Computer Graphics Charles University

Advanced 3D Graphics for Movies and Games

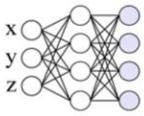
Lecture 11: Neural fields and Gaussian splatting

NPGR010
Winter Semester 2025/2026
Taught by Tomáš Iser

SIGGRAPH 2023 Course

- **Neural Fields for Visual Computing**
- https://doi.org/10.1145/3587423.3595477
- https://neuralfields.cs.brown.edu/siggraph23.html
- Towaki Takikawa and Shunsuke Saito and James Tompkin and Vincent Sitzmann and Srinath Sridhar and Or Litany and Alex Yu

- *The following slides are a combination of slides from the course above*



NEURAL FIELDS FOR VISUAL COMPUTING

SIGGRAPH 2023

Course - Sunday 6th August, 09:00-12:00 PDT

neuralfields.cs.brown.edu


Towaki Takikawa


Shunsuke Saito


James Tompkin


Alex Yu


Or Litany


Vincent Sitzmann


Srinath Sridhar

BROWN Computer Science

Computer Science UNIVERSITY OF TORONTO

Meta Reality Labs Research

LUMA AI

NVIDIA

TECHNION Israel Institute of Technology

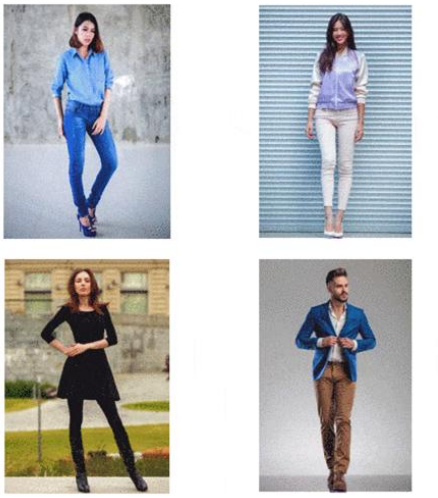
MIT

Neural Fields for Vision and Graphics

Captured images → Processing → Rendering of real-world place

[Mildenhall et al., Neural Radiance Fields (NeRF), ECCV 2020]
[Wu et al., Scalable Neural Indoor Scene Rendering, SIGGRAPH 2022]

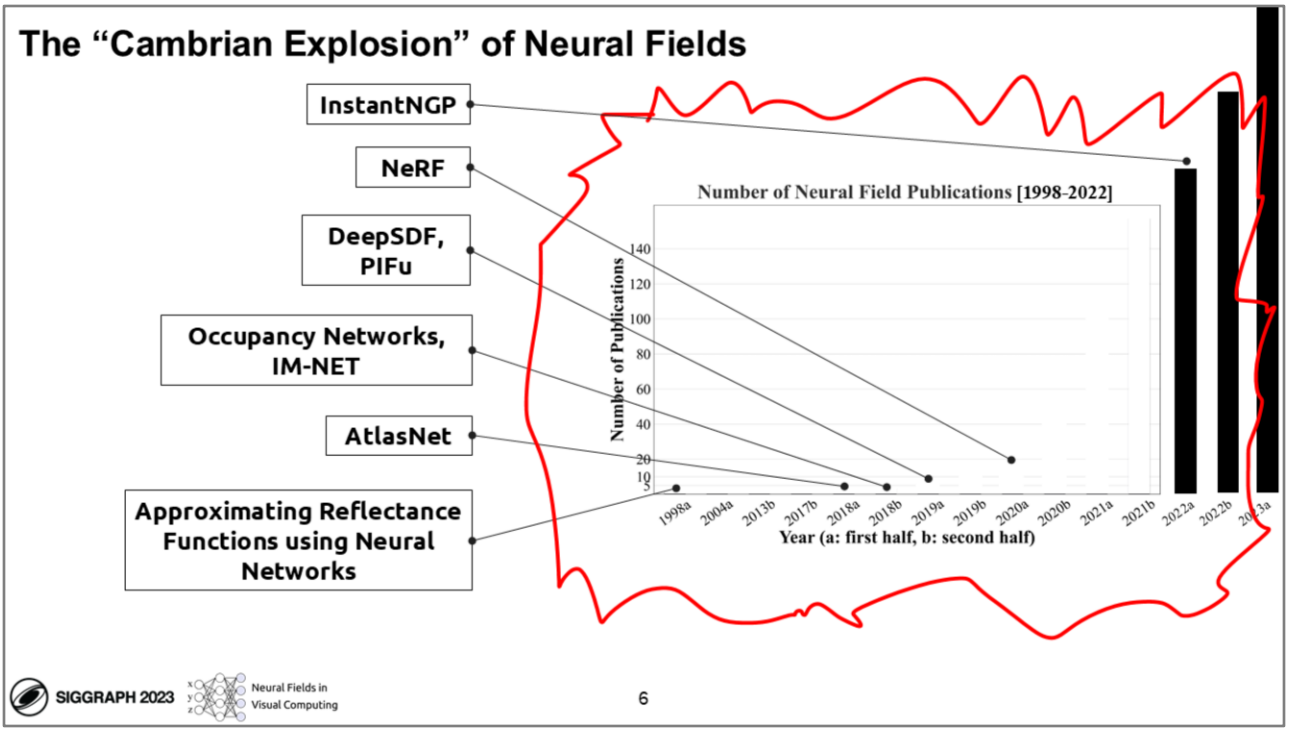
Neural Fields for Vision and Graphics



[Saito et al., PIFu, ICCV 2019]

[Chan et al., pi-GAN, CVPR 2021]

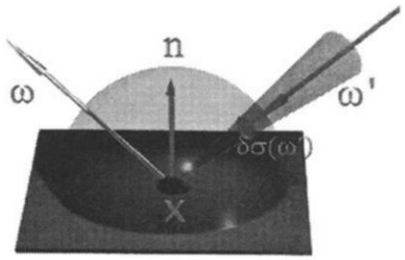
Priors



Just at CVPR 2023 two months ago, there were 170 papers on Neural Fields alone. That’s about as many as the whole SIGGRAPH Technical Papers program.

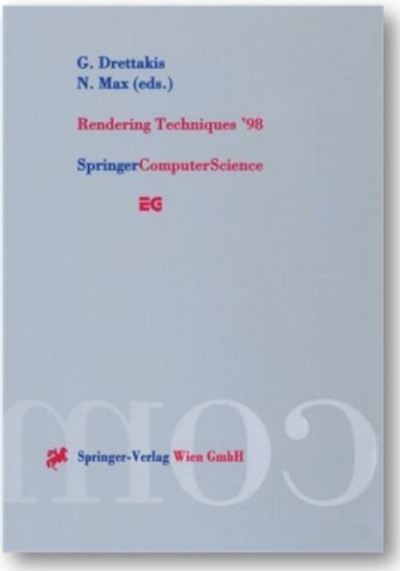
https://markboss.me/archive_presentations/cvpr23/

The “Cambrian Explosion” of Neural Fields



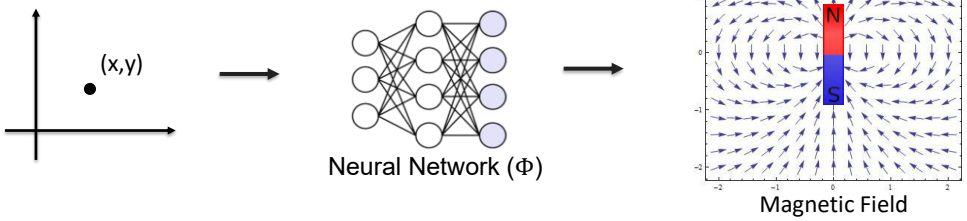
[Gargan and Neelamkavil 1998]

Approximating Reflectance Functions using Neural Networks

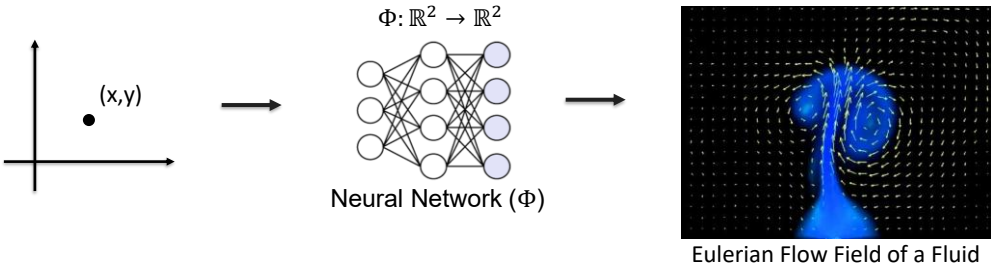


Rendering Techniques – old proceedings for Eurographics Symposium on Rendering

What are neural fields?



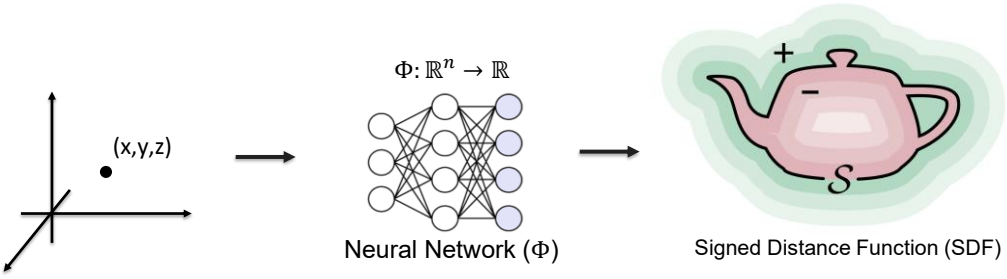
Magnetic Field

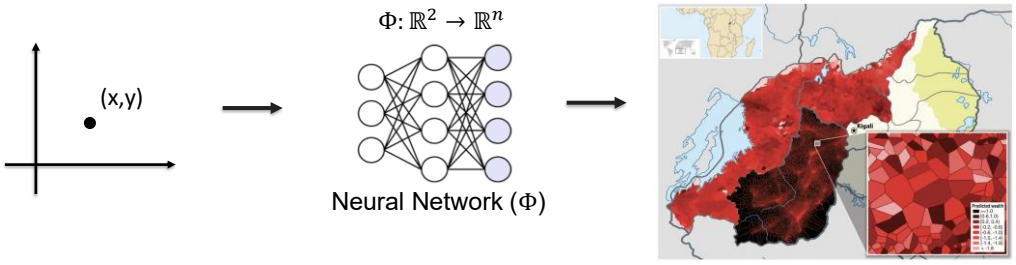


Eulerian Flow Field of a Fluid

[Koldora CC]

What are neural fields?



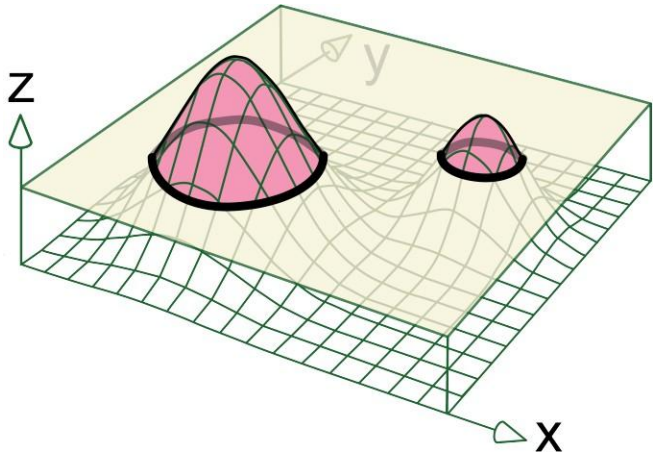


Kigali – capital of Rwanda

Geometry with Maths

$$f(x, y) = d$$

"Signed Distance Field"

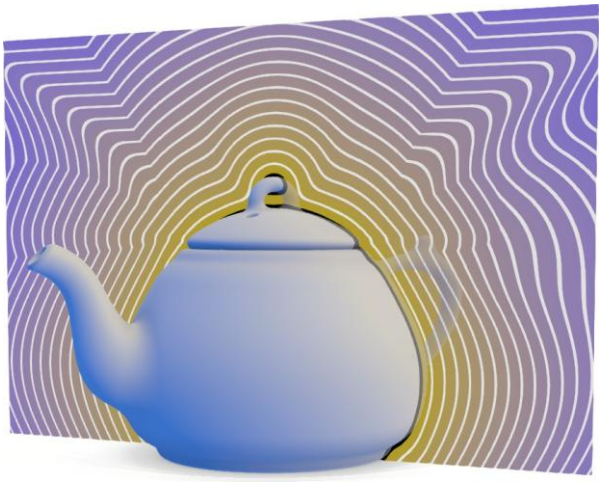


[Source: Takikawa et al]

Geometry with Maths

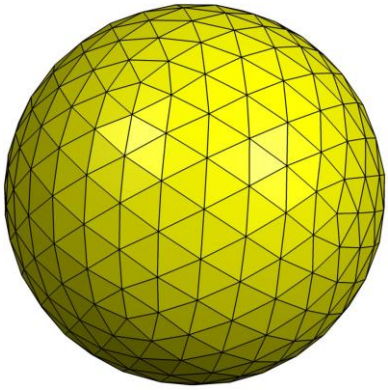
$$f(x, y, z) = d$$

Sometimes also referred to more generally as “Implicit Surfaces”



[Source: Takikawa et al]

Signed Distance Functions as Geometry

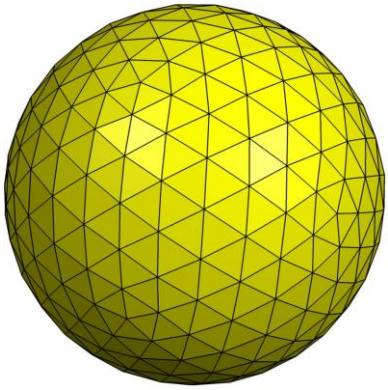


Polygon Mesh

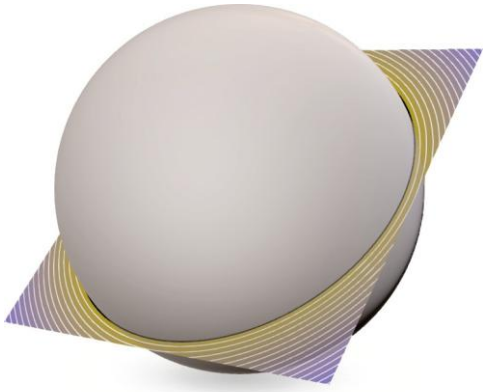
[Source: Wikipedia et al]

12

Signed Distance Functions as Geometry



Polygon Mesh



$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$

[Source: Wikipedia et al]

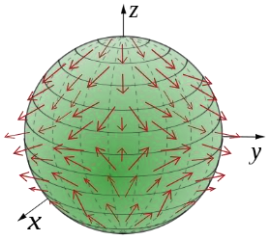
13

Definition

Definition 1
A *field* is a quantity defined for all spatial and / or temporal coordinates.

14

Examples of Fields

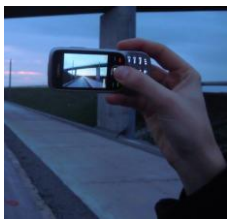


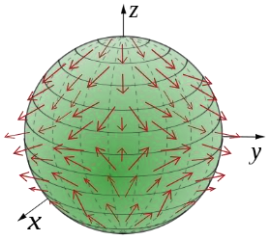
Vector Field

Fields

[Source: Wikipedia]

Examples of Fields





Image

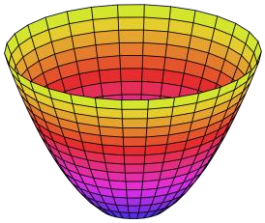
Vector Field

Fields

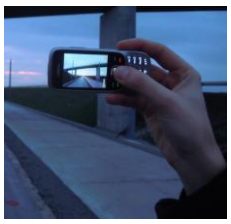
[Source: Wikipedia]

16

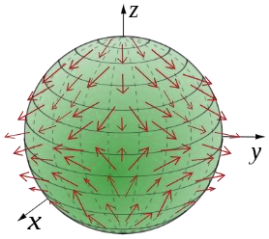
Examples of Fields



3D Parabola
(Explicit Surface)



Image



Vector Field

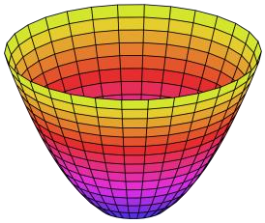
Fields

[Source: Wikipedia]

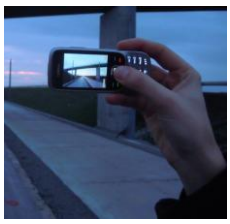
17

Examples of Fields

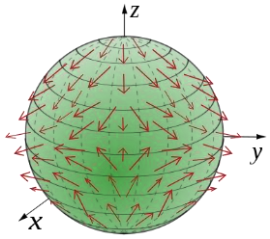
3D Signed Distance Fields
(Implicit Surface)



3D Parabola
(Explicit Surface)



Image

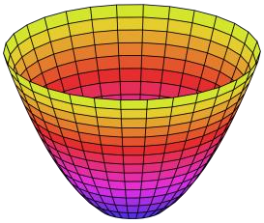


Vector Field

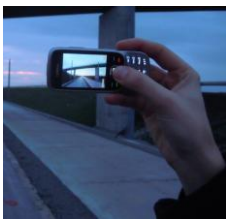
Fields

[Source: Wikipedia]

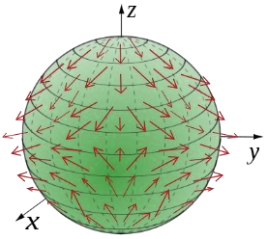
Examples of Fields



3D Parabola
(Explicit Surface)

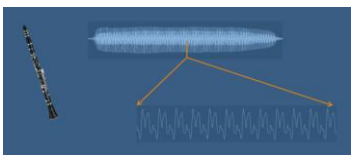


Image



Vector Field

3D Signed Distance Fields
(Implicit Surface)



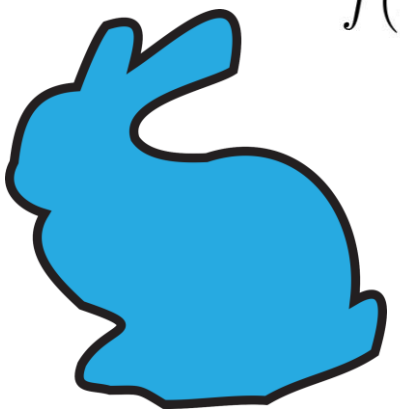
Audio

Fields

[Source: Wikipedia]

19

Sphere Tracing

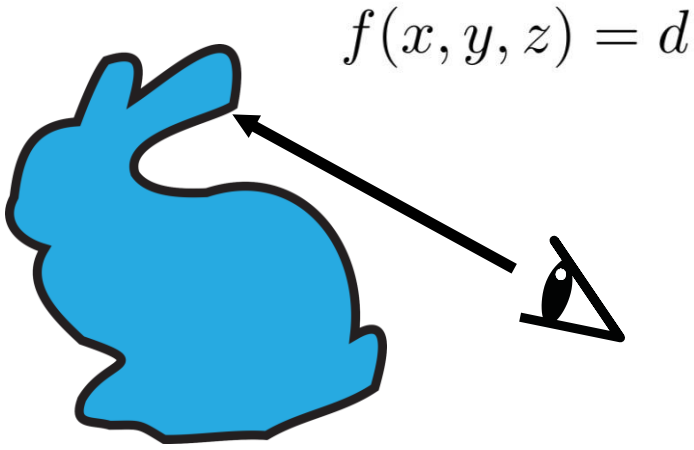


$$f(x, y, z) = d$$

[Source: Takikawa et al]

20

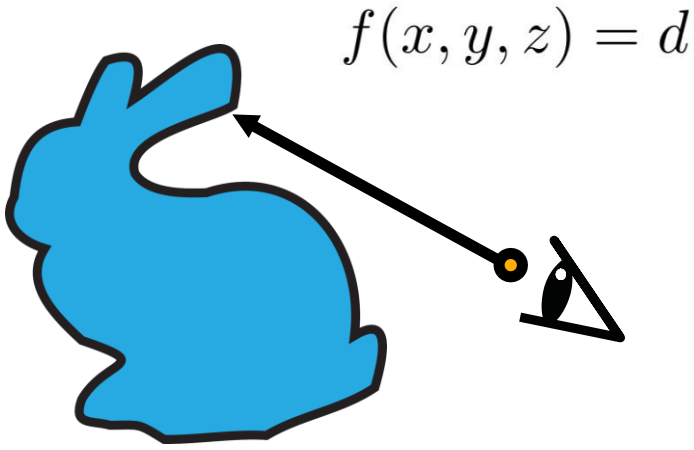
Sphere Tracing



[Source: Takikawa et al]

21

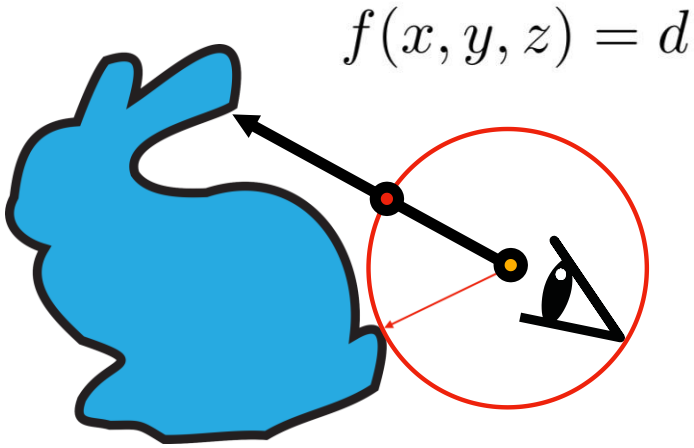
Sphere Tracing



[Source: Takikawa et al]

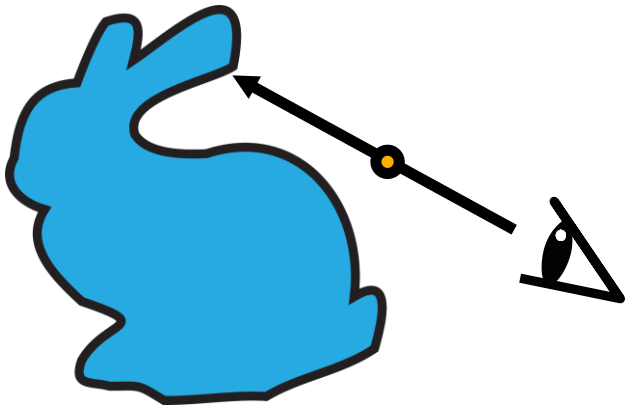
22

Sphere Tracing



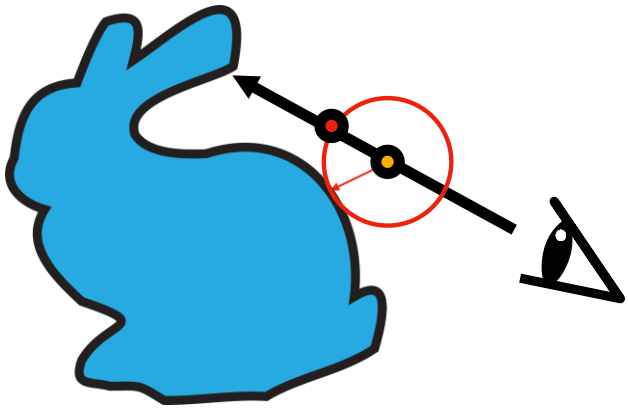
[Source: Takikawa et al]

Sphere Tracing



[Source: Takikawa et al]

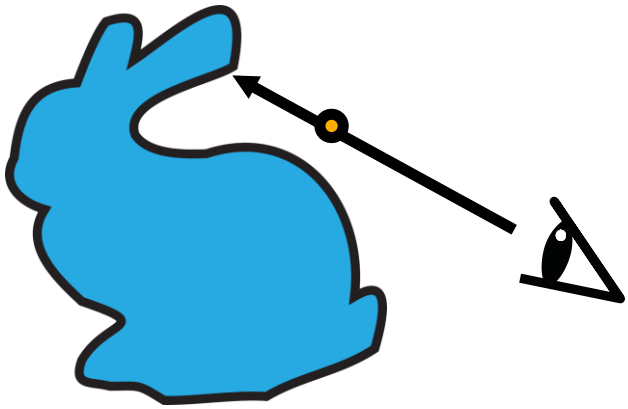
Sphere Tracing



[Source: Takikawa et al]

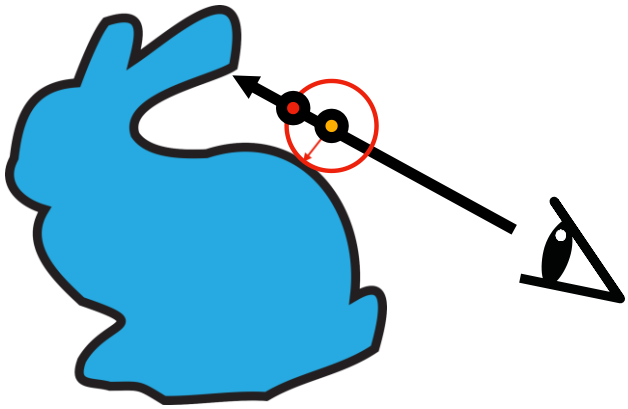
25

Sphere Tracing



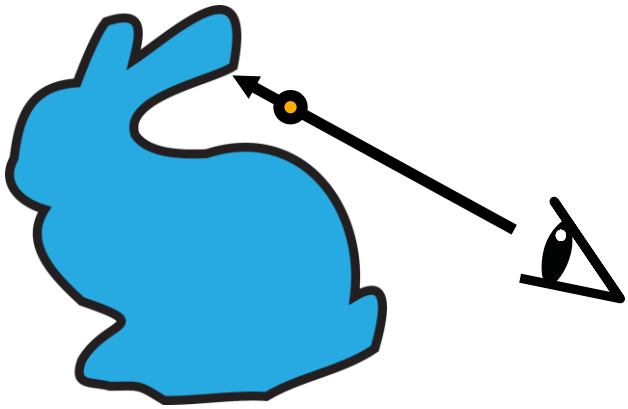
[Source: Takikawa et al]

Sphere Tracing



[Source: Takikawa et al]

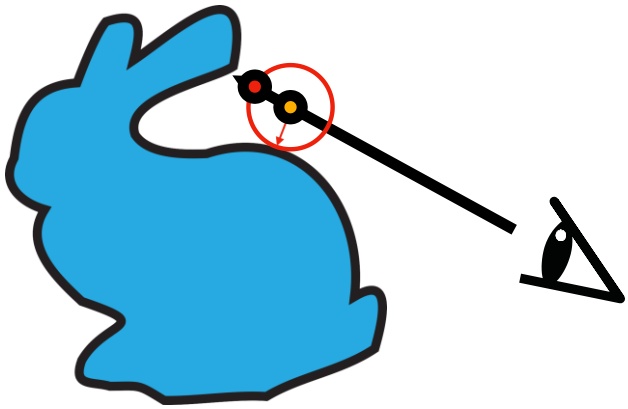
Sphere Tracing



[Source: Takikawa et al]

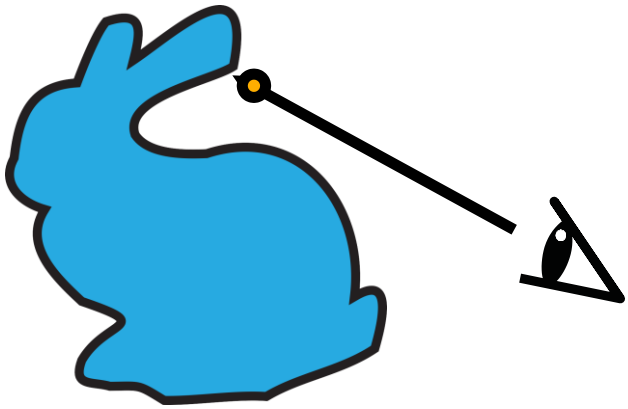
28

Sphere Tracing



[Source: Takikawa et al]

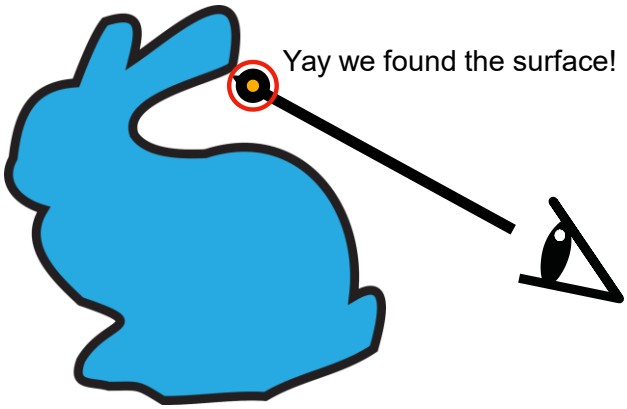
Sphere Tracing



[Source: Takikawa et al]

30

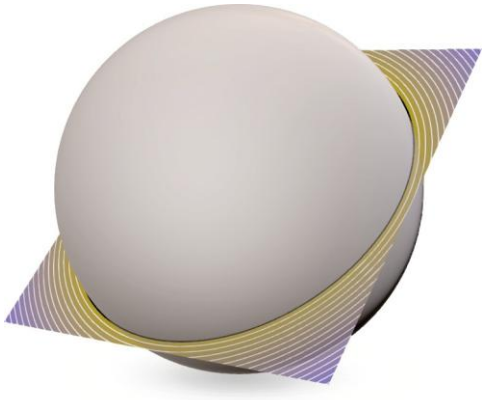
Sphere Tracing



[Source: Takikawa et al]

31

Signed Distance Functions as Geometry

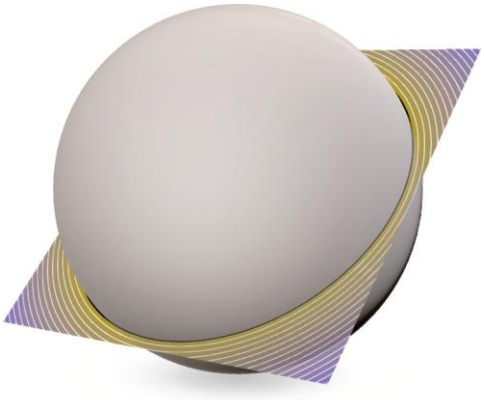


$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$

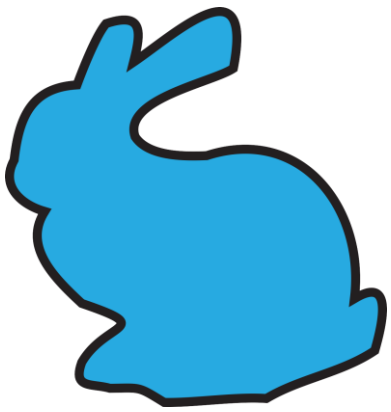
[Source: Wikipedia et al]

32

Signed Distance Functions as Geometry



$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$



??????????????? ???

[Source: Wikipedia et al]

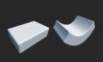
33

Boolean Operations

Twist

```
float opTwist( in sdf3d primitive, in vec3 p )
{
    const float k = 10.0; // or some other amount
    float c = cos(k*p.y);
    float s = sin(k*p.y);
    mat2  m = mat2(c,-s,s,c);
    vec3  q = vec3(m*p.xz,p.y);
    return primitive(q);
}
```

Bend



```
float opCheapBend( in sdf3d primitive, in vec3 p )
{
    const float k = 10.0; // or some other amount
    float c = cos(k*p.x);
    float s = sin(k*p.x);
    mat2  m = mat2(c,-s,s,c);
    vec3  q = vec3(m*p.xy,p.z);
    return primitive(q);
}
```

Finite Repetition

Infinite domain repetition is great, but sometimes you only need a few copies or instances of a given SDF, not infinite. A frequ
generate infinite copies and then clip the unwanted areas away with a box SDF. This is not ideal because the resulting SDF is
clipping through max() only produces a bound. A much better approach is to clamp the indices of the instances instead of the
the truncated/clamped indices. You can see this in action here as a 2D shader, although it works in any number of dimensions
https://www.shadertoy.com/view/3syGzz. Code:

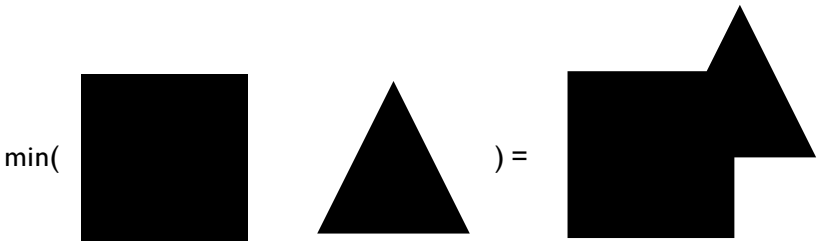
```
vec3 opRepLim( in vec3 p, in float c, in vec3 l, in sdf3d primitive )
{
    vec3 q = p-c*clamp(round(p/c),-l,l);
    return primitive( q );
}
```

This produces a rectangle of l.x times l.y instances. Naturally, you can create any rectangular shape you want (or other shape)
clamp function itself. Also note that the function can be specialized to 2 or 1 dimensions easily.

[Source: iquilezles.org]

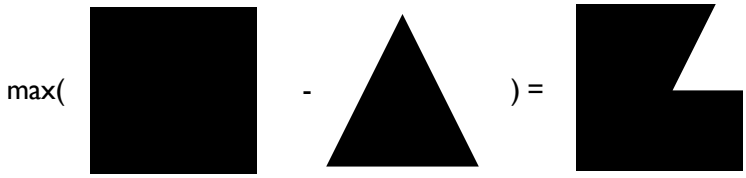
34

Boolean Operations

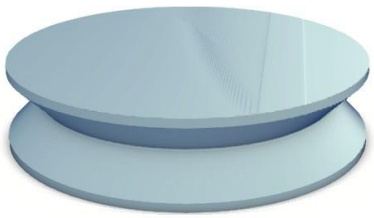


35

Boolean Operations



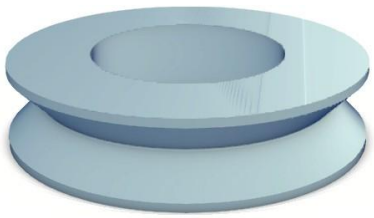
Constructive Solid Geometry



[Source: Takikawa et al]

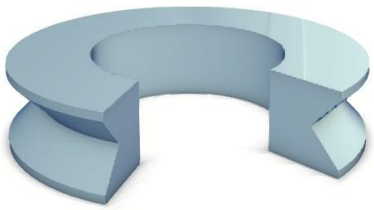
37

Constructive Solid Geometry



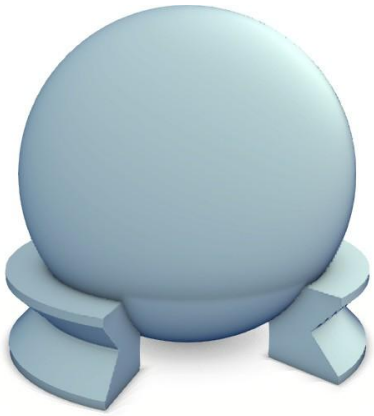
[Source: Takikawa et al]

Constructive Solid Geometry



[Source: Takikawa et al]

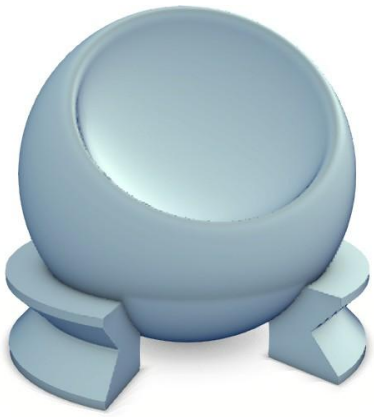
Constructive Solid Geometry



[Source: Takikawa et al]

40

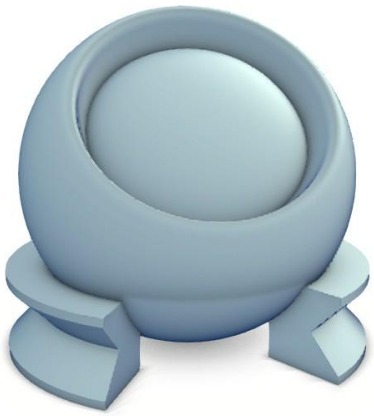
Constructive Solid Geometry



[Source: Takikawa et al]

41

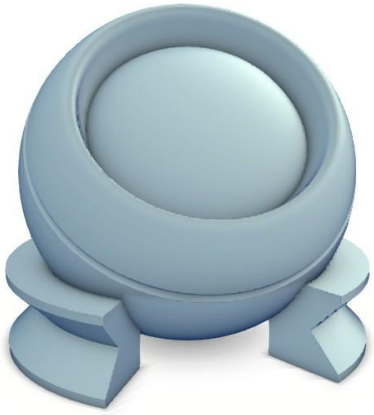
Constructive Solid Geometry



[Source: Takikawa et al]

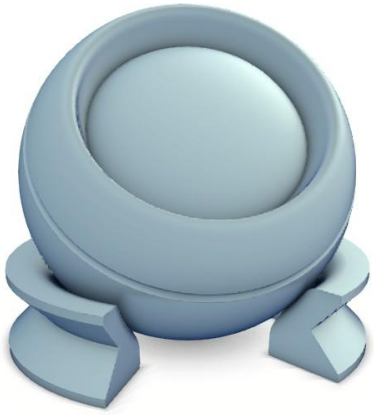
42

Constructive Solid Geometry



[Source: Takikawa et al]

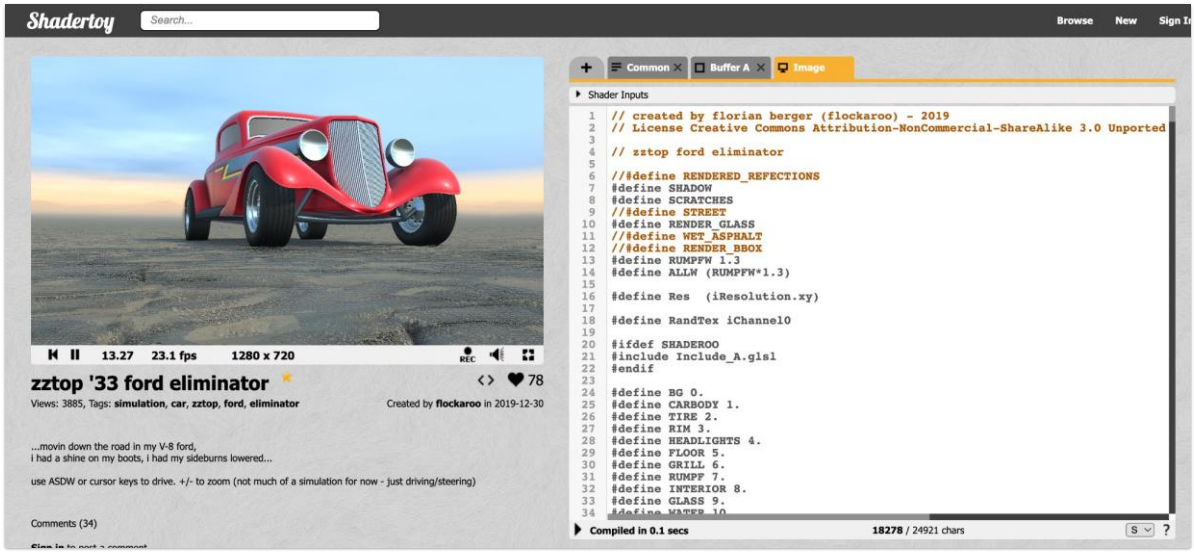
Constructive Solid Geometry



[Source: Takikawa et al]

44

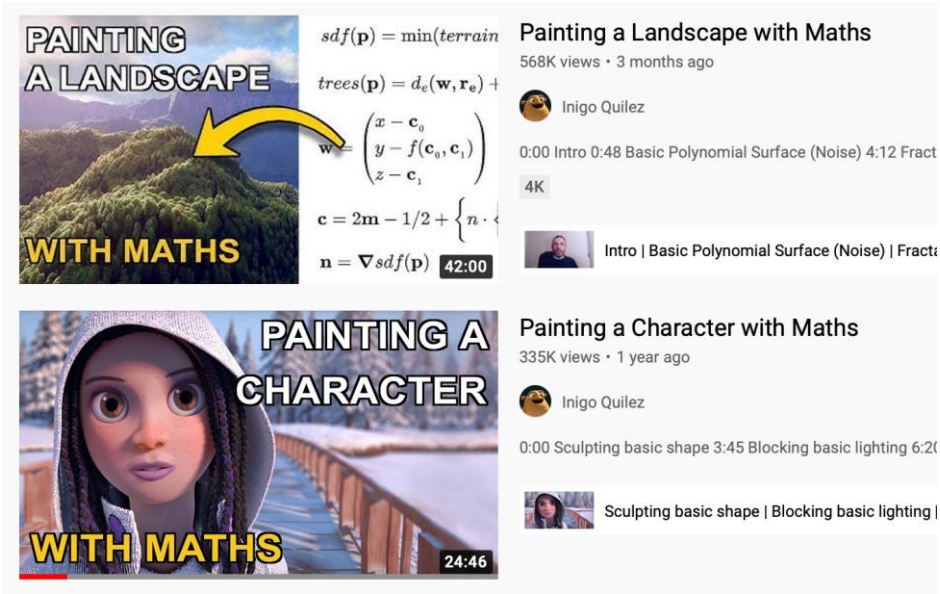
ShaderToy



[Source: shadertoy.com]

45

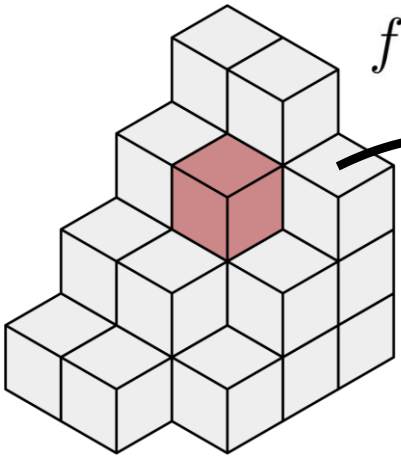
Modeling with SDFs



[Source: YouTube]

46

Voxel Grids



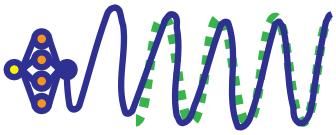
$$f(x, y, z) = d$$

Signed distance, color, density, etc…

[Source: Wikipedia]

47

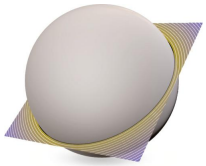
Fields and Signals



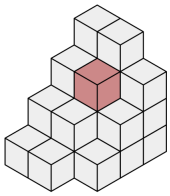
Neural (Fields)

Best of both worlds?

Smooth, compact, complex



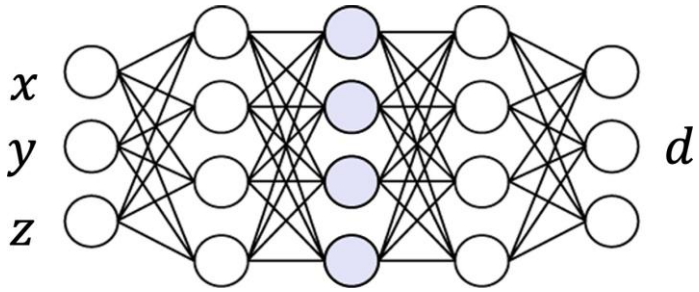
$$f(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$$



Simple, bulky, fast

48

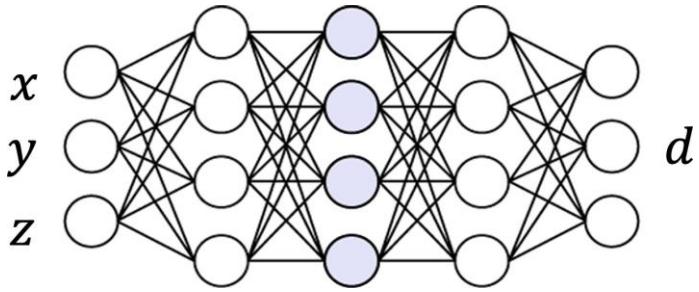
Multi-layer Perceptrons



$$f_\theta(x, y, z) = d$$

49

Multi-layer Perceptrons



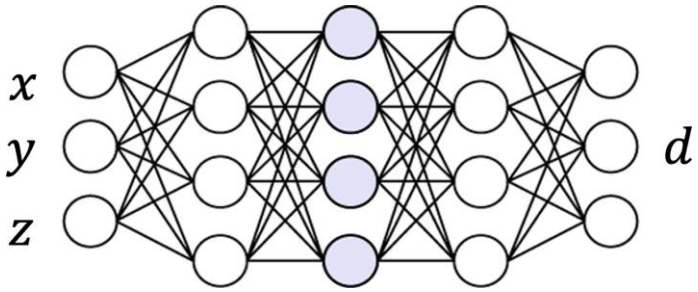
$$f_\theta(x, y, z) = d$$

In practice, can really be any parametric function!

50

Multi-layer Perceptrons

How do you obtain the parameters???



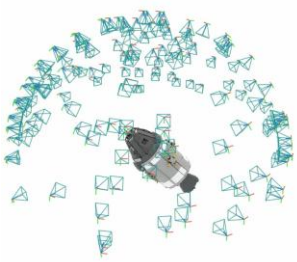
$$f_\theta(x, y, z) = d$$

In practice, can really be any parametric function!

51

Multi-layer Perceptrons

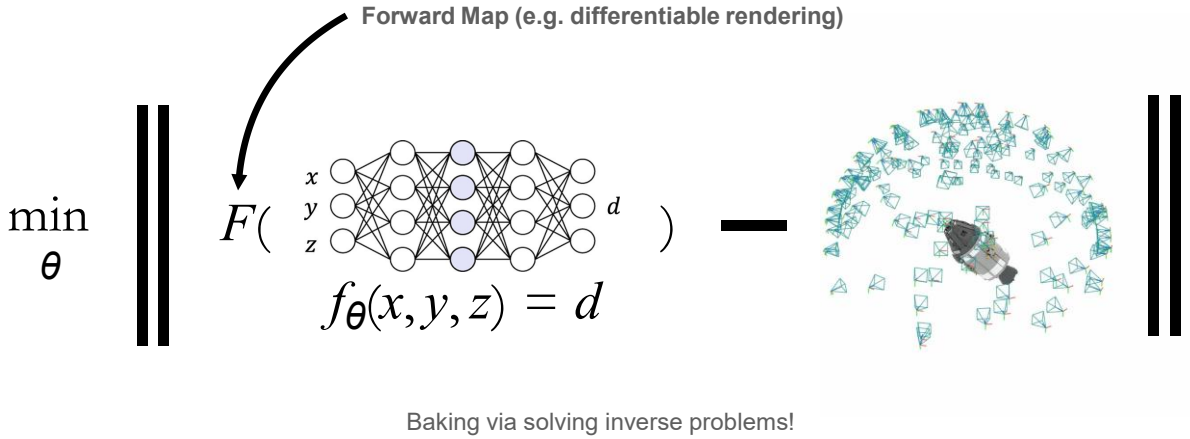
$$\min_{\theta} \left\| \begin{array}{c} x \\ y \\ z \end{array} \right. \quad d \quad - \quad \left. \right\|$$



$$f_{\theta}(x, y, z) = d$$

52

Multi-layer Perceptrons

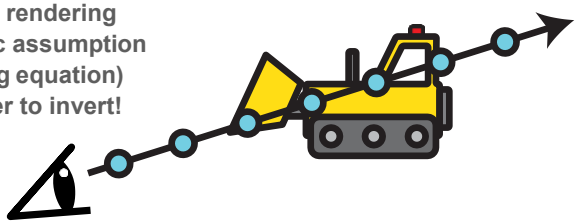


Baking via solving inverse problems!

53

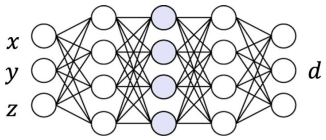
NeRF Innovation

NeRF uses volume rendering
(i.e. removes the Dirac assumption
in Kajiya's rendering equation)
to make things easier to invert!

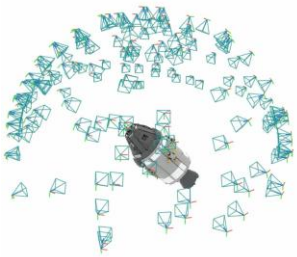


Forward Map (e.g. differentiable rendering)

$$\min_{\theta} \left\| F\left(\begin{matrix} x \\ y \\ z \end{matrix} \right) - \right\|$$



$$f_\theta(x,y,z) = d$$



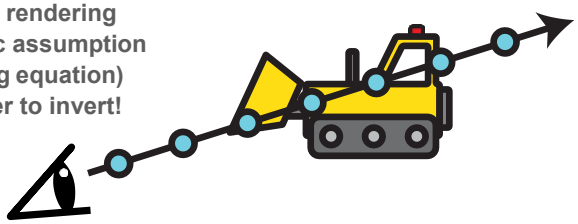
$$\left\|$$

Baking via solving inverse problems!

54

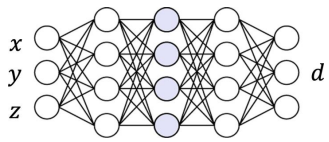
NeRF Innovation

NeRF uses volume rendering
(i.e. removes the Dirac assumption
in Kajiya's rendering equation)
to make things easier to invert!

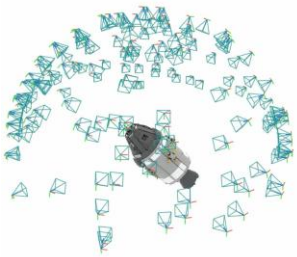


Forward Map (e.g. differentiable rendering)

$$\min_{\theta} \left\| F\left(\begin{matrix} x \\ y \\ z \end{matrix} \right) - \right\|$$



$$f_\theta(x,y,z) = d$$



$\| \quad \|$

Does this really need to be neural?

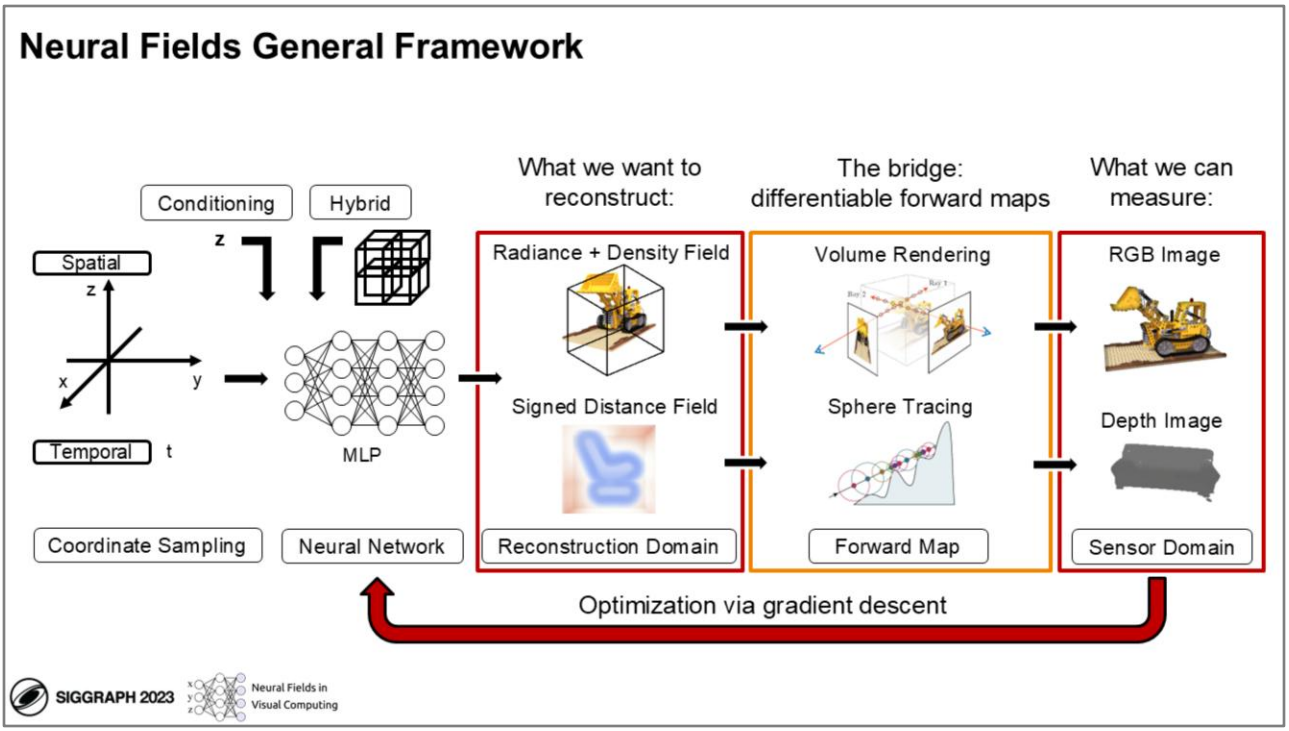
55

Neural Fields General Framework - Example

Neural radiance field with hybrid representation

Captured images → Processing → Rendering of real-world place

[Mildenhall et al., Neural Radiance Fields (NeRF), ECCV 2020]
[Wu et al., Scalable Neural Indoor Scene Rendering, SIGGRAPH 2022]



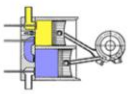
This is a natural place for us to introduce the general framework for using neural fields to solve inverse problems. We first sample * the coordinate, which are the inputs of the * neural network. The neural network predicts the * reconstructed signal. The reconstruction is then rendered * into what we can measure * with real-world sensors. In inverse problems, we are given sparse, noisy, and dimension-constrained information from * real-world sensors. However, we want to recover a * rich underlying representation from these limited observations of the real world. For example, how can we reconstruct a 3D geometry from sparse 2D images? We need a differentiable function mapping the reconstructed signal to sensor signals. * This is the differentiable forward map. In visual computing, this becomes a differentiable renderer. With that, we can now supervise * the learning of reconstructed signal, with limited sensor signals.

Strengths and Weaknesses of Neural Fields

In the course, we will highlight both **strengths** and **weaknesses** of neural fields, and what is missing for practical use. Some weaknesses have now been overcome with better understanding and engineering!

Strengths:

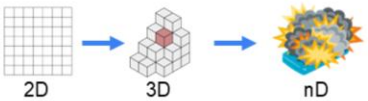
Compactness



Self-regularizing

$$\text{argmin}_x \|y - F(x)\| + \lambda P(x).$$

Domain agnostic



Weaknesses:

Computationally expensive

Not easily editable

Hard to model semantics and discrete data

Lack of theoretical understanding

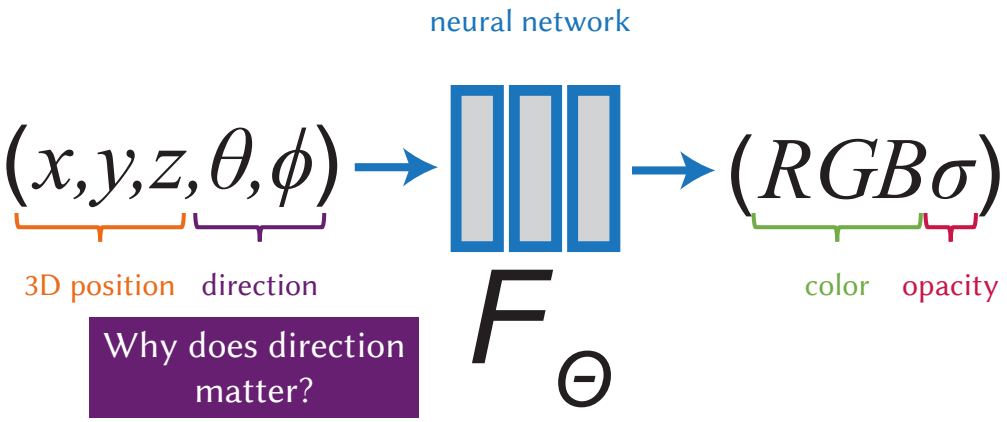
NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

NeRF

- https://www.matthewtancik.com/nerf
- https://arxiv.org/abs/2003.08934

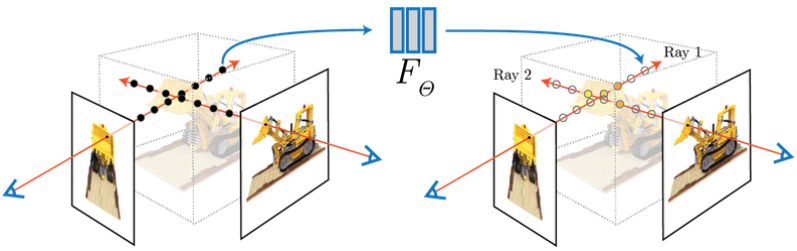
Architecture



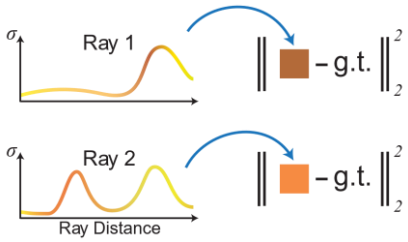
Why direction matters

Animation from https://pyimagesearch.com/2024/10/28/nerfs-explained-goodbye-photogrammetry/

Working principle of NeRF



differentiable volume rendering



loss function

Simplified volume rendering in NeRF

Computer Graphics Charles University

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Problem: It is fast, but not "real-time fast"

Computer Graphics Charles University

Solution: Gaussian splatting

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

Gaussian splatting

- https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/
- https://doi.org/10.1145/3592433

Principle of 3D Gaussians

Computer Graphics Charles University

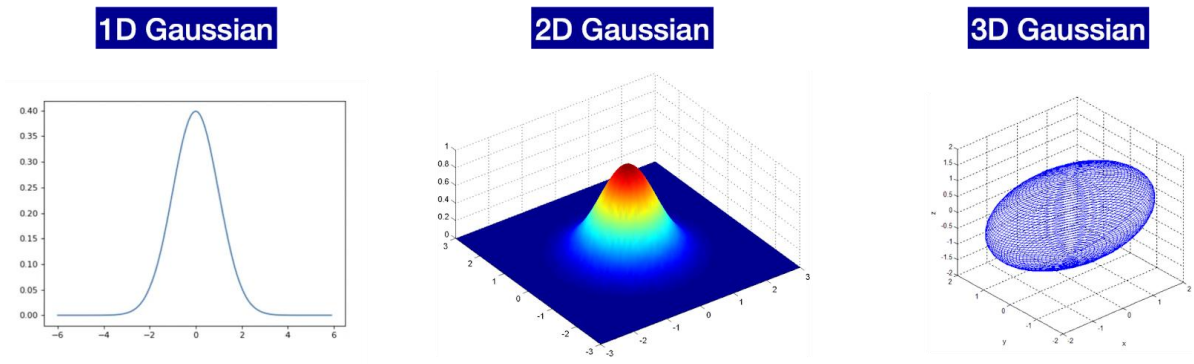


Image from https://pyimagesearch.com/2024/12/09/3d-gaussian-splatting-vs-nerf-the-end-game-of-3d-reconstruction/

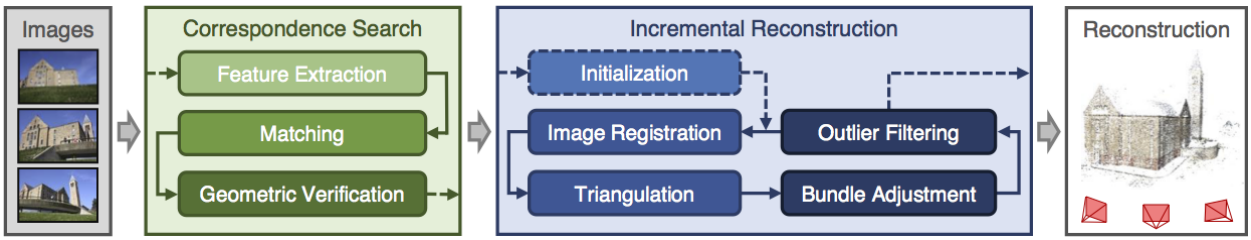
Lots of 3D Gaussians next to each other...

Computer Graphics Charles University

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

Step 1: Initialization

- Gaussian splatting does not do any initialization "by itself"
- Uses a structure from motion (SfM) algorithm such as COLMAP:



- SfM outputs a **point cloud**, which is then converted to 3D Gaussians

Step 2: Rendering

Computer Graphics Charles University

- Gaussian splatting is real-time because Gaussians can be **rasterized!**
- No need for volume rendering as in NeRF

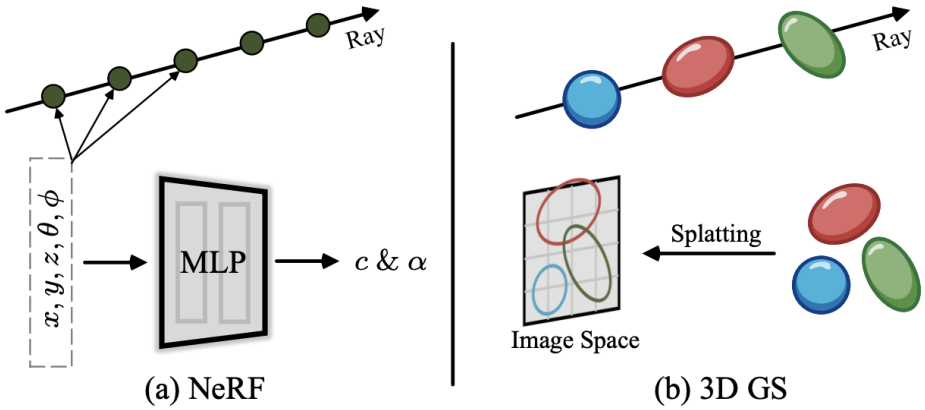


Image from https://arxiv.org/pdf/2401.03890

Step 2: Rendering

- Rasterization of Gaussians is easily parallelized

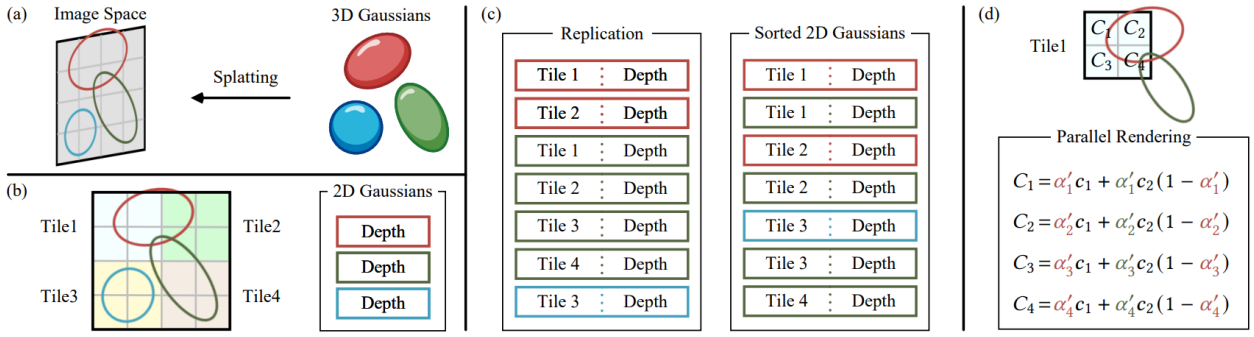


Image from https://arxiv.org/pdf/2401.03890

Step 3: Optimization

Computer Graphics Charles University

- Rendering Gaussians is trivially differentiable ⟶ gradient descent
- However, we should also perform **cloning** and **splitting**

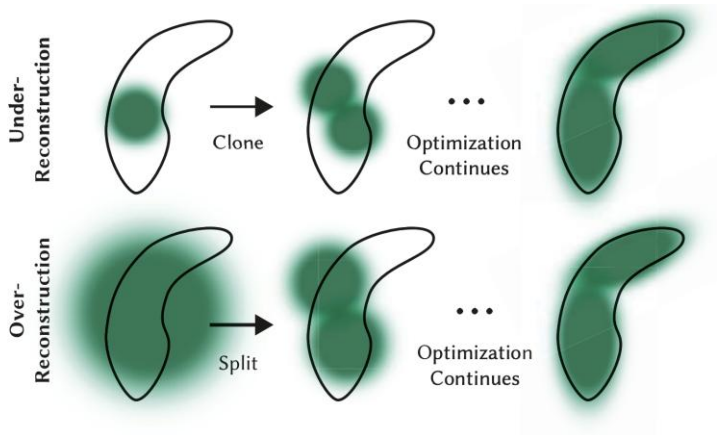
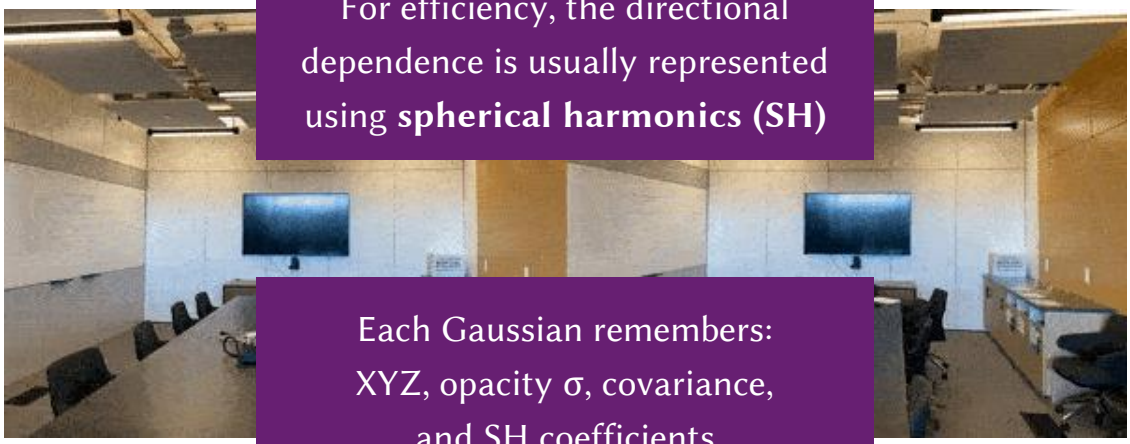


Image from the original Gaussian splatting paper

What about the directionality?

Computer Graphics Charles University



For efficiency, the directional dependence is usually represented using **spherical harmonics (SH)**

Each Gaussian remembers: XYZ, opacity σ, covariance, and SH coefficients

Animation from https://pyimagesearch.com/2024/10/28/nerfs-explained-goodbye-photogrammetry/

Computer Graphics Charles University

Advanced 3D Graphics for Movies and Games

Lecture 11: Neural fields and Gaussian splatting

NPGR010
Winter Semester 2025/2026
Taught by Tomáš Iser