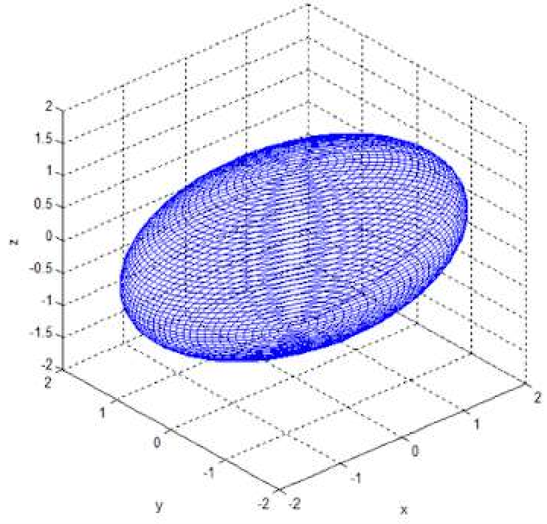
1D Gaussian



2D Gaussian



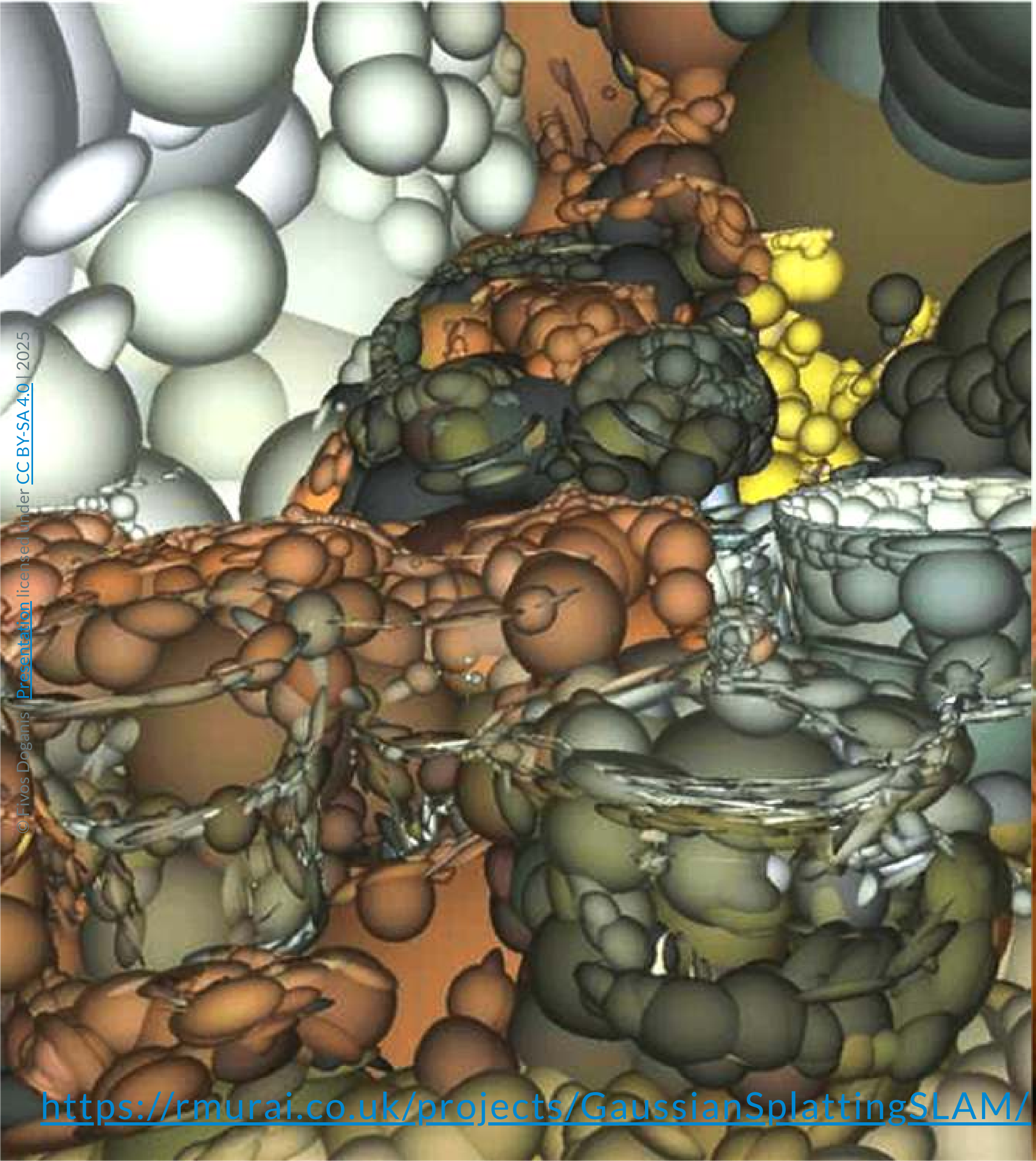
3D Gaussian



https://pyimagesearch.com/2024/12/09/3d-gaussian-splatting-vs-nerf-the-end-game-of-3d-reconstruction/

Gaussian Splats

- 3D points / particles ➡️ ~~blobs~~ **3D Gaussians** projected in 2D
- Rendering Steps
  - **Project** each gaussian into 2D from the camera perspective.
  - **Sort** the gaussians by depth ⏳
  - For each **pixel**
    - iterate over each gaussian **front-to-back**
    - **blend** them together
- See training and rasterization details



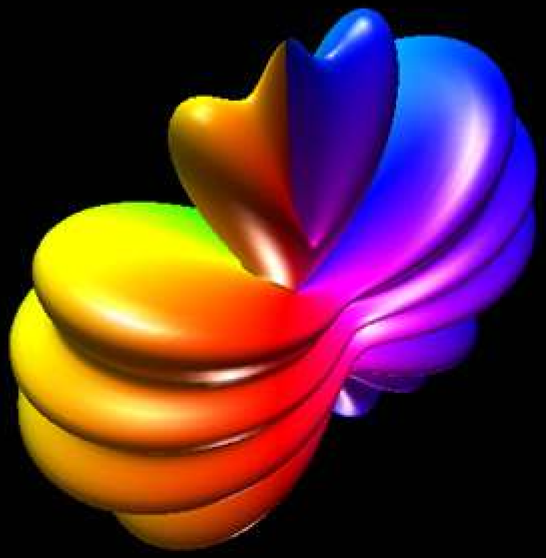
https://rmurai.co.uk/projects/GaussianSplattingSLAM/

https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362/

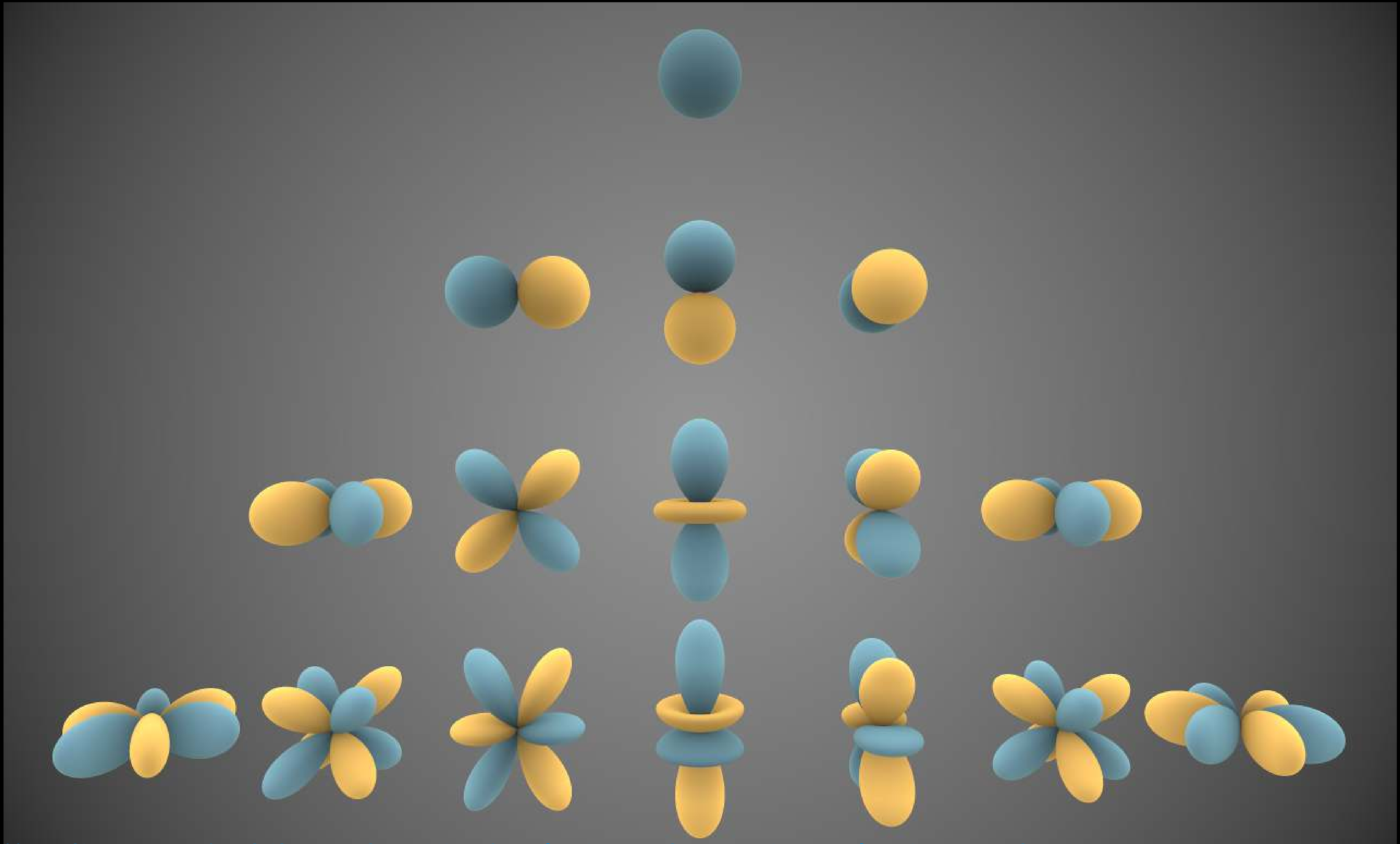
Gaussian Splats ⭐

- 3D points / particles ➡️ ~~blobs~~ **3D Gaussians** projected in 2D
- each particle has a
  - **position**: x, y, z (mean μ)
  - **rotation** + non-uniform **scale**: Mat3 (covariance Σ)
  - **opacity** (sigmoid $\sigma(\alpha)$)
  - **view-dependent color**: r, g, b + SH Coeffs (**Spherical Harmonics**)

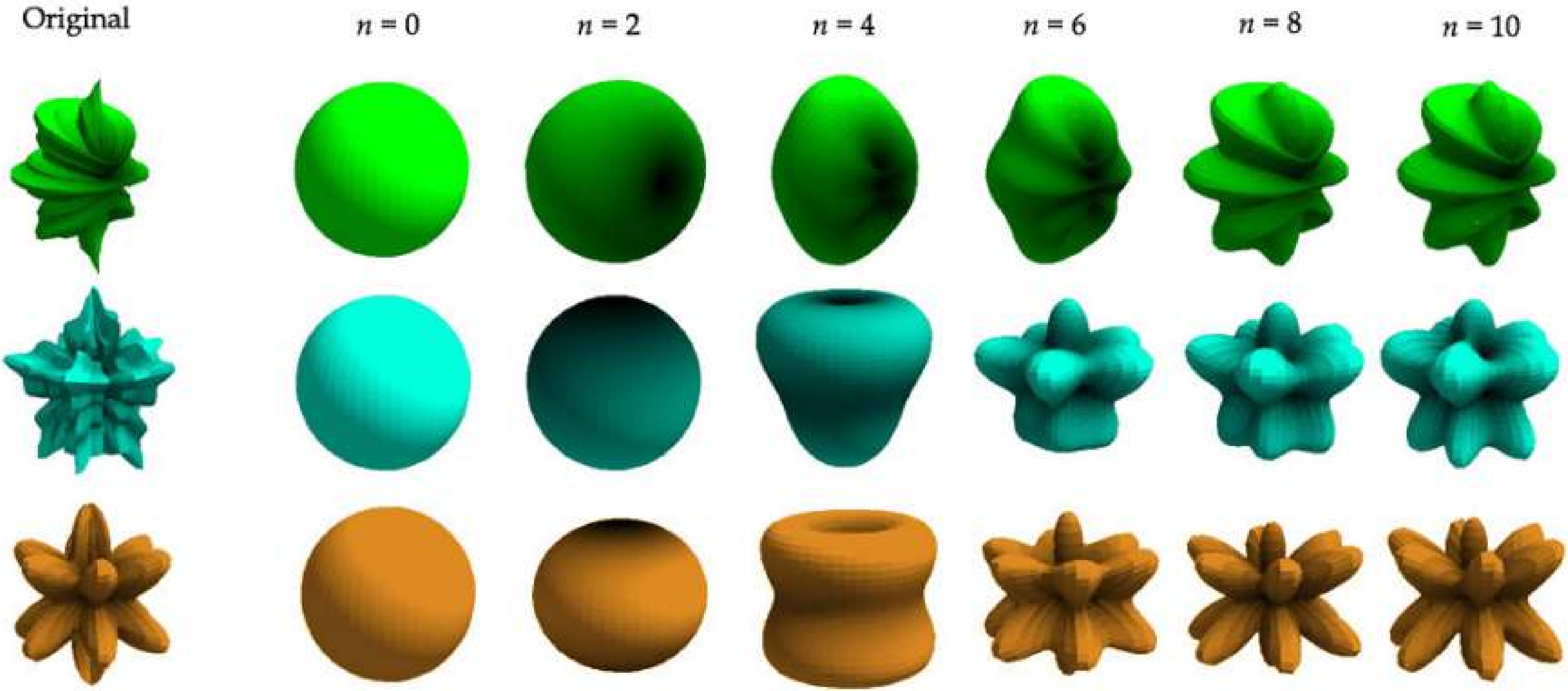
Spherical Harmonics



https://paulbourke.net/geometry/sphericalh/



https://medium.com/@thibaut.chauffier/exploring-3d-gaussian-splatting-e50b8c9cee93



https://patapom.com/blog/SHPortal/

Spherical Harmonics (SH)

- form a **orthonormal basis on a sphere**
- each function defined on a sphere can be expressed through a combination of SH basis functions

https://towardsdatascience.com/a-comprehensive-overview-of-gaussian-splatting-e7d570081362/#4cd8

File Formats

.PLY file format

- similar to RGB point cloud
  - basic color described using `f_dc_` values
- optional coefficients and parameters
- nx, ny, nz unused

```
ply
format binary_little_endian 1.0
element vertex 1534456
property float x
property float y
property float z
property float nx
property float ny
property float nz
property float f_dc_0
property float f_dc_1
property float f_dc_2
property float f_rest_0
(... f_rest from 1 to  43...)
property float f_rest_44
property float opacity
property float scale_0
property float scale_1
property float scale_2
property float rot_0
property float rot_1
property float rot_2
property float rot_3
end_header
```

https://www.magnopus.com/blog/how-we-wrote-a-gpu-based-gaussian-splats-viewer-in-unreal-with-niagara

Other formats

- **.SPLAT**
  - created by antimatter15
  - supported by PlayCanvas / SuperSplat
- **.SPZ**, "the JPG of 3D"
  - by Scaniverse
- ➡️ Use **.PLY** for maximum compatibility

Drawbacks

- **file size** (~100 MB per scene, if unoptimized)
- **rendering order** is important
  - **sorting** primitives ➡️ performance issues
  - **popping** artifacts
- optimizing can take a long time
- models need to be **cleaned** up
  - "**floaters**"
- **no real geometry** (for collision detection, relighting etc)
  - but convex hull mesh can be created from point cloud

Web Viewers

- SuperSplat by PlayCanvas
    - viewer + **free editor**
- Scaniverse by Niantic ⭐
    - viewer + **best free mobile scanning app (local mode)**
- lumai.ai
    - beautiful captures (view only) + mobile capture app (cloud)
- Polycam
    - mobile capture app (cloud) + **viewer** (download with account)
- splatter.app
    - **highly optimized** viewer
- blurry : viewer


WESTERN
PACIFIC

https://blog.42yeah.is/rendering/opengl/2023/12/20/rasterizing-splats.html

Web Libraries

Main Web implementations

- Kevin Kwok (@antimatter)
  - reference WebGL implementation (MIT)
- Mark Kellogg (@mkkellogg)
  - THREE.js implementation (MIT)
- ~~Forge.dev~~ ➡️ Spark (new name)
  - THREE.js implementation (MIT) : easier to use, but slower
- Marcus Andreas Svensson
  - WebGPU implementation (MIT)

- Gaussian Splats using HTML, based on antimatter15 **(demo)**

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://aframe.io/releases/1.4.2/aframe.min.js"></script>
    <script src="https://quadjr.github.io/aframe-gaussian-splatting/index.js"></script>
  </head>
  <body>
    <a-scene renderer="antialias: false">
      <a-entity rotation="10 0 0">
        <a-entity position="1.2 1.2 -2.7"
        animation="property: rotation; to: 0 360 0; dur: 10000; easing: linear; loop: true">
          <a-sphere position="0 0 0.5" radius="0.2" color="#EF2D5E"></a-sphere>
          <a-box position="0.5 0 0" rotation="0 45 0" height="0.4" width="0.4"
          depth="0.4" color="#4CC3D9" shadow></a-box>
          <a-cylinder position="0 0 -0.5" radius="0.25" height="0.4" color="#FFC65D" shadow></a-cylinder>
        </a-entity>
      </a-entity>
      <a-entity gaussian_splatting="src: https://huggingface.co/cakewalk/splat-data/resolve/main/truck.splat;"
      rotation="0 0 0" position="0 1.5 -2"></a-entity>
      <a-sky color="#000"></a-sky>
    </a-scene>
  </body>
</html>
```

Gaussian Splats + Standard WebGL rendering

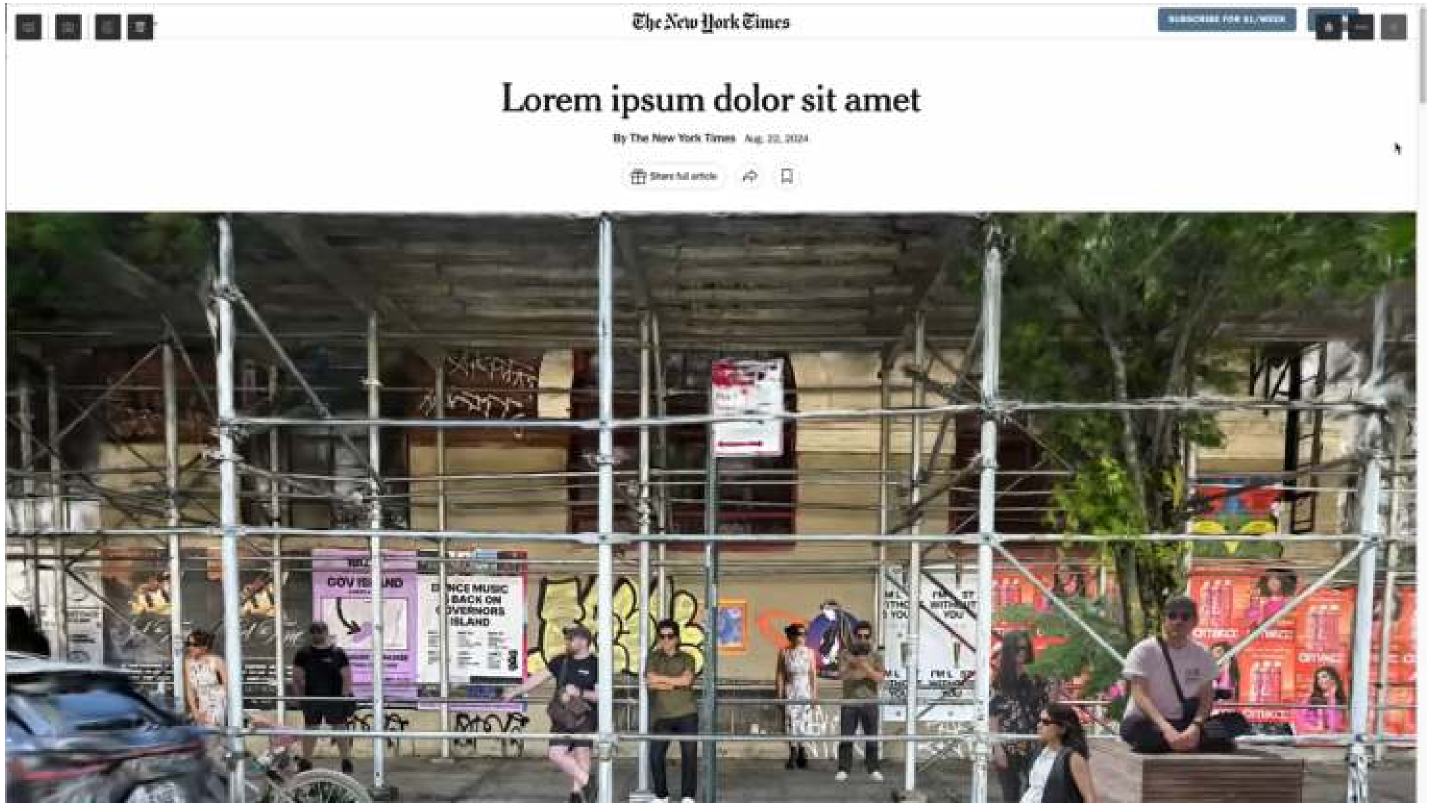
- making of
- article

https://www.8thwall.com/wbd/wonder-cars

Other Tools

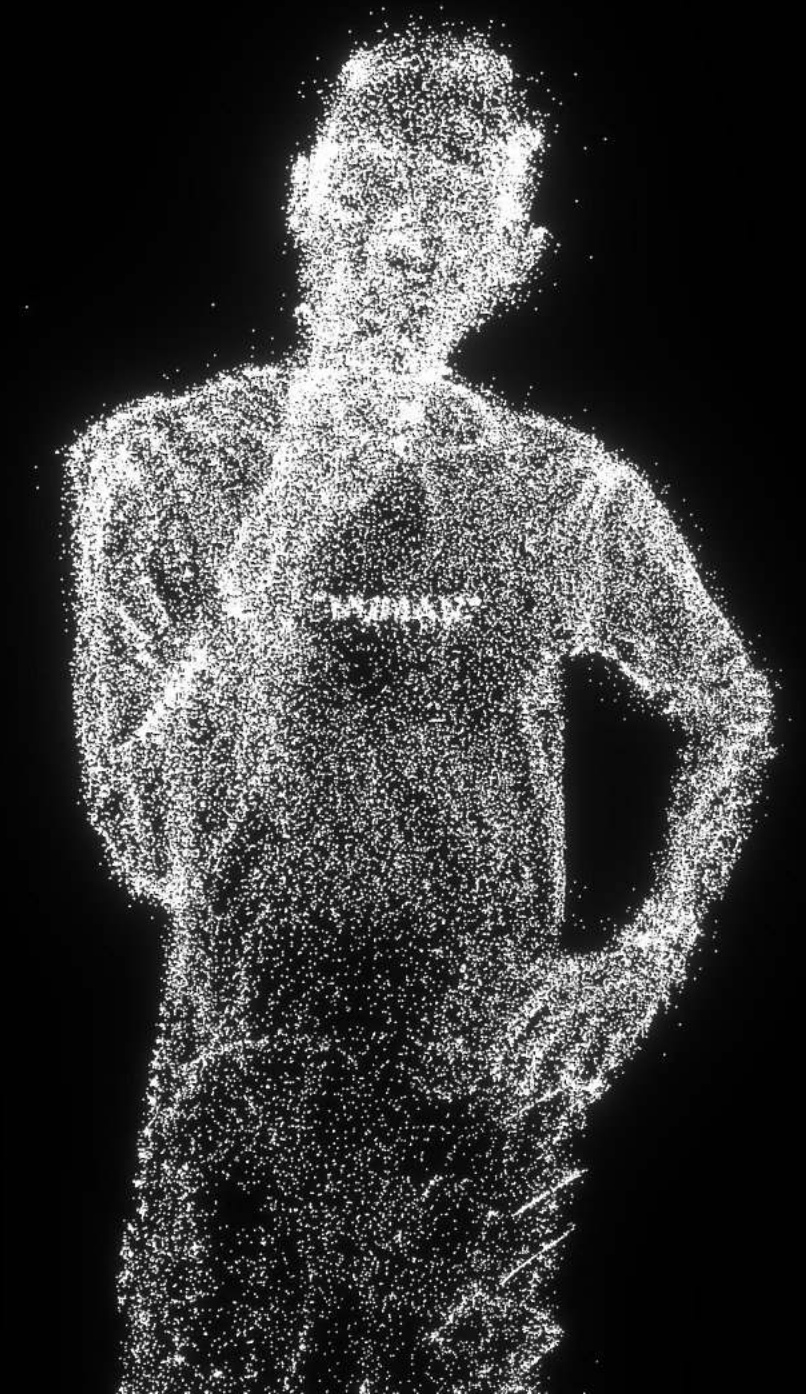
Postshot by Jawset

- Free Desktop app (beta)



The End

Questions?



The Future

https://tobias-kirschstein.github.io/gghead/