Content Generation (2): Lightfields and NERFs, Splatting, Differential Rendering, Diffusion Models

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University

Content Generation

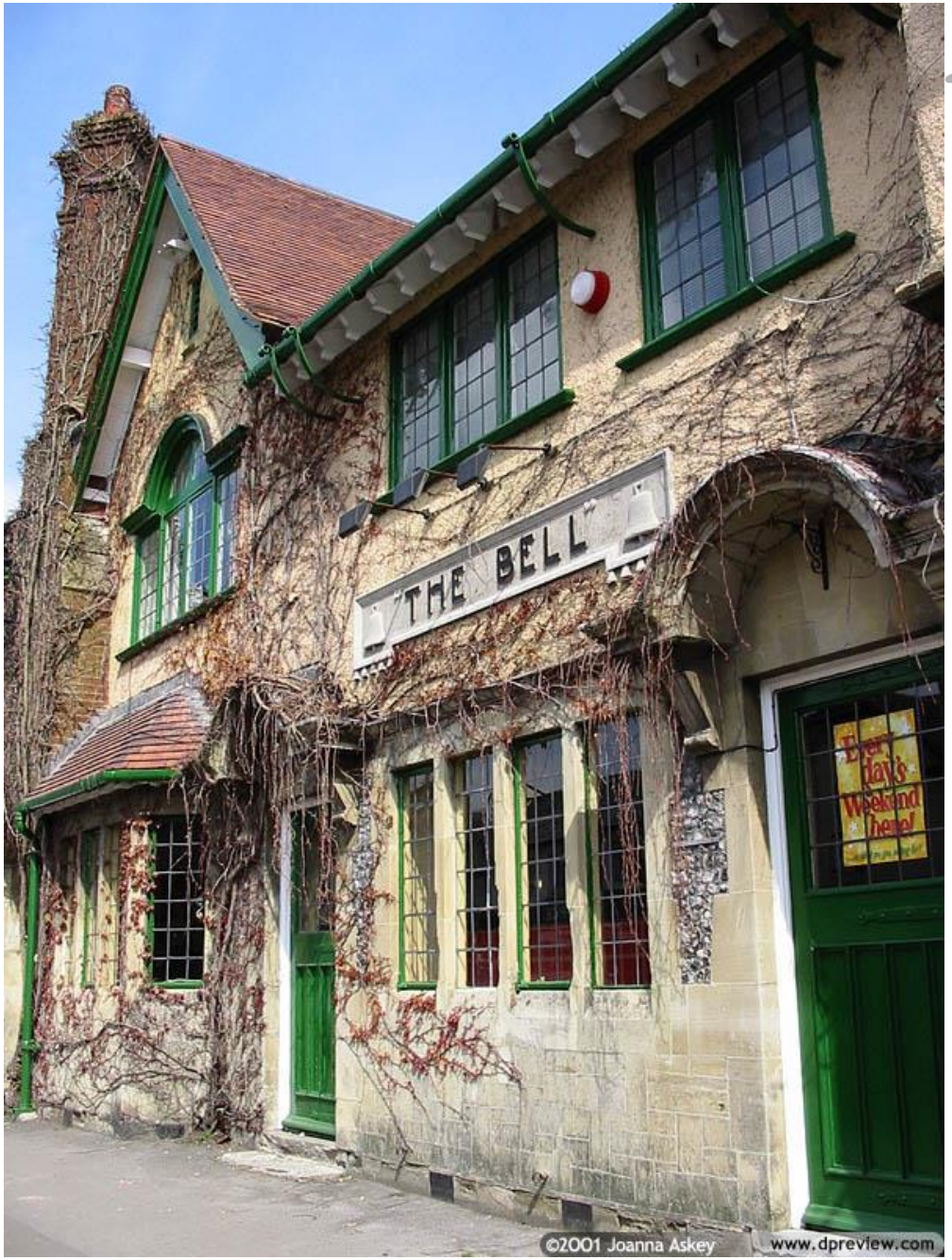
- Lightfields and NERFs
- Splatting
- Differential Rendering
- Diffusion Models

Lightfields and NERFs

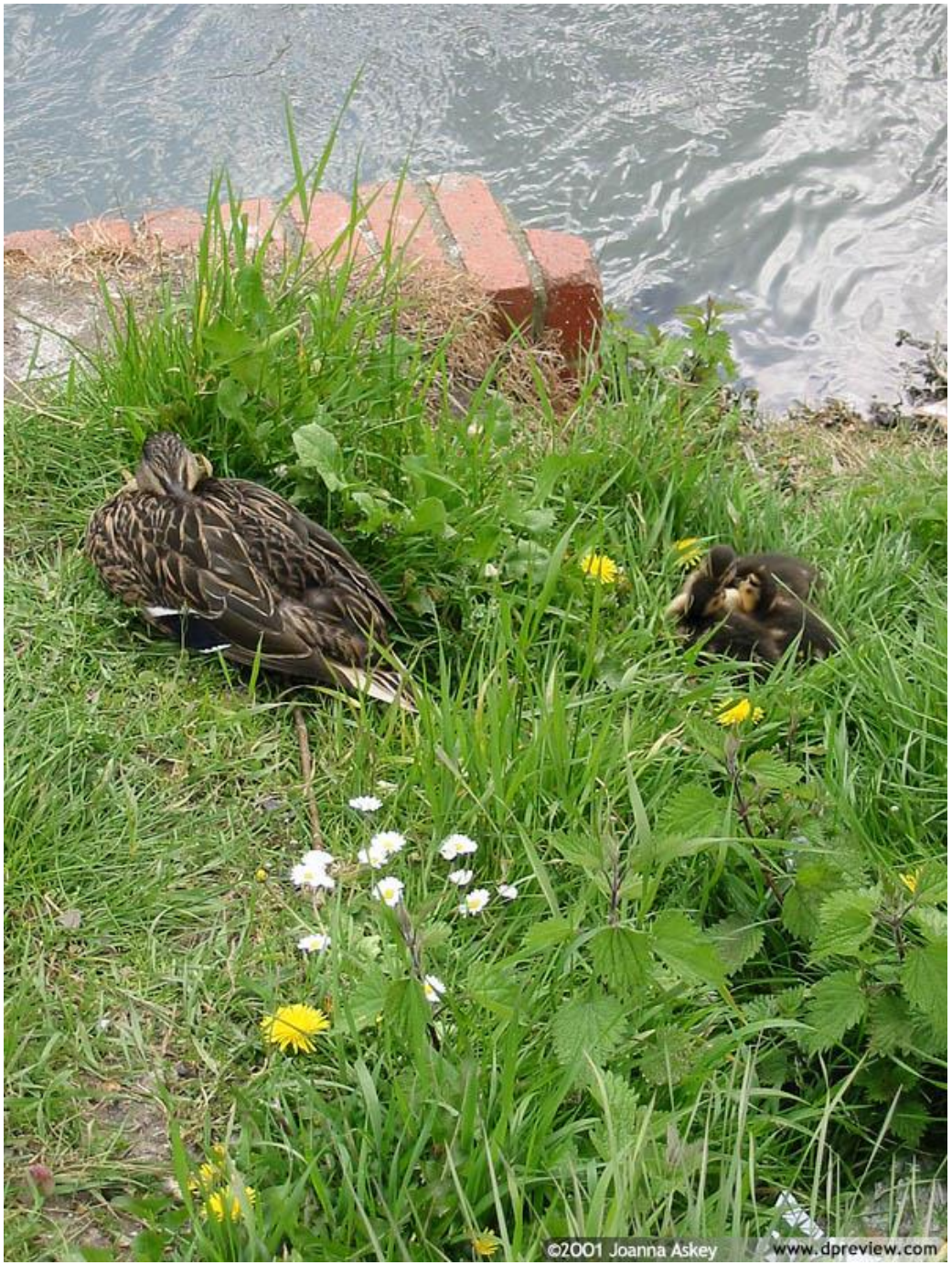
CS535
Daniel G. Aliaga
Department of Computer Science
Purdue University

Photographs

- We have tools that acquire and tools that display photographs at a convincing quality level


THE BELL
Every day's Weekend here!
©2001 Joanna Askey
www.dpreview.com


TIME
©2001 Phil Askey
www.dpreview.com


©2001 Joanna Askey
www.dpreview.com

Online

- 24 MP:
  - https://www.flickr.com/photos/markgaler/39926271622
- Gigapixel:
  - https://360gigapixels.com/360-paris-skyline-gigapixel-photo/

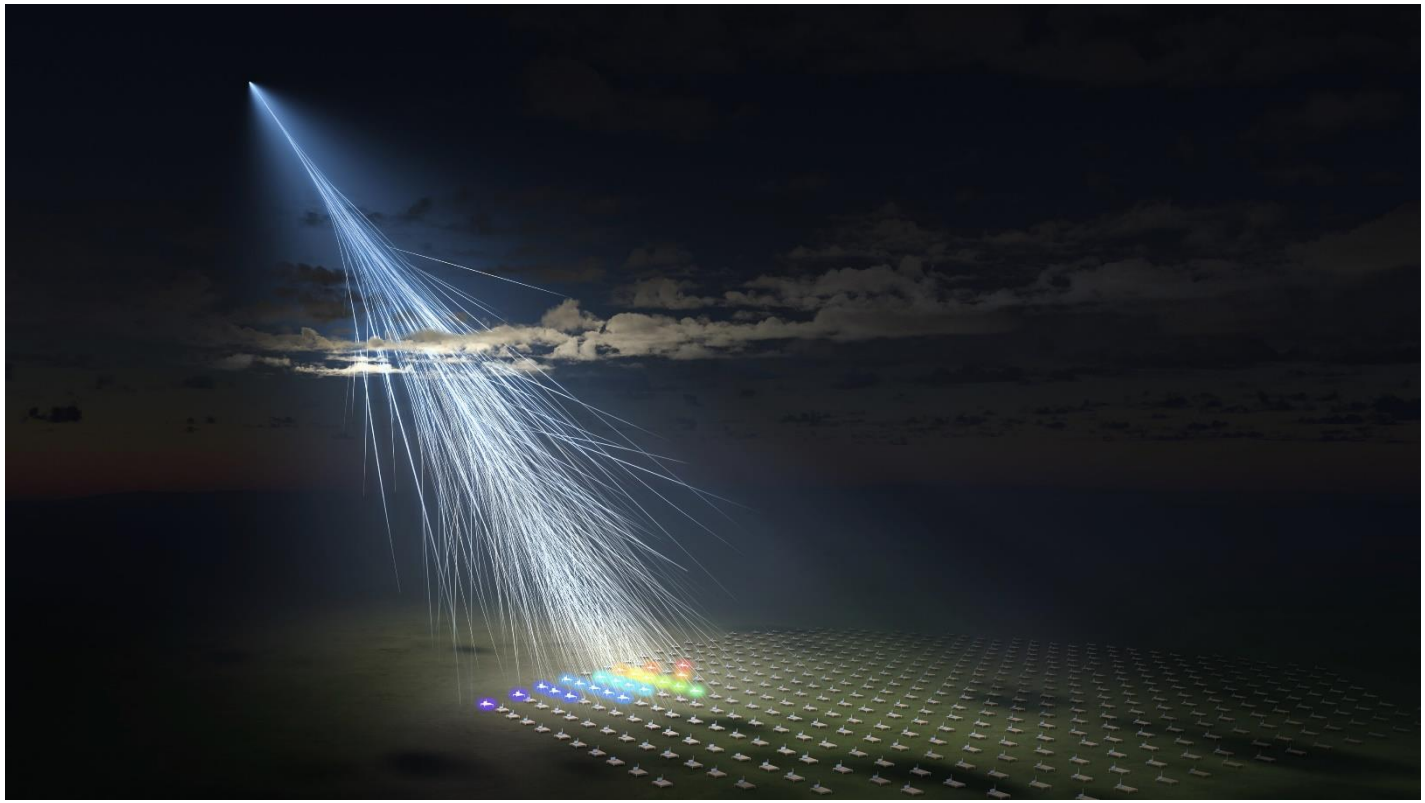
Plenoptic Function

- $P(x, y, z, \phi, \varphi, \lambda, t)$
  - 7D function to describe light intensity passing through every viewpoint, for every direction, for every wavelength, and for every time instant

Plenoptic Function

- “Holodeck” (Star Trek)
- Layered Depth Images
- 3D Image Warping
- View Interpolation
- (Sea of Images)
- Lightfield/Lumigraph
- (Plenoptic Stitching)
- Concentric Mosaics
- Panoramic Images

What is a “light ray through space”?



Lines, rays, bundles of them, line is 2D+2D, ray is 3D+2D

Light Ray Organization

- Surface-centric
- Viewpoint-centric

or

- Inside-looking-out
- Outside-looking-in

Draw different parameterizations

Reducing Dimensions of the Plenoptic Function

- Use constant frequencies
- Use static environments
- Use open spaces

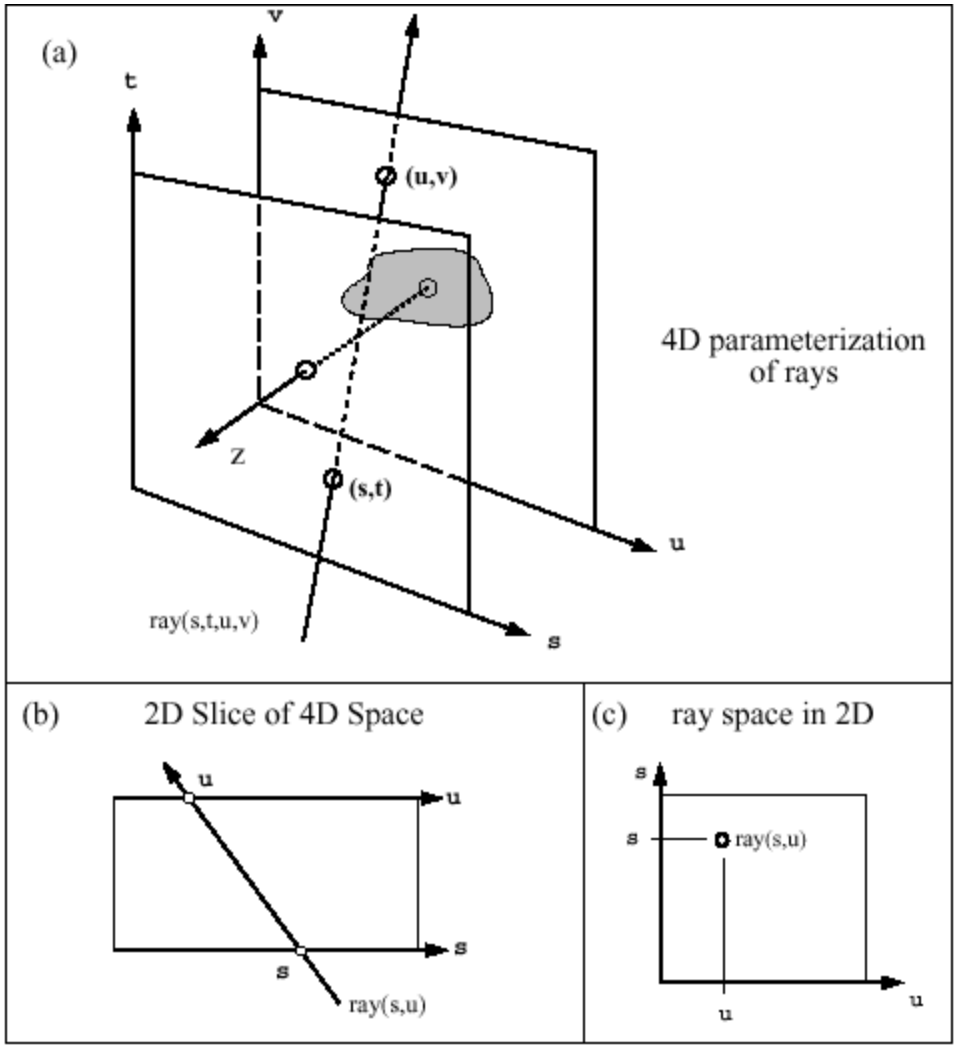
End-up with two slab-like pic

Light Ray Parameterization

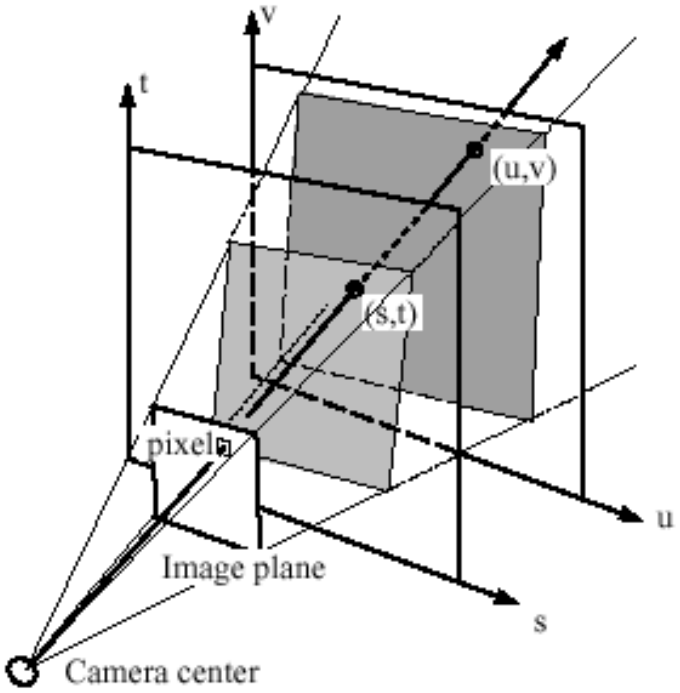
- Random collection of rays
- Two slab representation (s,t,u,v)
- Box representation

Draw each paramet.

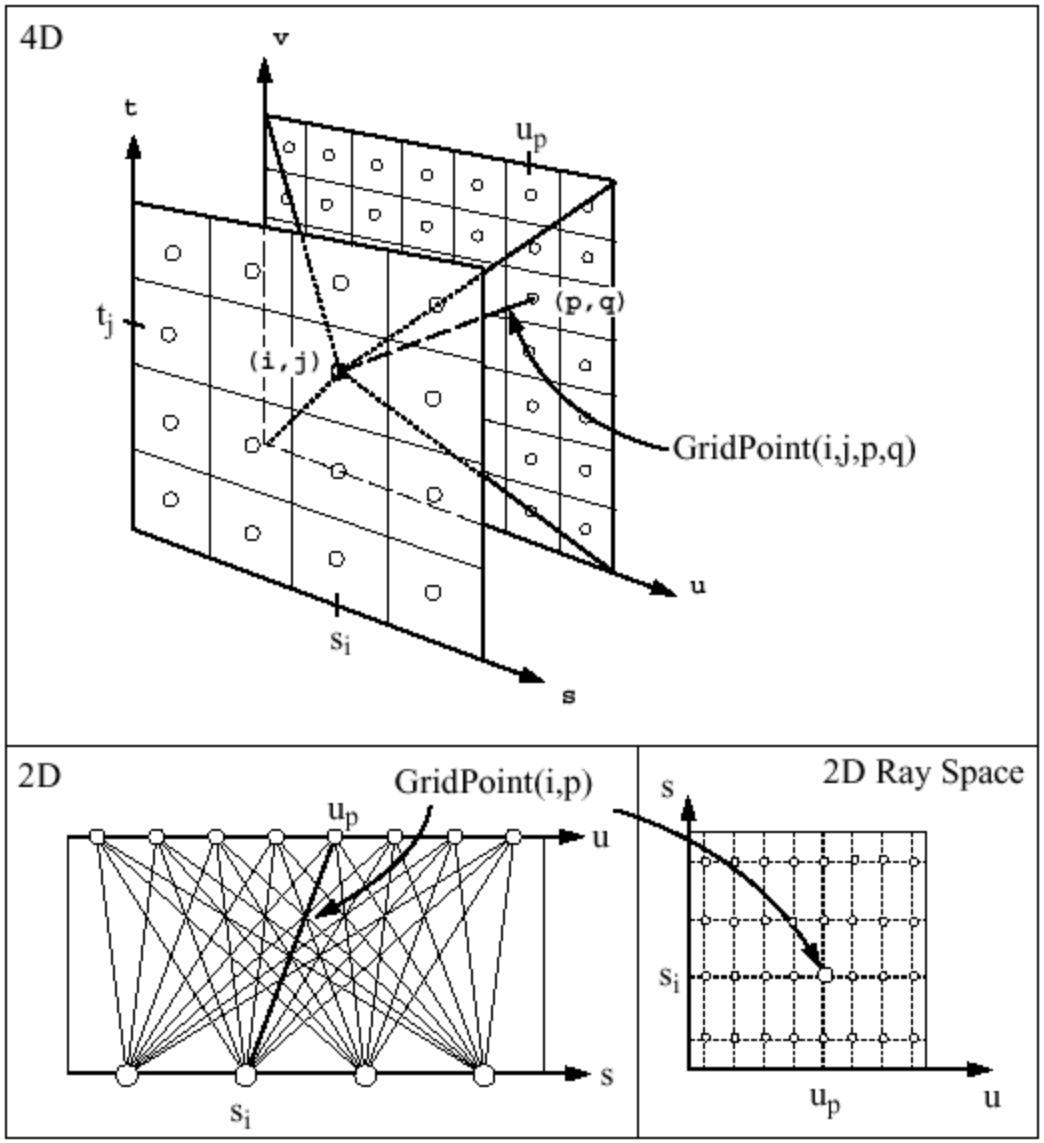
4D Lightfield / Lumigraph



4D Lightfield / Lumigraph

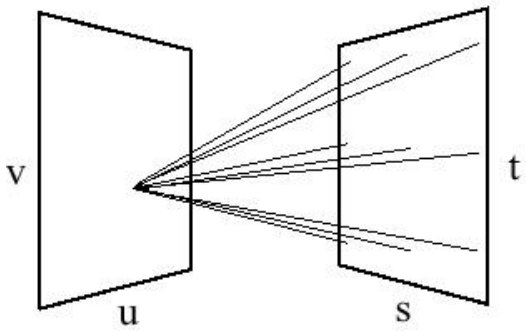


Discreet 4D Lightfield

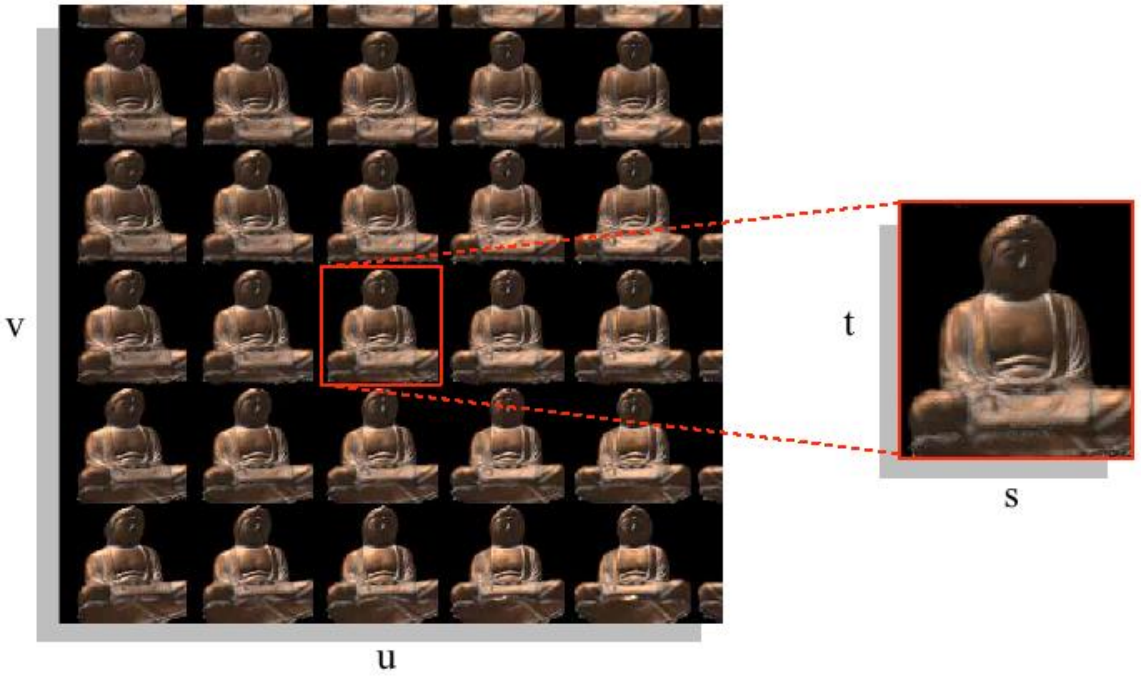


Lightfield

- Set of images with COPs on regular grid

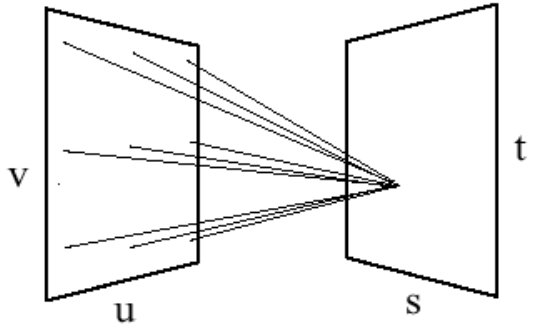


(a)

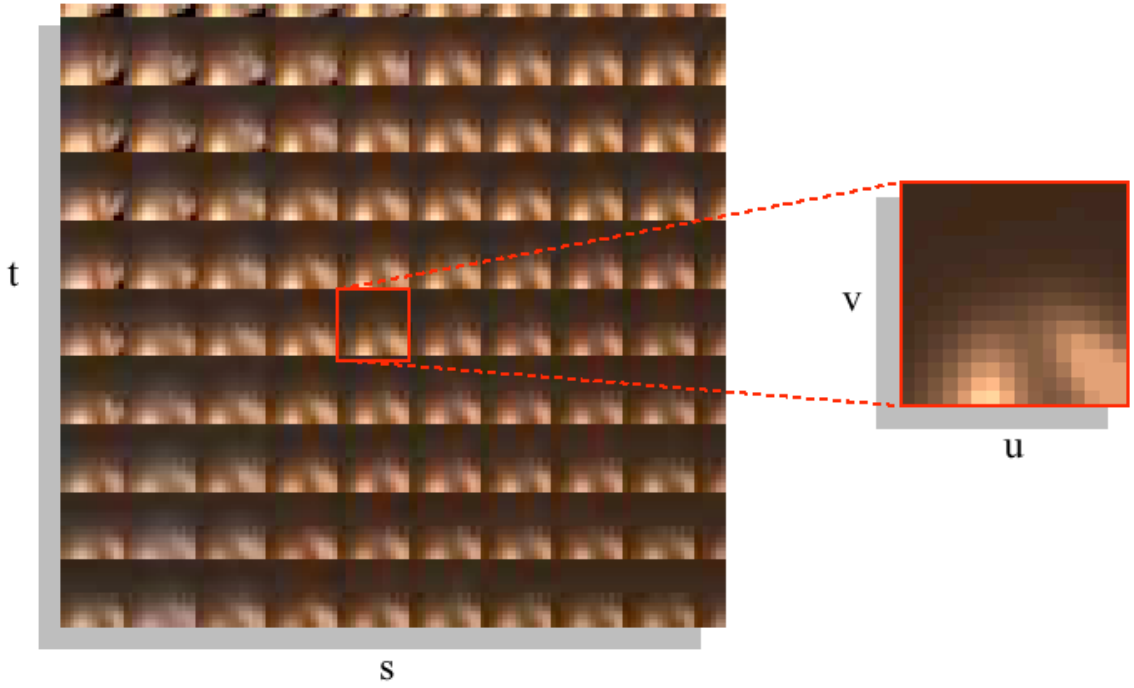


Lightfield

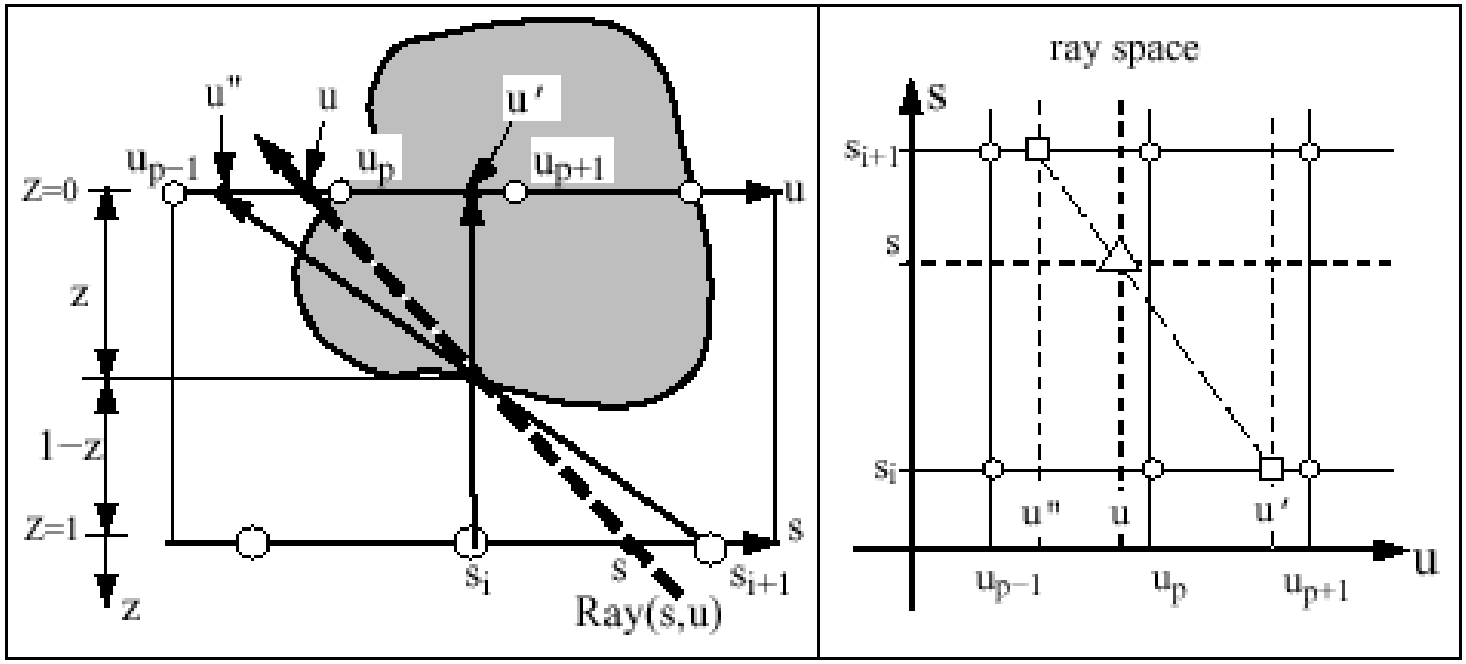
- Set of images of a point seen at various angles



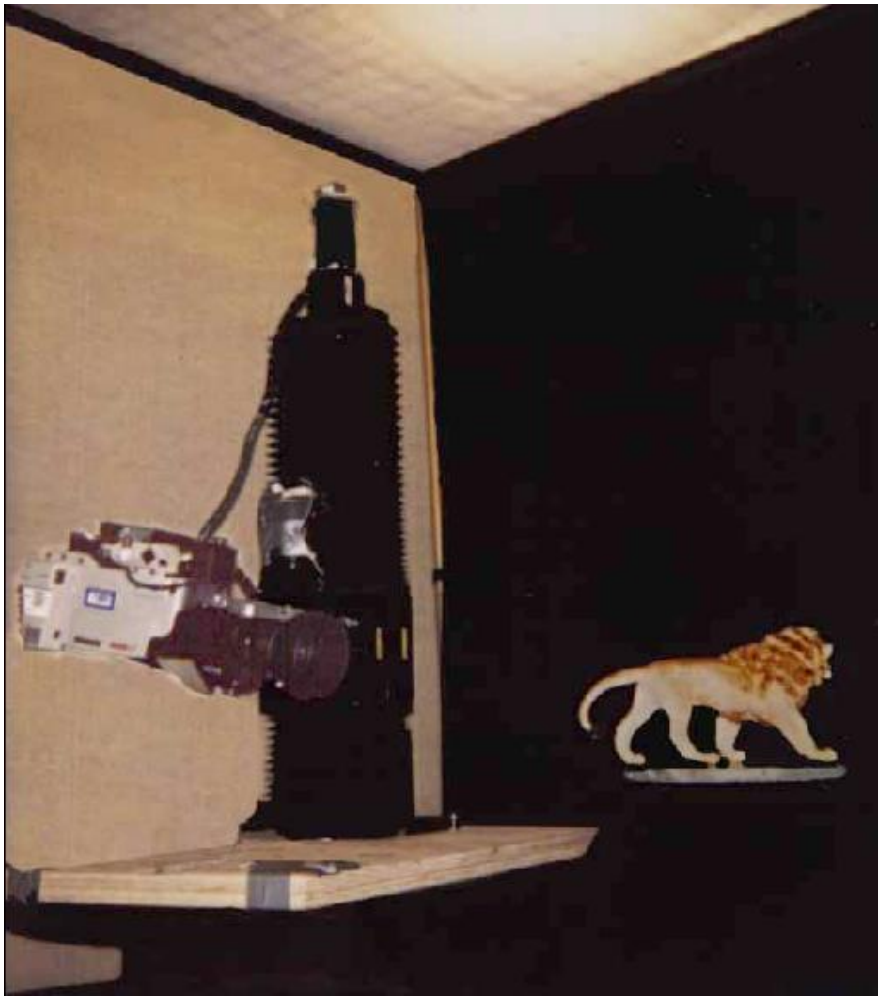
(b)

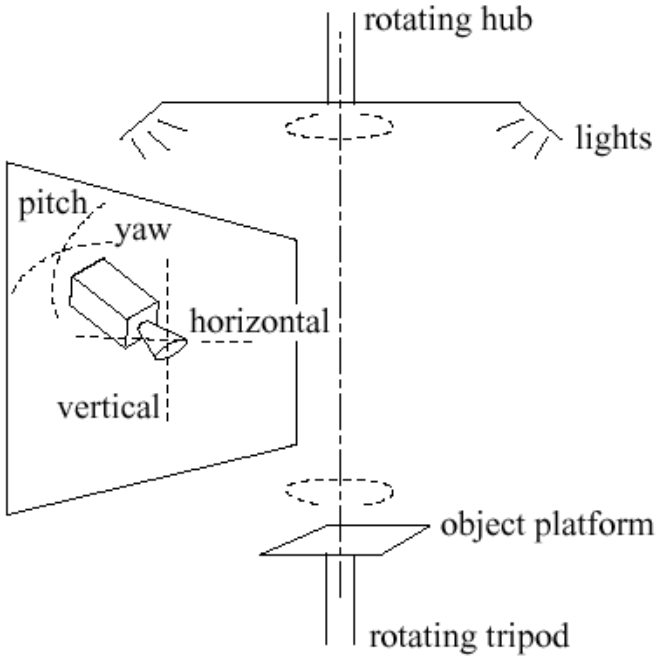


Depth Correction of Rays

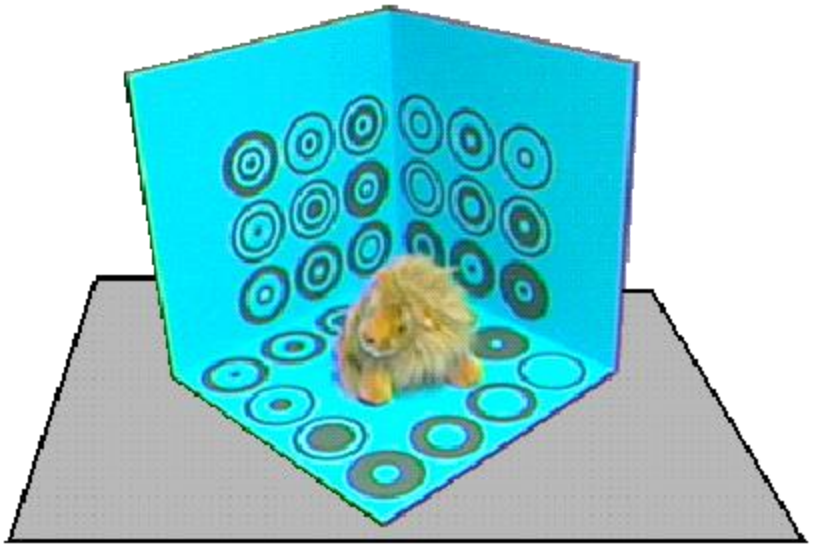


Capture a dense set of photographs



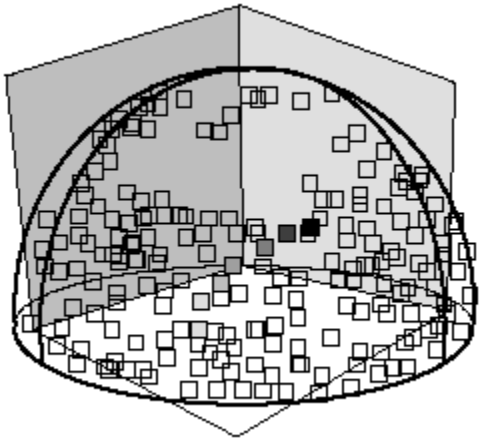


Capturing a sparse set of photographs



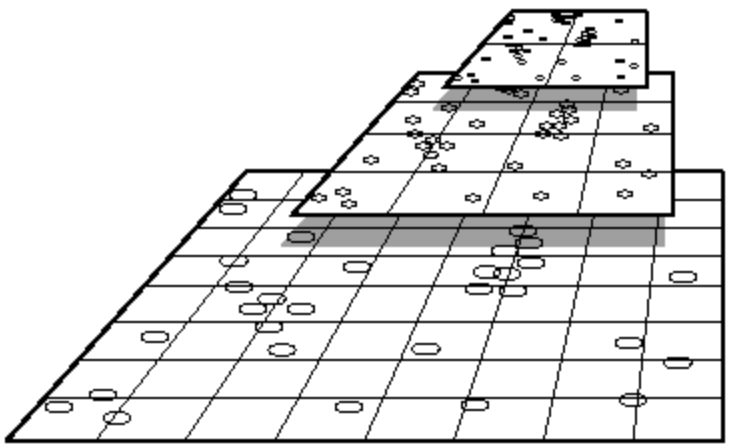
acquisition stage

camera positions



blue screening

Filling in gaps using pull-push algorithm



- Pull phase
  - low res levels are created
  - gaps are shrunk
- Push phase
  - gaps at high res levels are filled using low res levels

Acquiring a 4D Lightfield/Lumigraph

- Capture (many images)
- Organize into a (s,t,u,v) parameterization
  - Do not “need” to resample the pixels
  - Use (linear) interpolation to extract an arbitrary ray/line
  - Optionally compress/decompress data
  - Interactively extract rays/lines to create a visual representation

Limitations of a Lightfield/Lumigraph

- Resolution
- High storage requirement
- Difficult capture (?)
- No geometry
  - Cannot add new geometry and (easily) do occlusion and re-illumination

Demos and Videos

- Google, AR/VR, and Lightfields:
  - https://www.youtube.com/watch?v=lRK0MtIyj0U
- Seeing through things with lightfields:
  - http://graphics.stanford.edu/papers/plane+parallax_calib/
- Microscope Lightfields
  - http://graphics.stanford.edu/projects/lfmicroscope/
- Stanford New Lightfield Archive
  - http://graphics.stanford.edu/data/LF/lfs.html
    - e.g.,"http://graphics.stanford.edu/data/LF/chess_lf/preview.zip&zoom=1"
  - Old: http://graphics.stanford.edu/software/lightpack/lifs.html

Deep Learning Lightfields

- Learning-Based View Synthesis for Light Field Cameras
  - https://www.youtube.com/watch?v=RCD2B5o1K8U
- Light Field Video Capture Using a Learning-Based Hybrid Imaging System
  - https://www.youtube.com/watch?v=TqVKcssYfAo

NeRF: representing scenes as neural radiance fields for view synthesis

by Mildenhall et al. 2021

- Scene is a 5D vector-valued function

  3D location + 2D direction → RGB color + density α

- Train MLP to produce this 5D (plenoptic) function

NeRF: representing scenes as neural radiance fields for view synthesis

by Mildenhall et al. 2021

- Given RGB posed images, predict color and density for every viewing location and direction:
  - Scene is a 5D vector-valued function:

    3D location + 2D direction → RGB color + density α
- Training
  - Use calibrated image set and randomly sample along each ray to train MLP and reproduce this 5D (plenoptic) function
- Inferencing
  - Repeat above but obtain RGB + density

NERF

Volume rendering estimation: integrating color along a ray

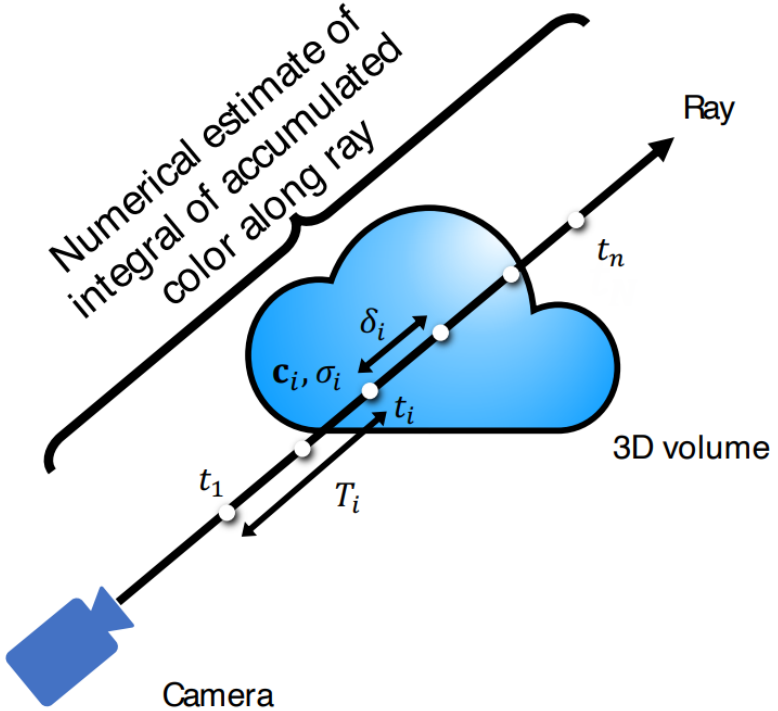
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

weights

colors



How much light is blocked earlier along ray:

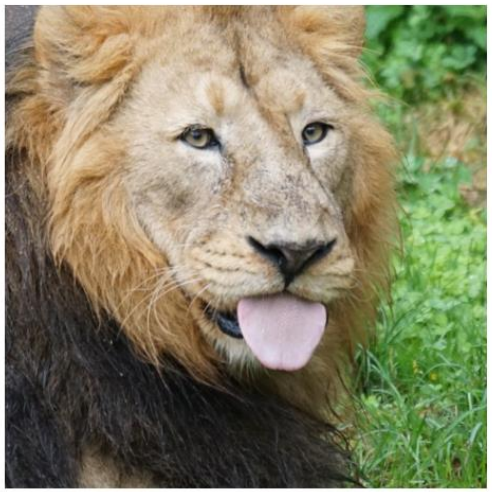
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$

[Ronisen, 2023]

MLP

- Just train $\{x, y, z, \theta, \phi)$-> $\{r, g, b, \alpha\}$
  - High frequencies lost…
- Toy example:

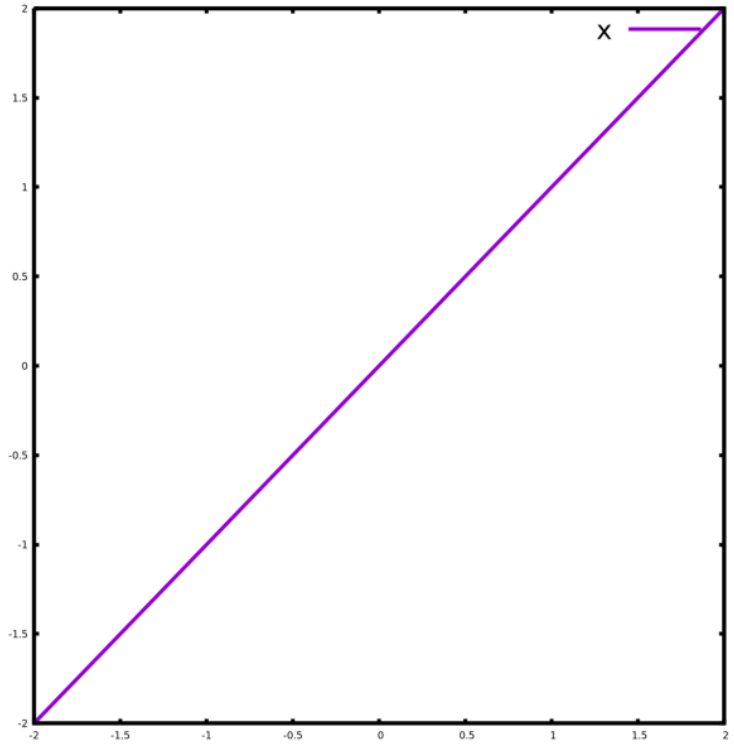


Ground truth image

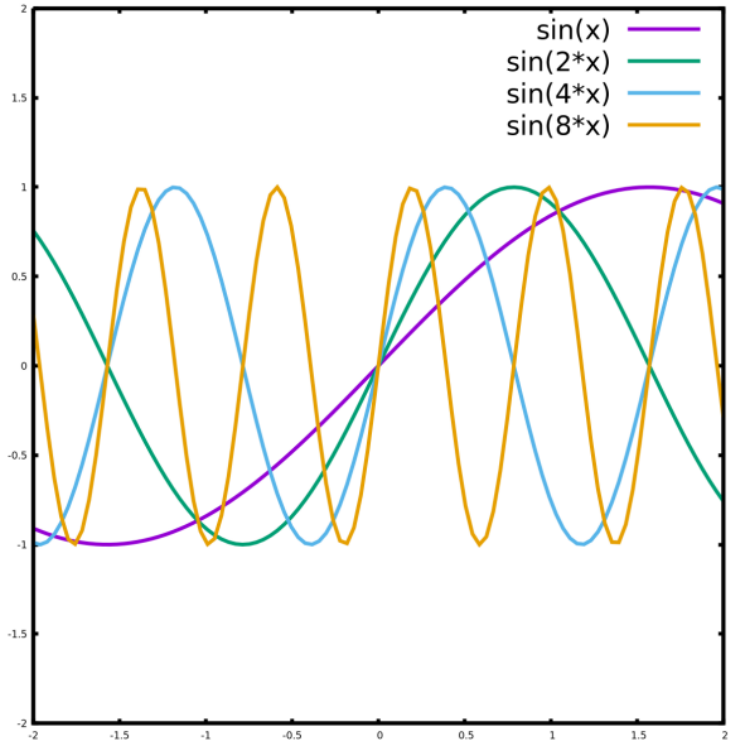
Neural network output without high frequency mapping

(fails!)

Positional Encoding



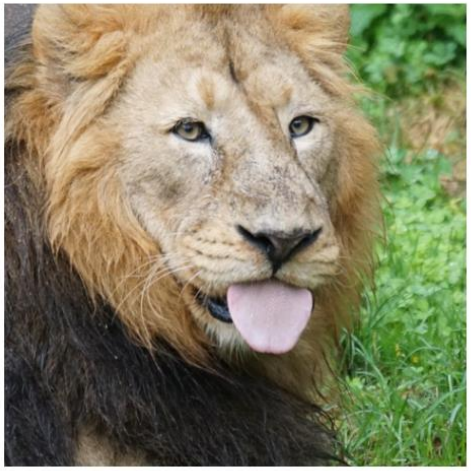
Raw encoding of a number ***x***



"Positional encoding" of a number ***x***

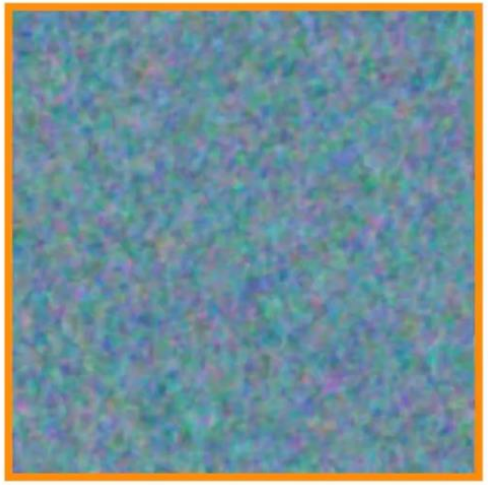
MLP

- Just train $\{x, y, z, \theta, \phi)$-> $\{r, g, b, \alpha\}$
  - High frequencies lost…
- Toy example:



Ground truth image

Neural network output without high frequency mapping



Neural network output with high frequency mapping

(works!)

NERF Lightfields

Instant NERF

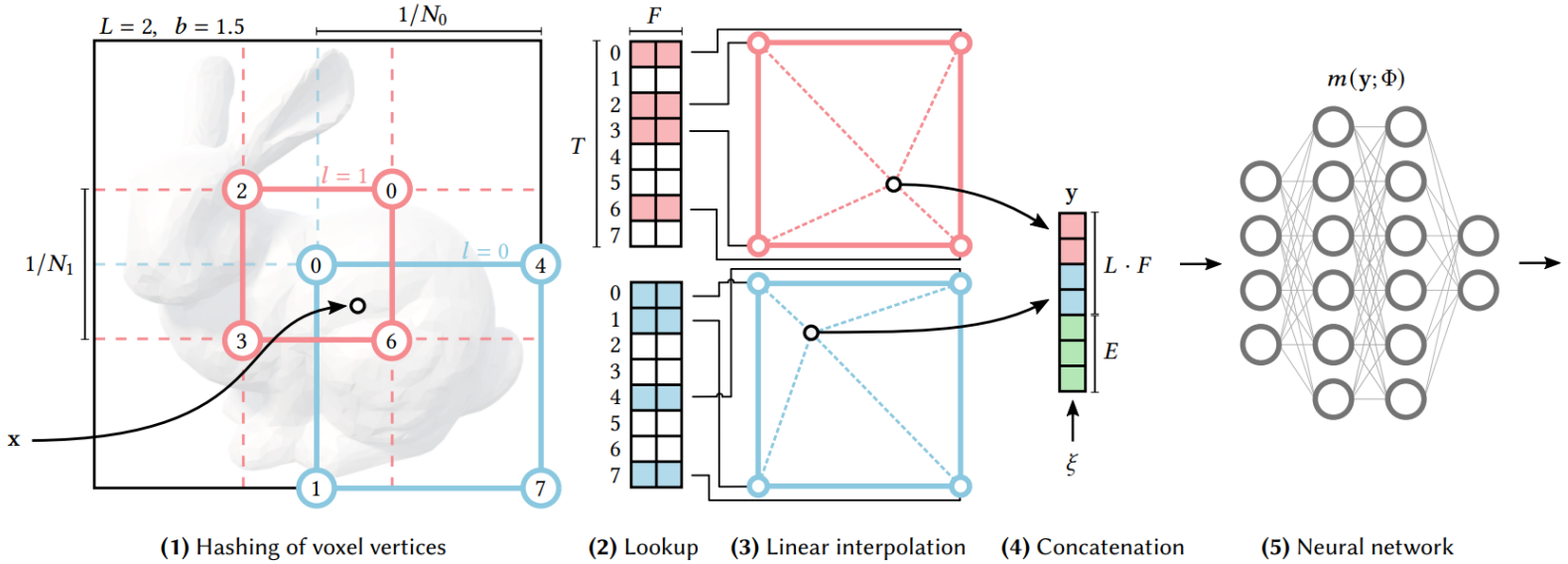


Fig. 3. Illustration of the multiresolution hash encoding in 2D. **(1)** for a given input coordinate $\mathbf{x}$, we find the surrounding voxels at L resolution levels and assign indices to their corners by hashing their integer coordinates. **(2)** for all resulting corner indices, we look up the corresponding F-dimensional feature vectors from the hash tables θ_l and **(3)** linearly interpolate them according to the relative position of $\mathbf{x}$ within the respective l-th voxel. **(4)** we concatenate the result of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$, producing the encoded MLP input $y \in \mathbb{R}^{LF+E}$, which **(5)** is evaluated last. To train the encoding, loss gradients are backpropagated through the MLP **(5)**, the concatenation **(4)**, the linear interpolation **(3)**, and then accumulated in the looked-up feature vectors.

- https://blogs.nvidia.com/blog/instant-nerf-research-3d-ai/

StyleNERF

- NERF + GAN

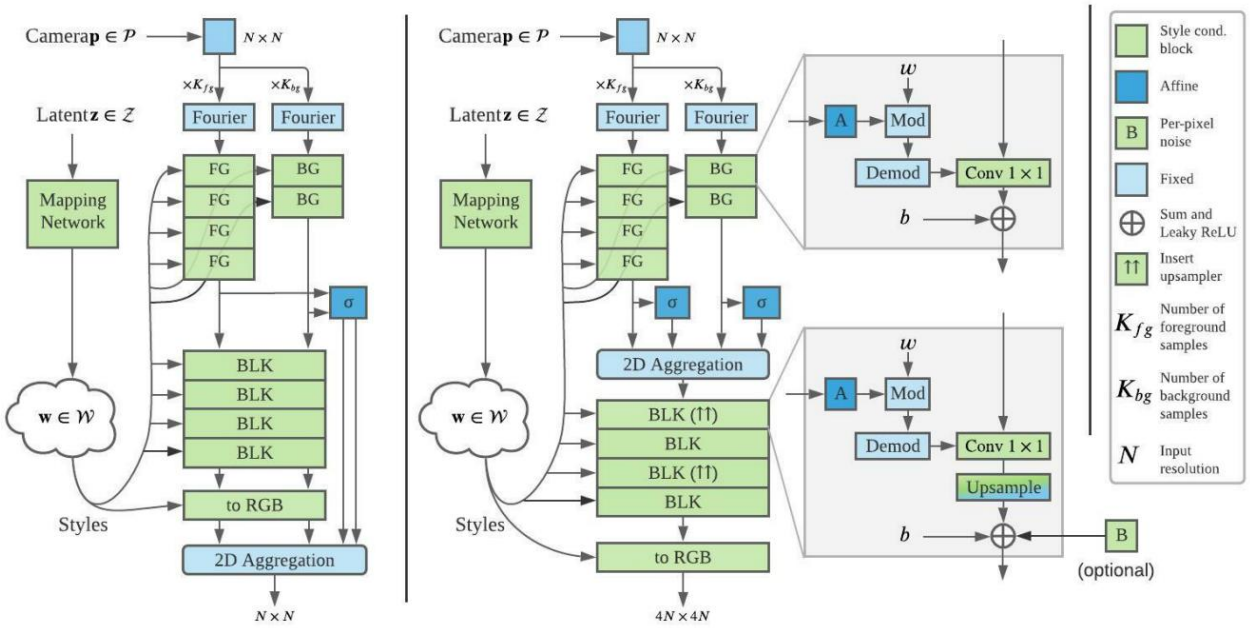


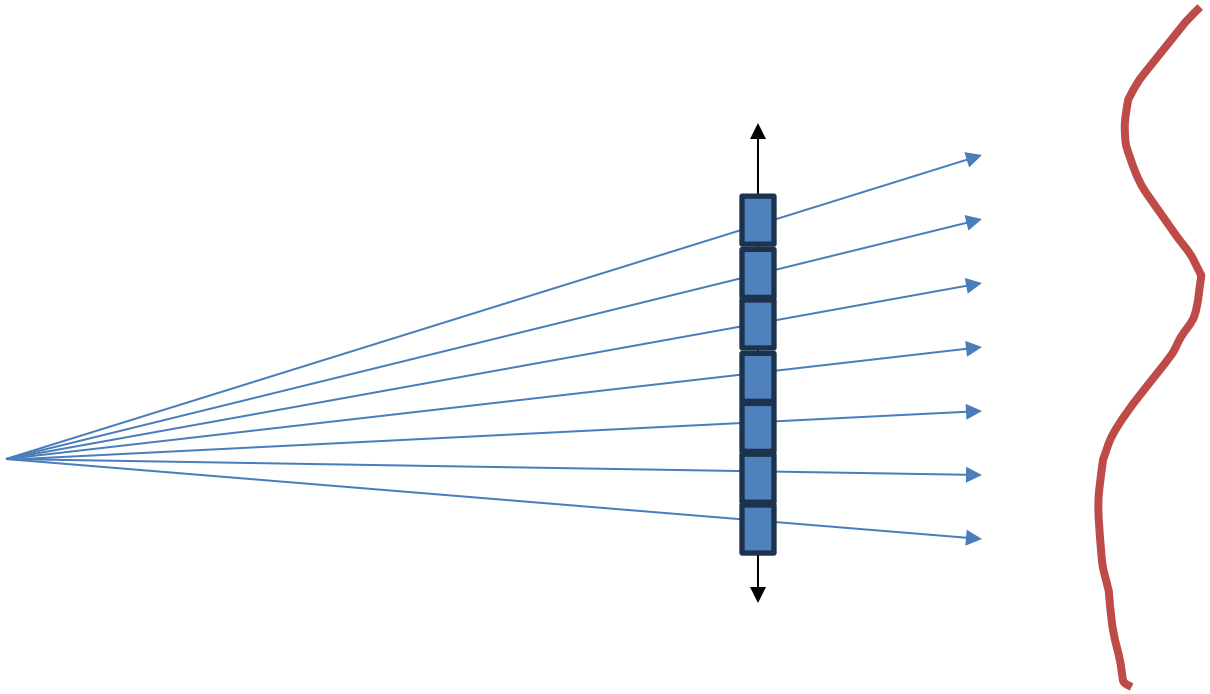
Figure 3: Example architecture. (Left) Original NeRF path; (right) Main path of StyleNeRF.

https://jiataogu.me/style_nerf

Content Generation

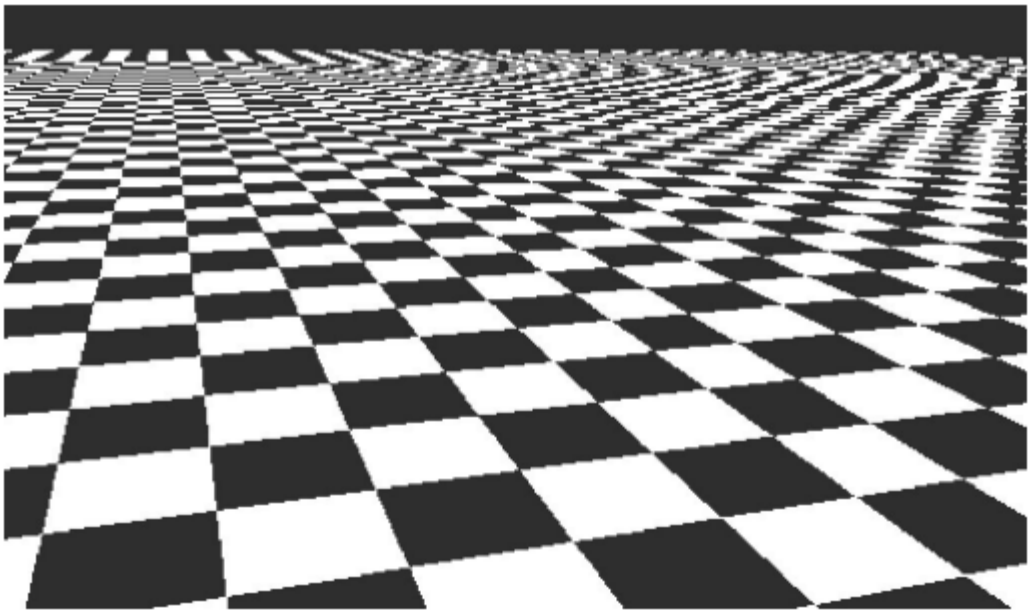
- Lightfields and NERFs
- Splatting
- Differential Rendering
- Diffusion Models

Problem

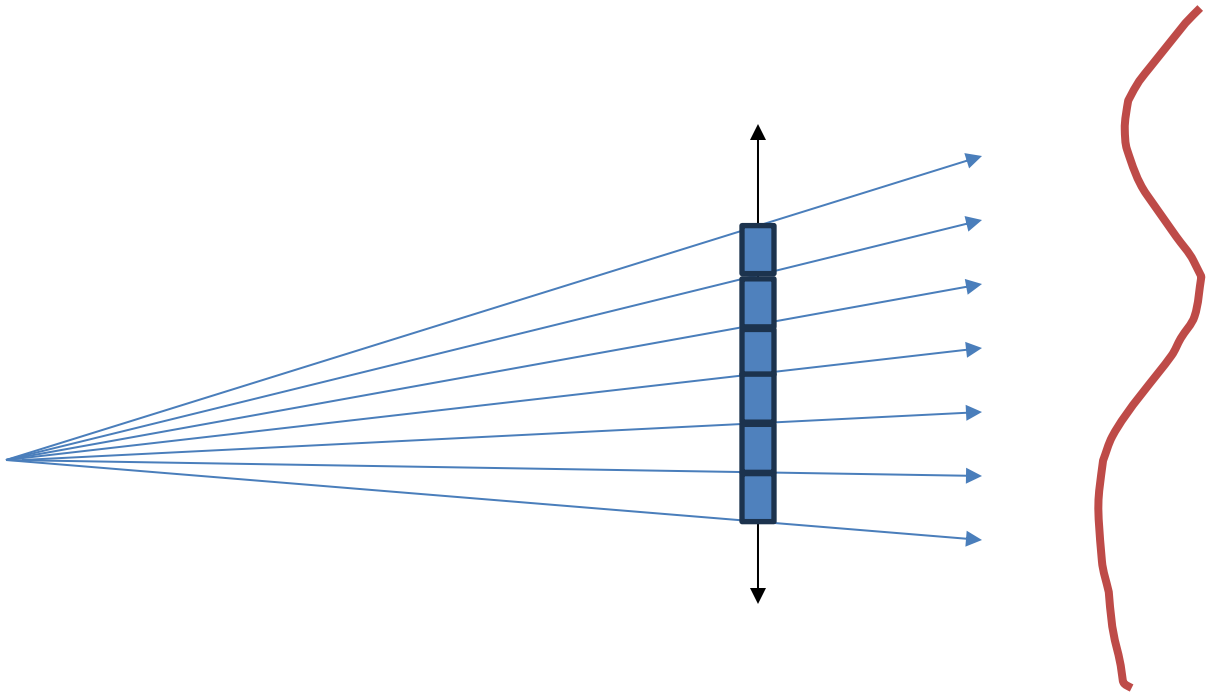


With what do we fill in the pixel?

Problem: aliasing?

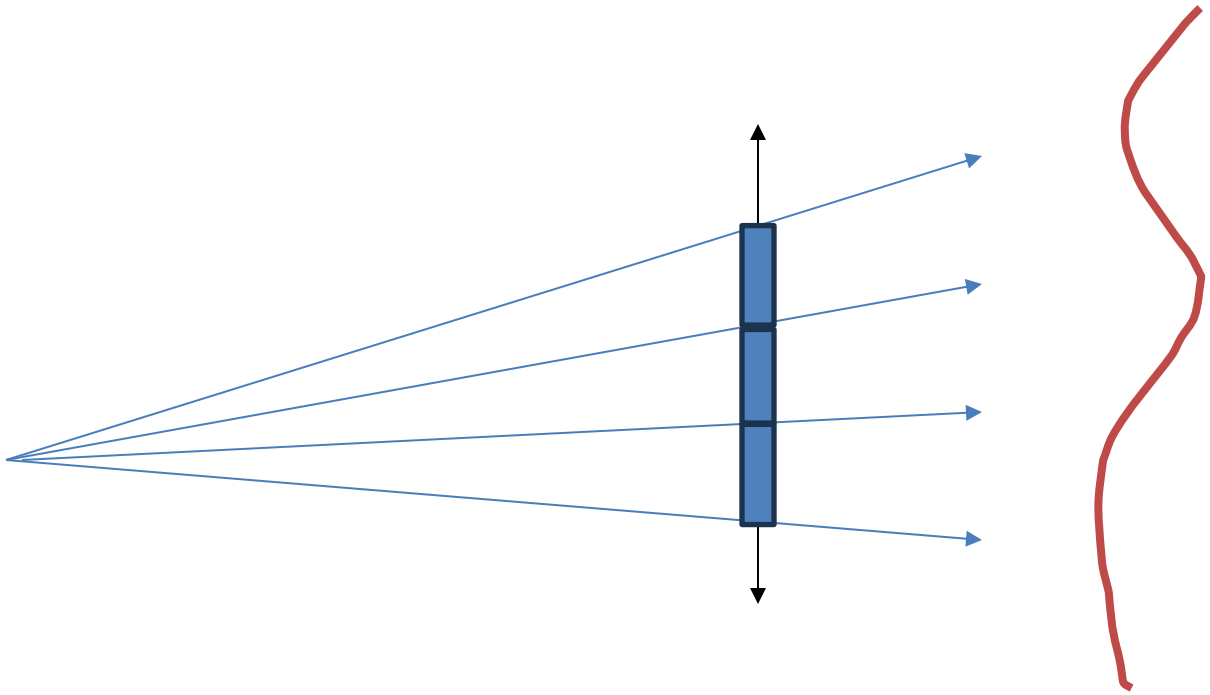


Simple solution: 1x1 splat



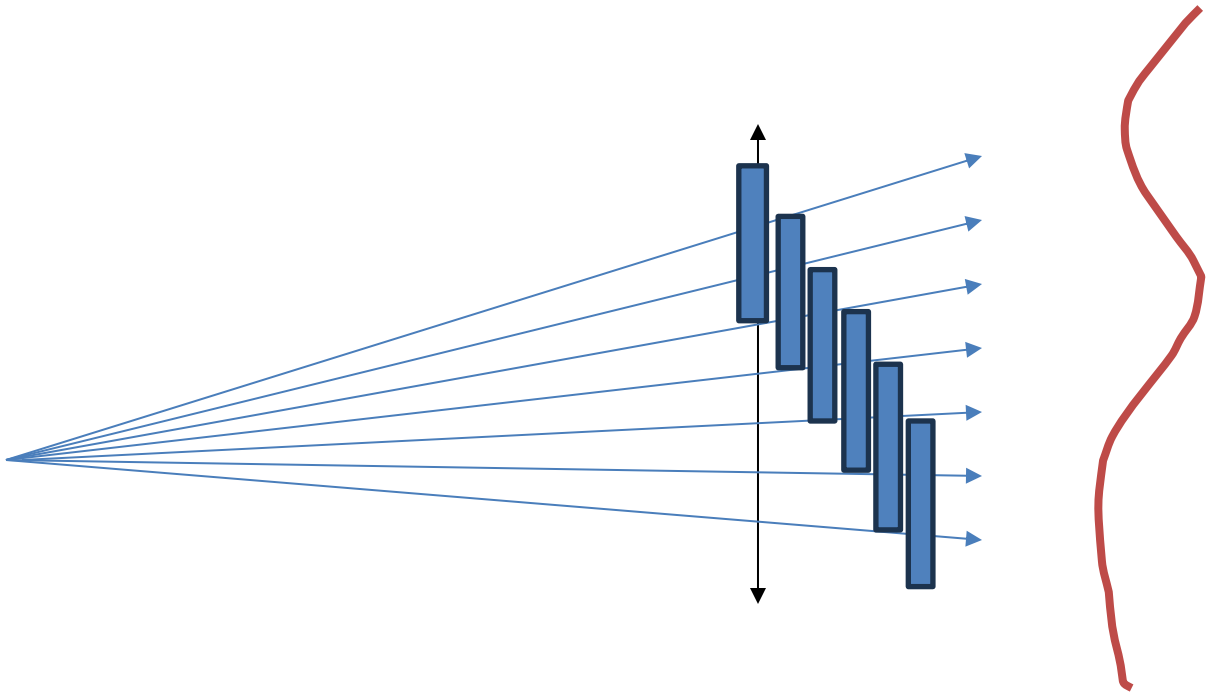
Issues?

Simple solution: scaled



Issues?

Simple solution: 3x3 splat

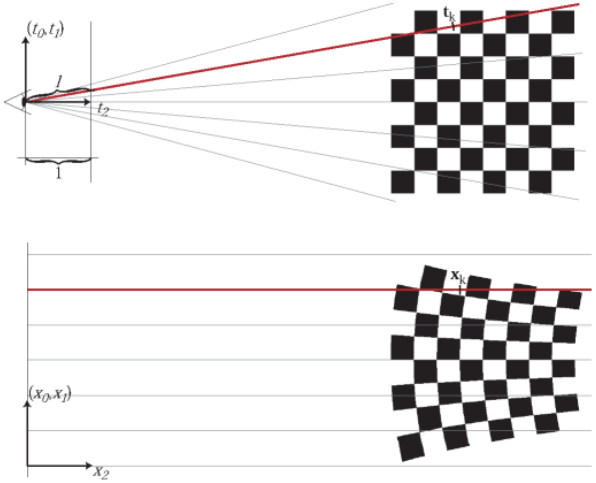


Issues?

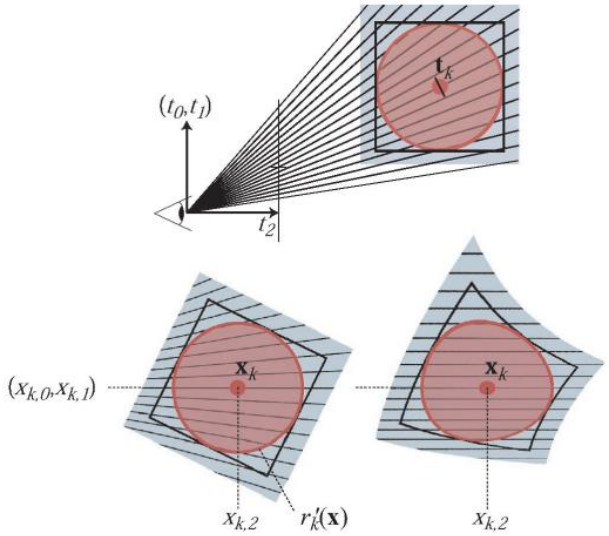
Problem

- If you render/ray-trace/ray-cast/NERF/warp “one pixel” how do you render it to the framebuffer?

Camera space



Ray space



Westover splatting:

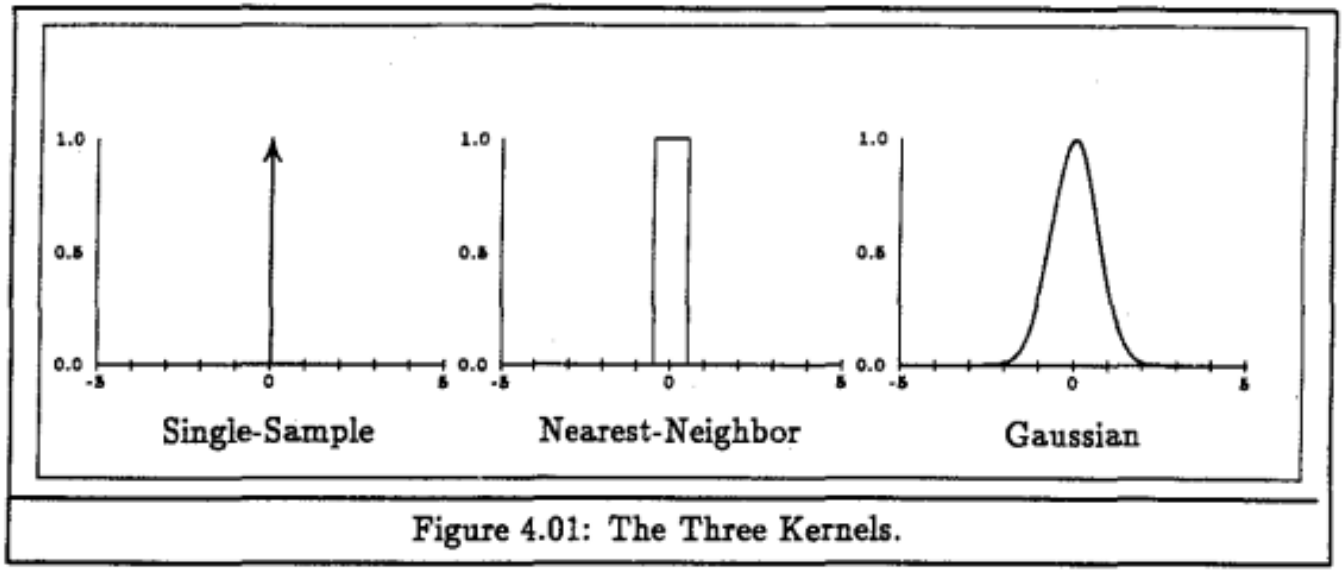
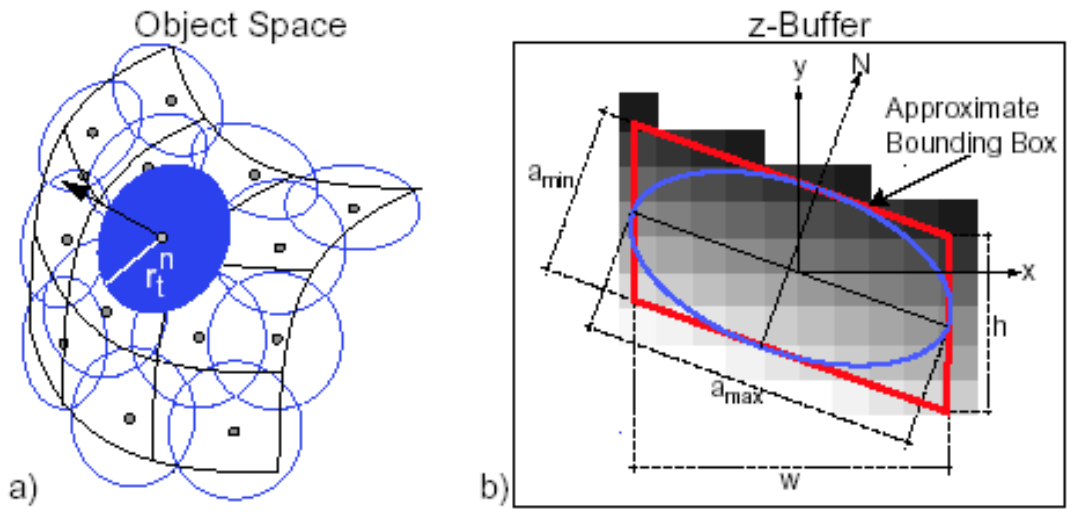


Figure 4.01: The Three Kernels.

“EWA Splatting” and Reconstruction



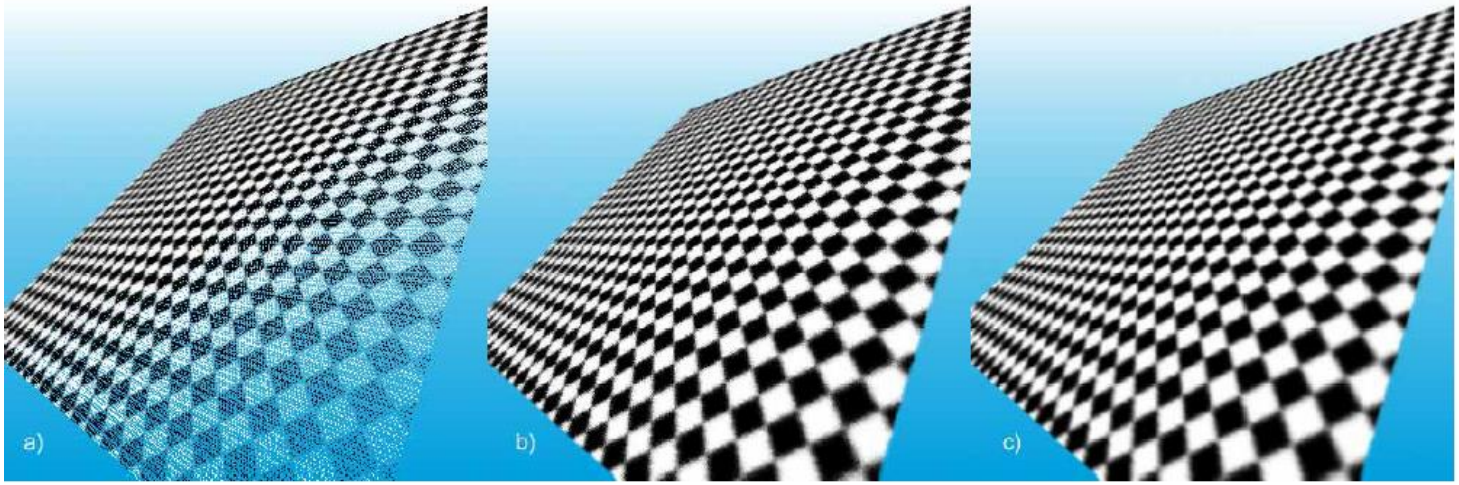


Figure 11: *Tilted checker plane. Reconstruction filter: a) Nearest neighbor. b) Gaussian filter. c) Supersampling.*

3D Gaussian Splatting

Kerbl et al. 2023

Inspiration:
Differentiable Surface Splatting for Point-based Geometry Processing, by Yifan et al. 2019

- Optimize points+normals to match a reference rendering (using differentiable point-based renderer)

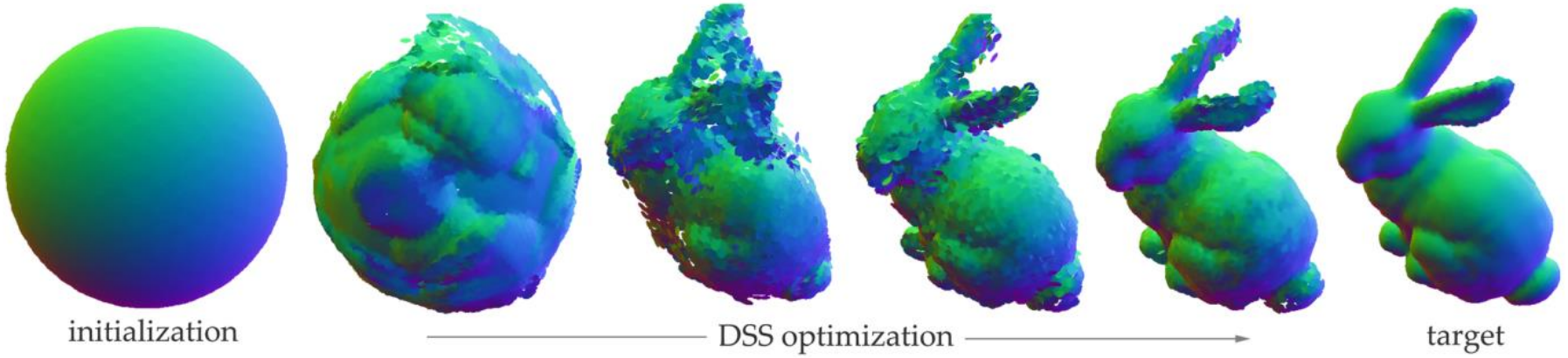
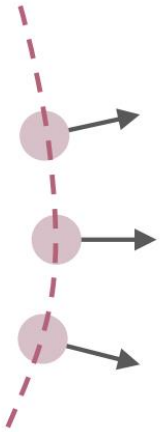


Fig. 1. Using our differentiable point-based renderer, scene content can be optimized to match target rendering. Here, the positions and normals of points are optimized in order to reproduce the reference rendering of the Stanford bunny. It successfully deforms a sphere to a target bunny model, capturing both large scale and fine-scale structures. From left to right are the input points, the results of iteration 18, 57, 198, 300, and the target.

3D Gaussian Splatting

Kerbl et al. 2023

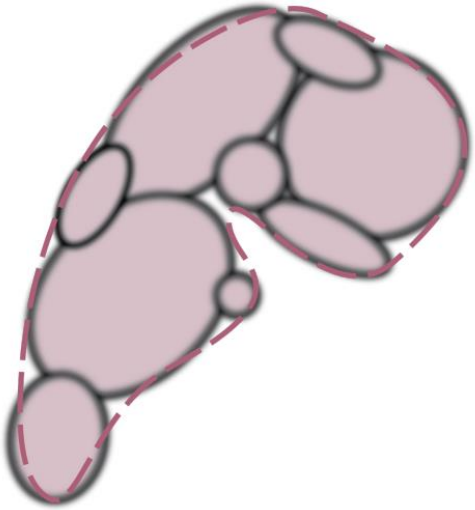
- Here, change from surface splatting to volume splatting…



position normal

$$p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad n = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \quad \sigma \text{ std dev} \quad f \text{ appearance}$$

Surface splatting



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

covariance matrix

Volume splatting

3D Gaussian Splatting

Kerbl et al. 2023

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].

- https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Content Generation

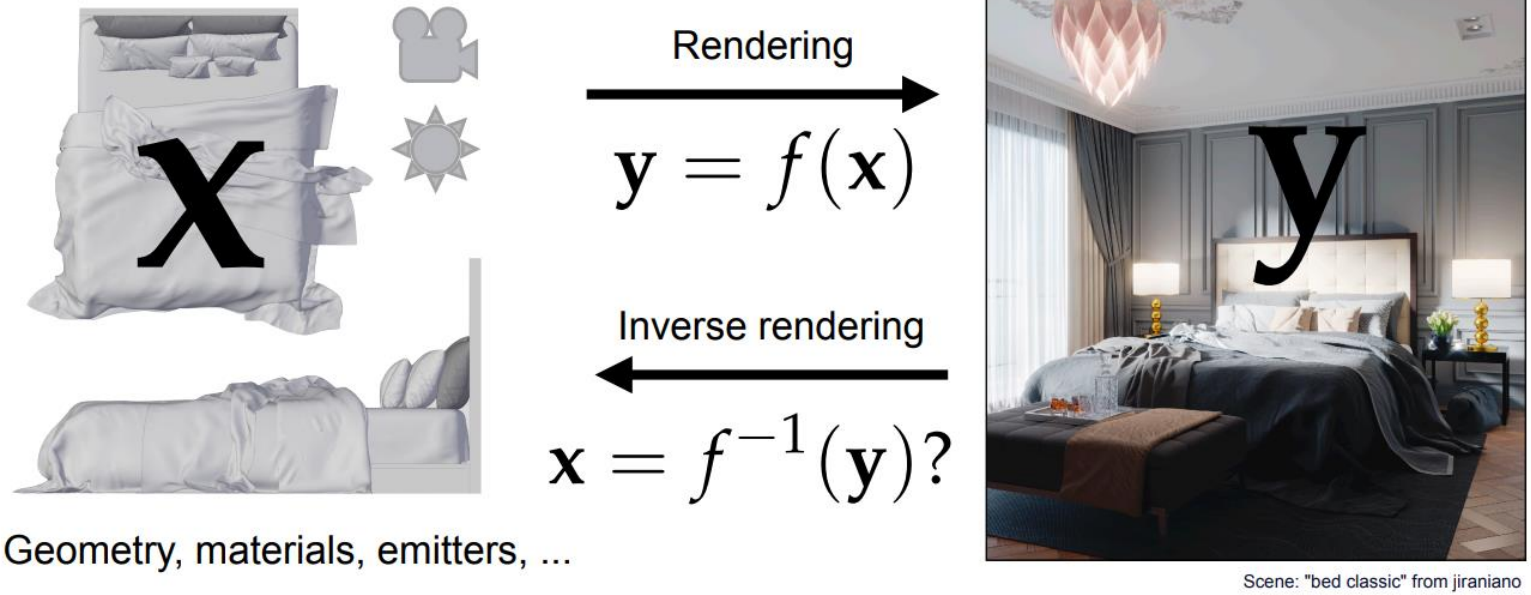
- Lightfields and NERFs
- Splatting
- Differential Rendering
- Diffusion Models

Photometric Stereo

- From images to normals

Differentiable Rendering

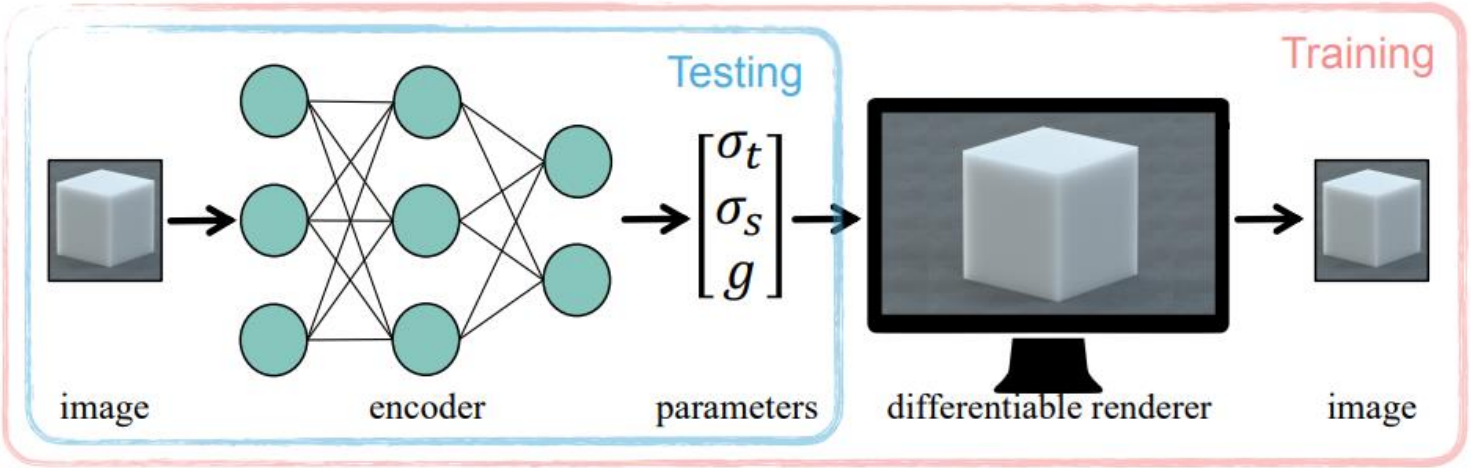
- From images to scene parameters
  - aka “a form of inverse rendering”



[from SIGGRAPH 2020 Course]

Differentiable Rendering

- One option: combine physics-based rendering and ML and inference techniques
- e.g., inverse subsurface scattering (Che et al. 2020)



- Utilizing *image loss* (provided by a volume path tracer) to regularize training
- Use the trained encoder to solve inverse problems during testing

[from SIGGRAPH 2020 Course]

Differentiable Rendering

- Why?
  - Can render/change existing renderings
  - Machine learning/deep learning is well suited

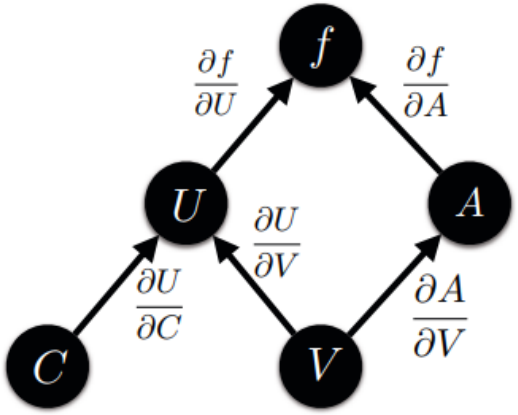
- Three types of DR:
  - **Physics-Based**
  - NeRF-Based
  - 3DGS-Based

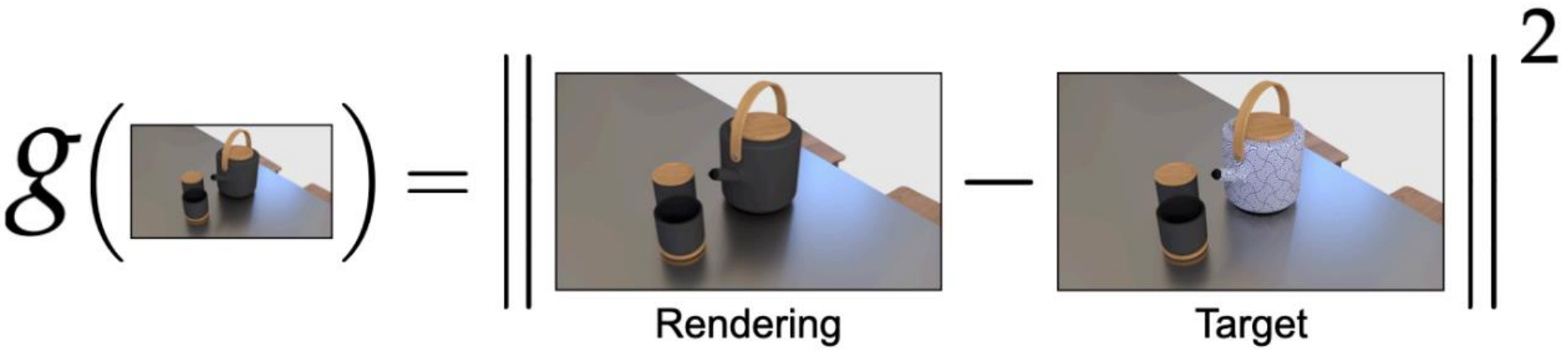
"OpenDR: An Approximate Differentiable Renderer" [Loper and Black, 2014]

- $f(\theta)$ is the renderer, where θ are all parameters used to create the image
- θ can be factored into $\{V, C, A\}$ which are vertices V, camera parameters C, and per-vertex brightness A.
- Let U be intermediate variables of 2D projected vertex coordinate positions

OpenDR

- Derivatives of interest are:
  - $\frac{\partial f}{\partial A}$ (changes of appearance)
  - $\frac{\partial U}{\partial C}$ and $\frac{\partial U}{\partial V}$ (changes of projected coords)
  - $\frac{\partial f}{\partial U}$ (changes in image-space deformation)





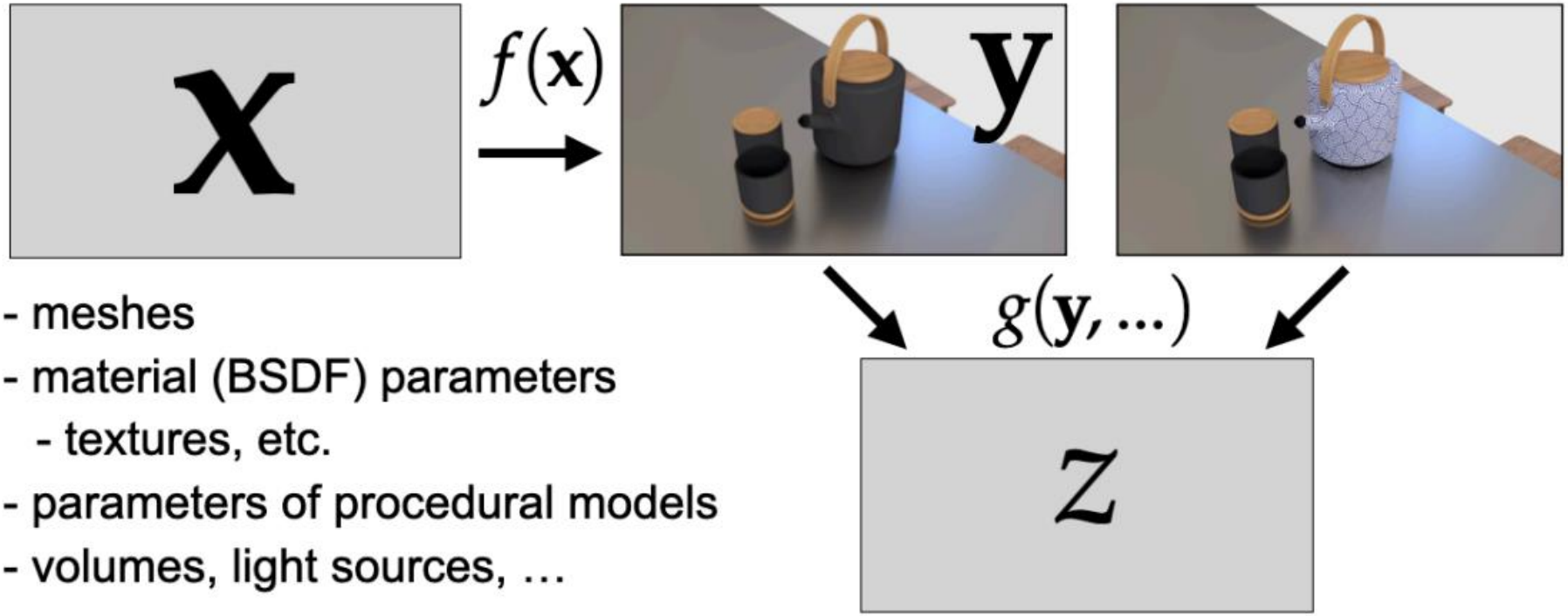
The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$

Scene parameters

Objective

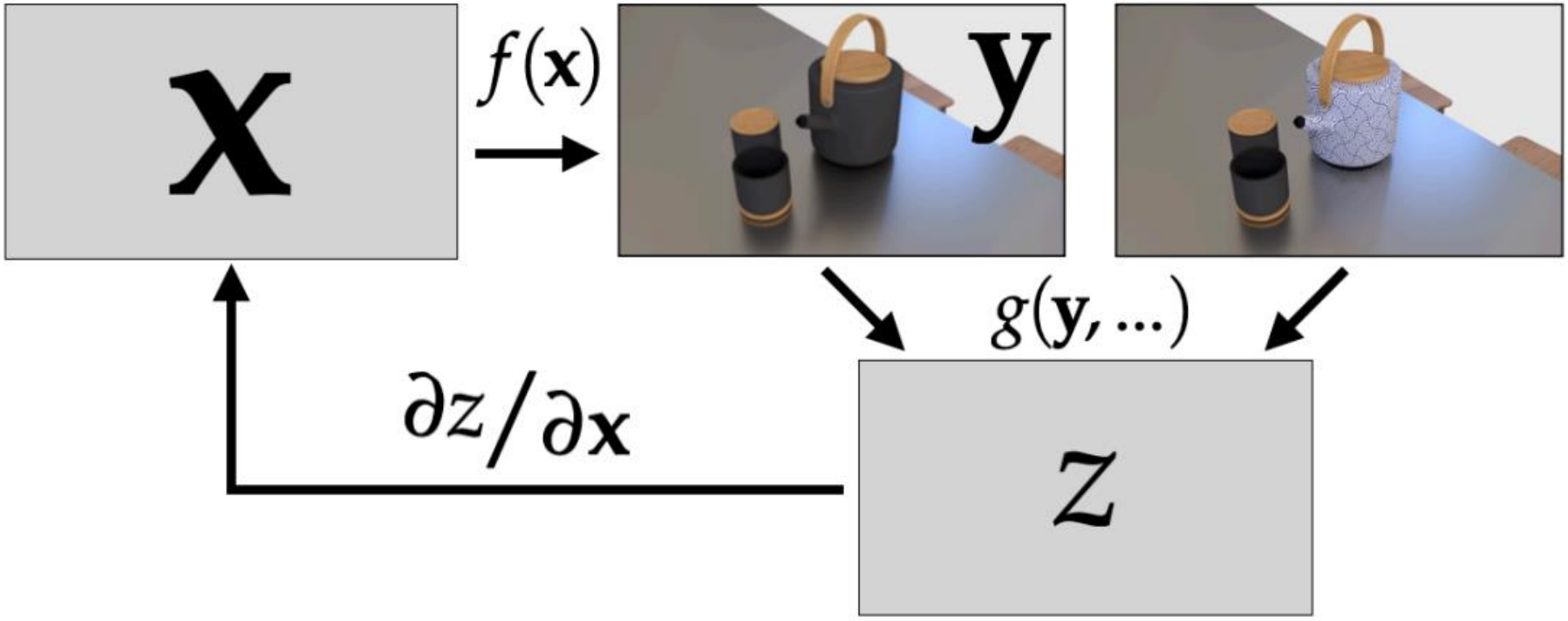
Rendering algorithm

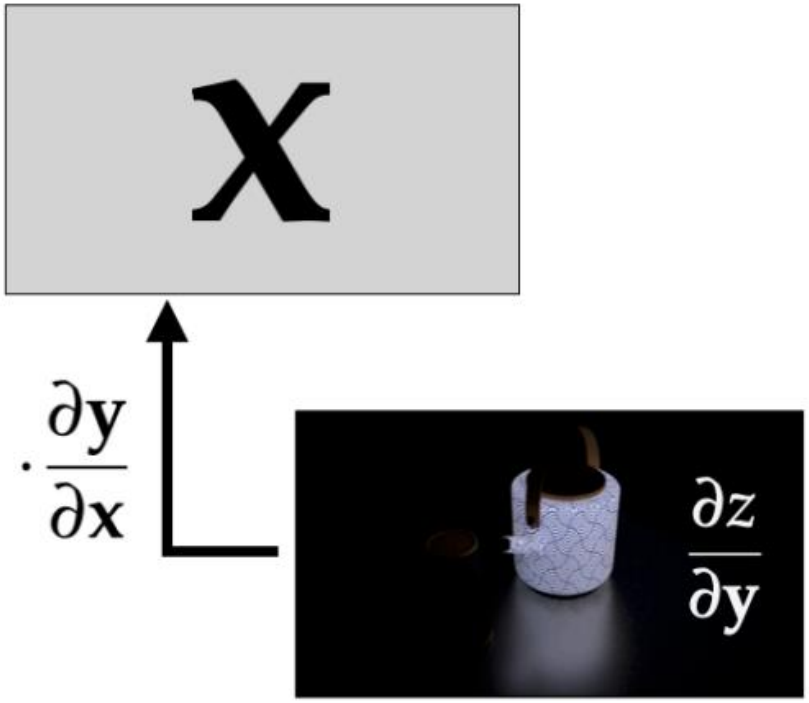
The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$



- meshes
- material (BSDF) parameters
  - textures, etc.
- parameters of procedural models
- volumes, light sources, …

The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$





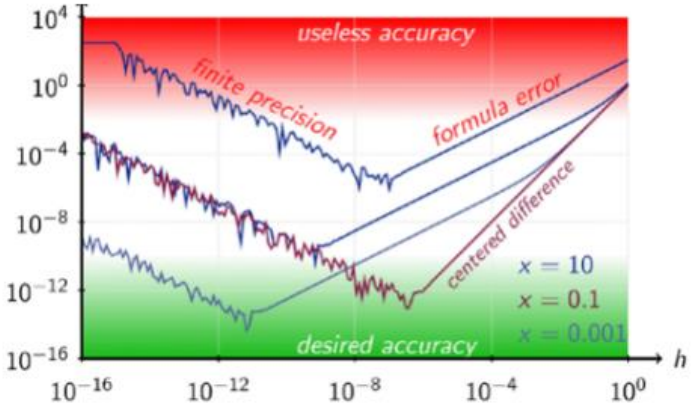
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

Challenges

1. Differentiating f
2. Matrix multiplication
3. Efficiency?
4. How to deal with edges?

Use finite differences!

$$\frac{\partial \mathbf{y}}{\partial x_i} = \frac{f(\mathbf{x} + \varepsilon\,\mathbf{e}_i) - f(\mathbf{x} - \varepsilon\,\mathbf{e}_i)}{2\,\varepsilon}$$

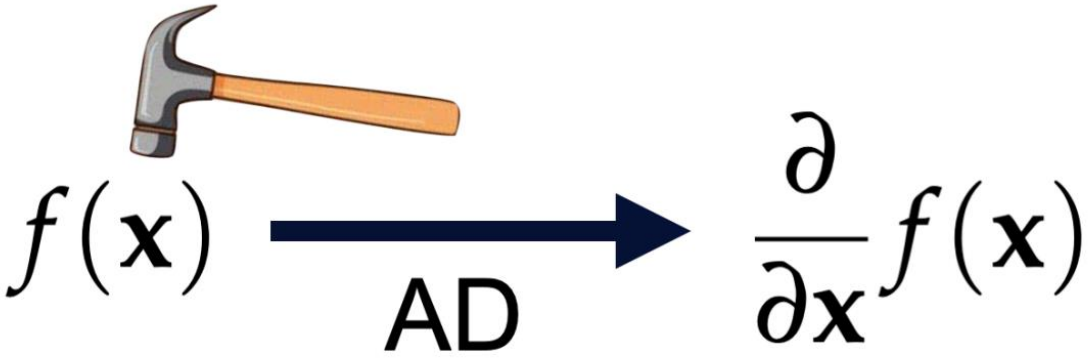


[Wikipedia]

Potential problems:

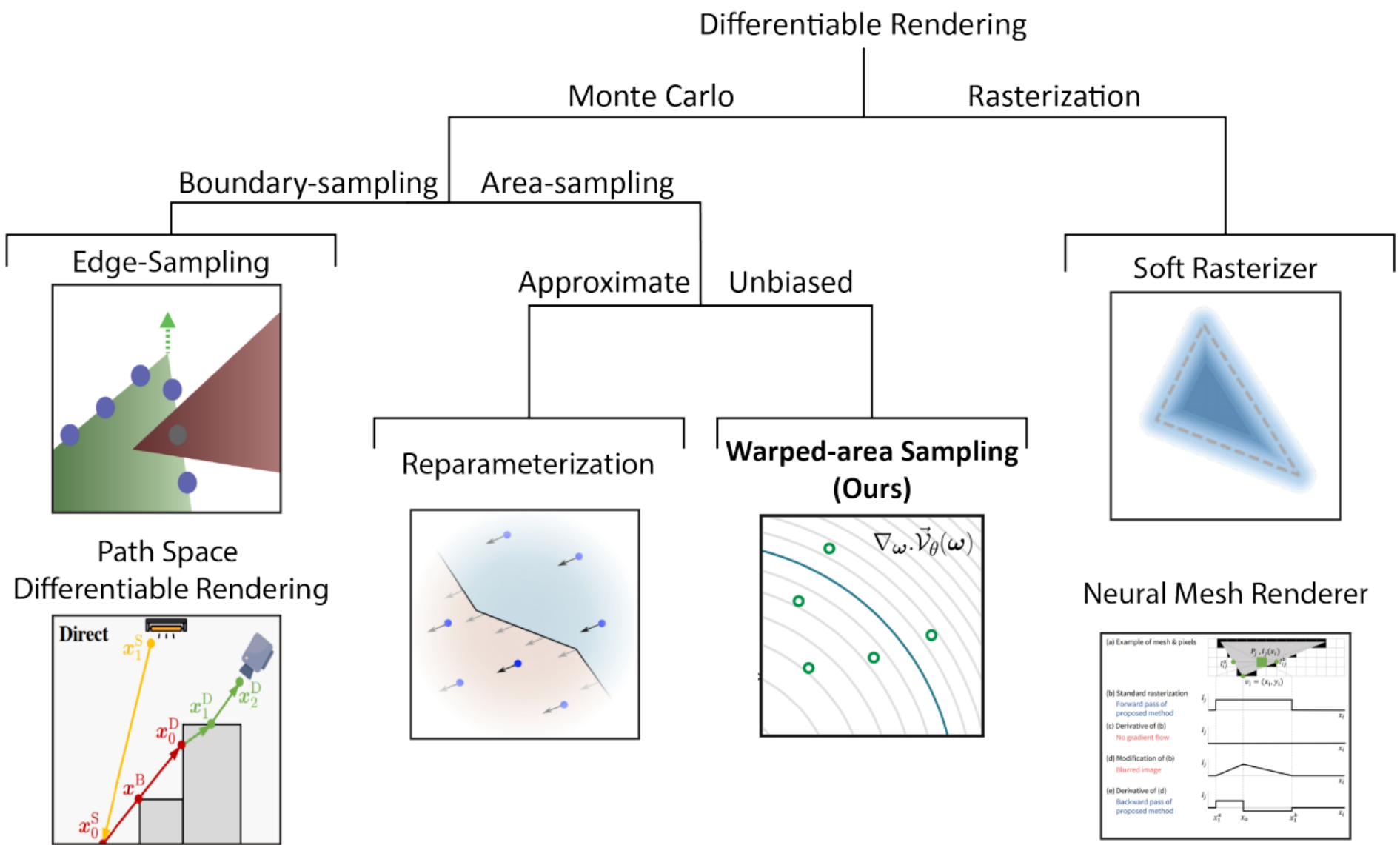
- Bad approximation (big ε), rounding error (small ε)
- Need to correlate Monte Carlo samples
- Extremely slow when many there are many parameters.

Automatic Differentiation

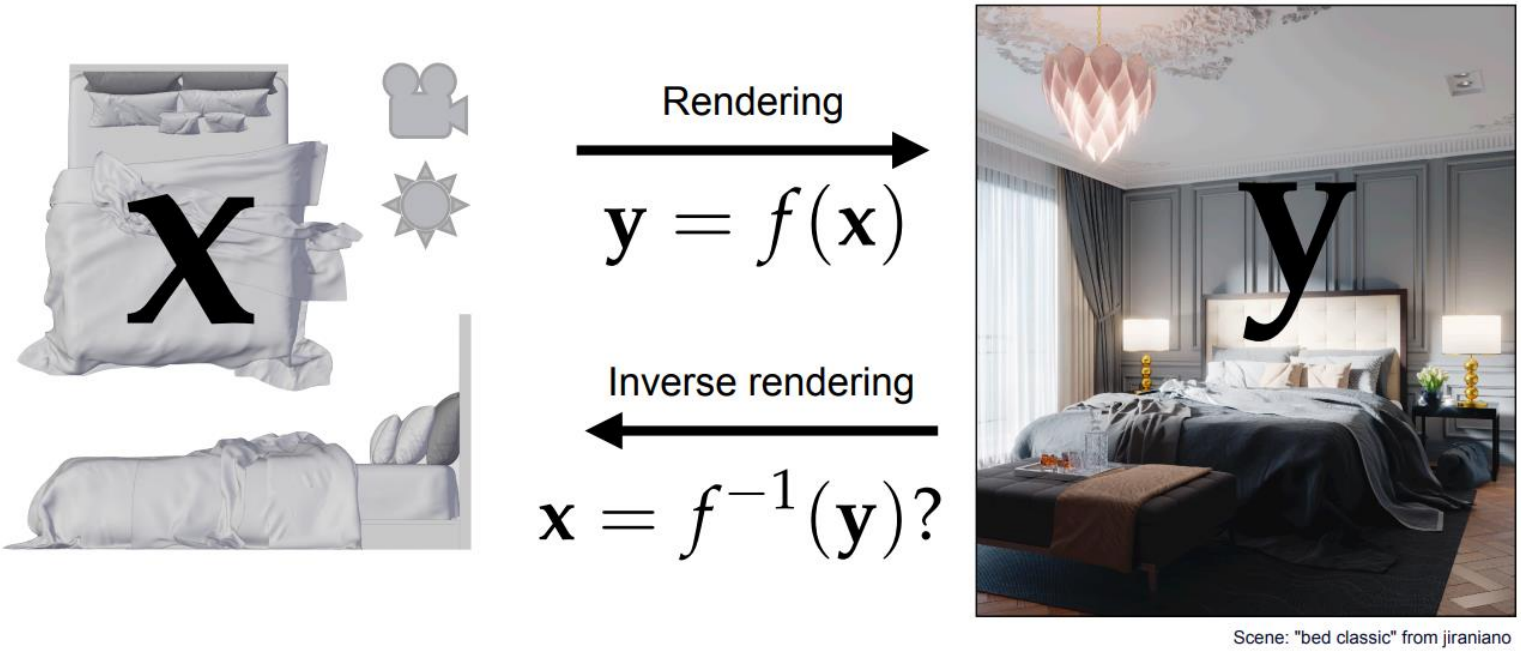


Maybe?

or…. (from Bangaru et al. 2020)



Differentiable Rendering



From image to X, where X = {volume, surface, triangles, mesh…} with color, material, and lighting parameters

Examples

- NERF
  - To volume…
- 3DGS
  - To volume/splats…
- Recursive Control Variates for Inverse Rendering
  - To path tracing, SIGGRAPH 2023, https://research.nvidia.com/labs/rtr/publication/nicolet2023recursive/
- Triangle Splatting for Real-Time Radiance Field Rendering
  - *To triangles*, 3DV 2026, https://trianglesplatting.github.io/
- MeshSplatting: Differentiable Rendering with Opaque Meshes
  - *To meshes*, CVPR 2026, https://meshsplatting.github.io/
- Radiance Surfaces: Optimizing Surface Representations with a 5D Radiance Field Loss
  - *To surfaces*, SIGGRAPH 2025, https://rgl.epfl.ch/publications/Zhang2025Radiance
- NVIDIA
  - https://research.nvidia.com/labs/rtr/tag/differentiable-rendering/

Examples (another flavor)

- Constructive Solid Geometry
- CSG and Inverse CSG

Solid Modeling

History:

CNC: ~1950

Mainframe Computers: ~1960's

BREP: 1970 (Baumgart)

CSG: 1974 (Ian Braid)

Computerized Drafting

Advantages:

Saves on storage/retrieval;

Easy modification, update;

Shortcomings:

Can't analyse the

strength, shape,

geometry, weight

center of mass, center of inertia

Popular Commercial tools: AutoCAD, CADKEY…

Constructive Solid Geometry (CSG)

Introduced: Ian Braid (Cambridge University, ~74)

Concepts:

Primitives: small set of shapes

Transformations: scaling, Rotation, Translation

Set-theoretic Operations Union, Intersection, Difference

[Euler operators]

Combinations of these → Solid part

Euler operators

U* (regular union)

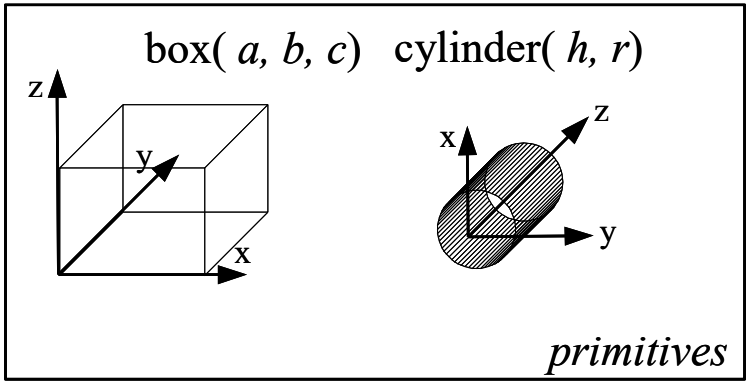
-* (regular difference)

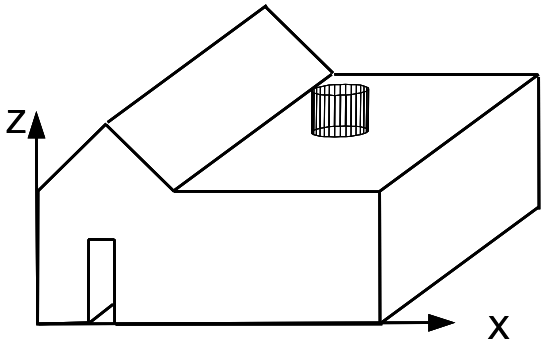
∩* (regular intersection)

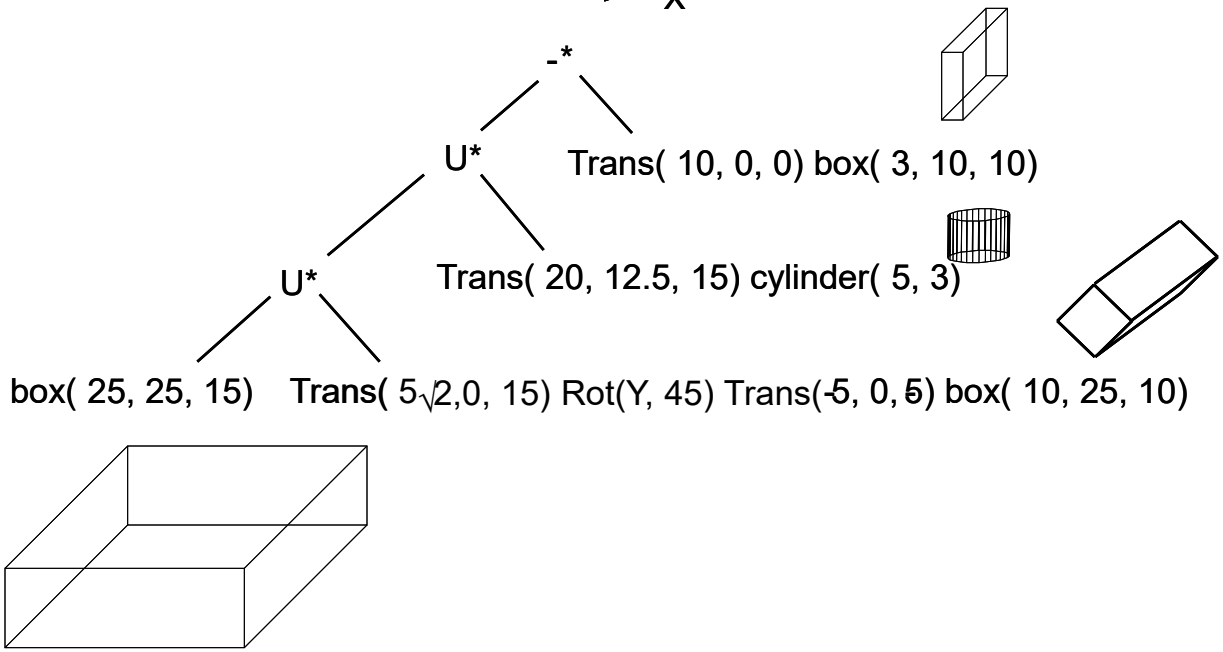
CSG Tree:

Sequence of operators → design

Examples of CSG







Questions:

Can we use a different set of primitives ?

Is the CSG representation unique ?

[how to determine if two solids are identical ?]

Regularized operators

Is the set of 3D solids is closed with respect to (U, -, ∩)?

closure of a set S: kS

interior of a set S: iS

$$A \cup^* B = k\, i\, (A \cup B)$$

$$A -^* B = k\, i\, (A - B)$$

$$A \cap^* B = k\, i\, (A \cap B)$$

Why is closure over operations important?

uniform data structures

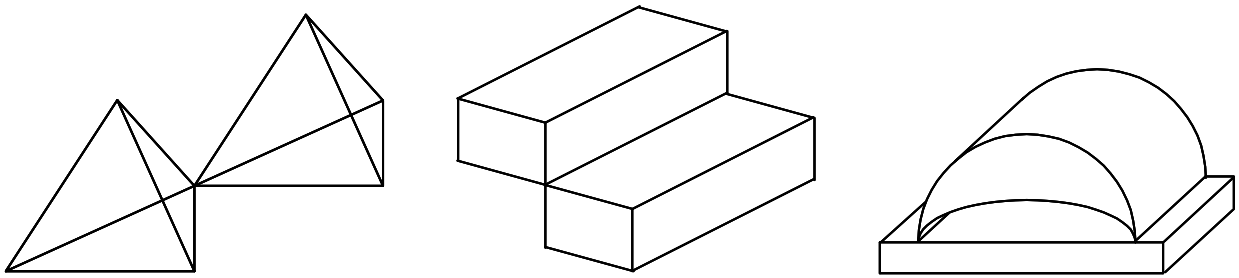
Regularized Euler Operators

Maintain solid as a *regular 2-Manifold*

2-Manifold regular solids

Open neighborhood of each point is similar to an open disc

Non 2-Manifold:



Problems with CSG

Non-Unique representation

Difficulty of performing analysis for some tasks

BREP (Boundary REPresentation)

What entities define the

Boundary of a solid ?

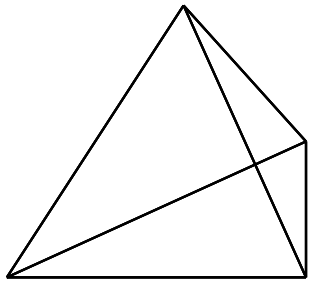
Boundary of surfaces?

Boundary of curves (edges) ?

Boundary of points ?

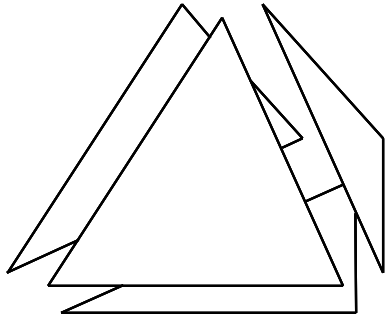
BREP

Boundary of a solid…



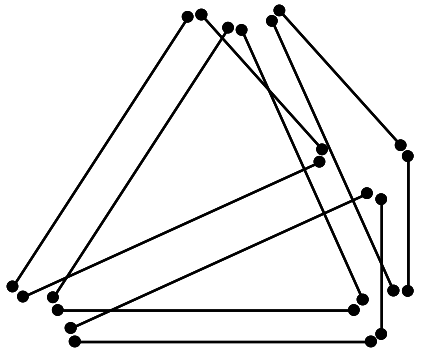
(a) Solid: bounded, connected subset of E^3

Boundary of surfaces…



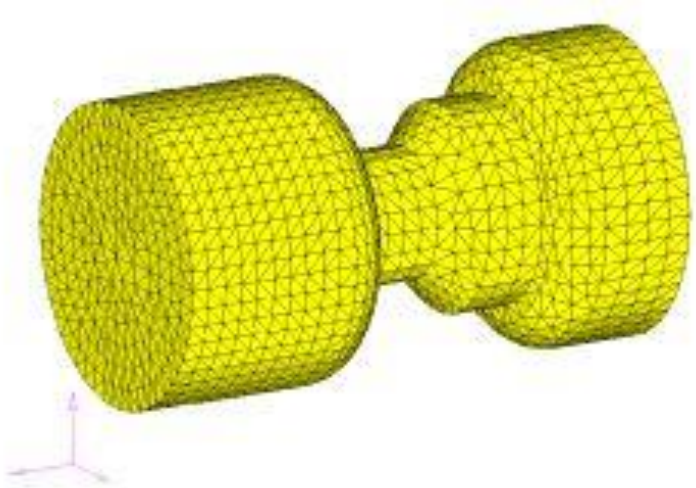
(b) Faces: boundary of solid
bounded, connected subsets of Surfaces

Boundary of curves (edges)…



(c) Edges: boundary of faces
bounded, connected subsets of curves

BREP: Polyhedral models



Using a Boundary Model

Compute Volume, Weight

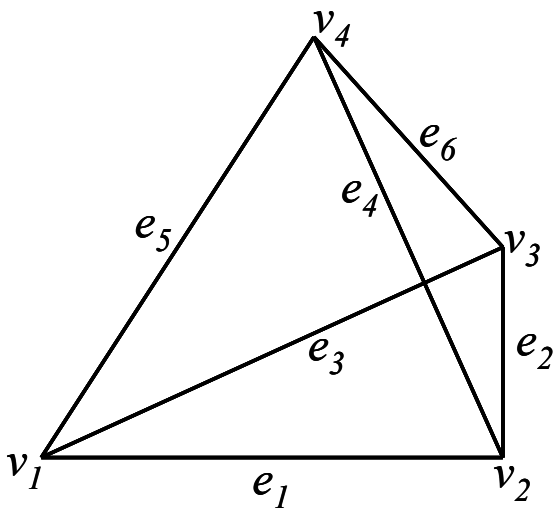
Compute Surface area

Point inside/outside solid

Intersection of two faces

…

An Edge-Based Model



Faces:

f_1	$e_1\ e_4\ e_5$
f_2	$e_2\ e_6\ e_4$
f_3	$e_3\ e_5\ e_6$
f_4	$e_3\ e_2\ e_1$

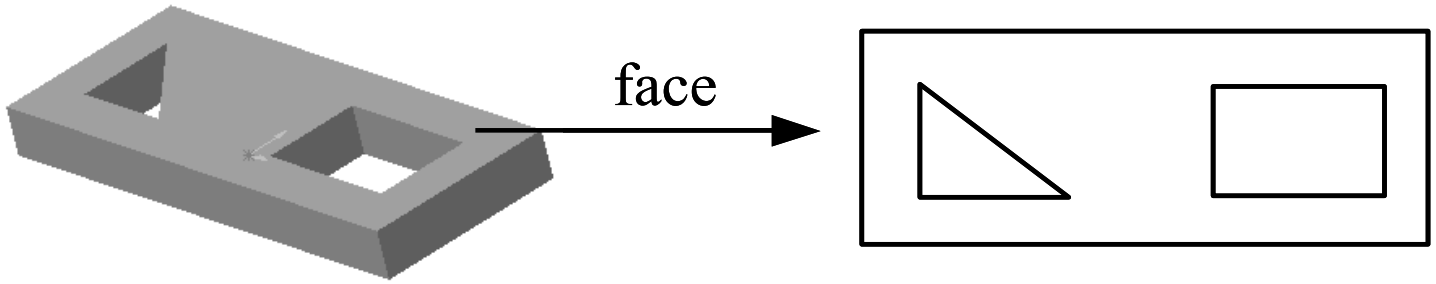
Edges:

e_1	$v_1\ v_2$
e_2	$v_2\ v_3$
e_3	$v_3\ v_1$
e_4	$v_2\ v_4$
e_5	$v_1\ v_4$
e_6	$v_3\ v_4$

Vertices:

v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$

Edge-Based Models: *inefficient algorithms*



Compute Surface Area:

1. Identify Loops
2. Compute area of each loop
3. Compute area of face

The Winged-Edge Data Structure

Efficient implementation of often-used algorithms

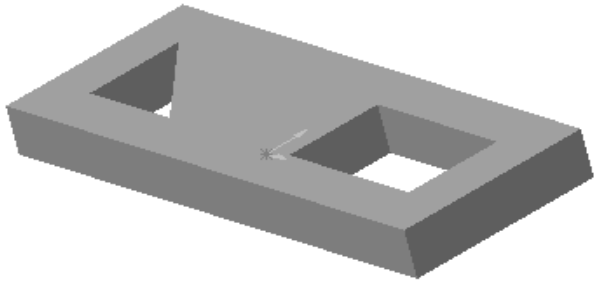
Area of Face

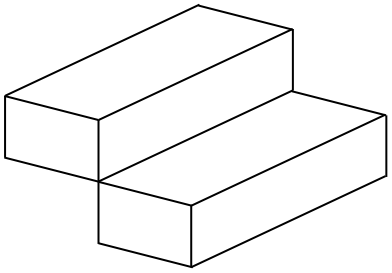
Hidden surface removal

Find neighbor-faces of a face

Observations

2-Manifold => Each edge is shared by exactly 2 faces

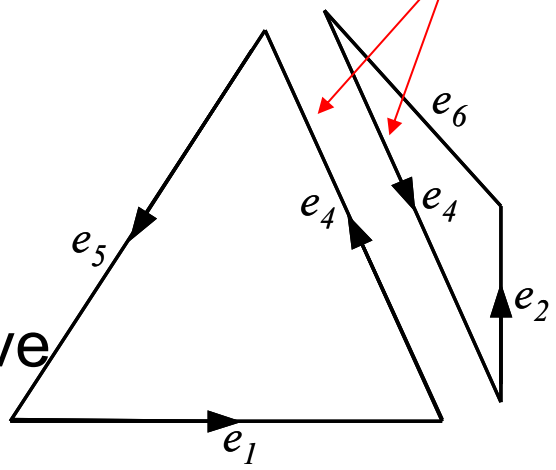




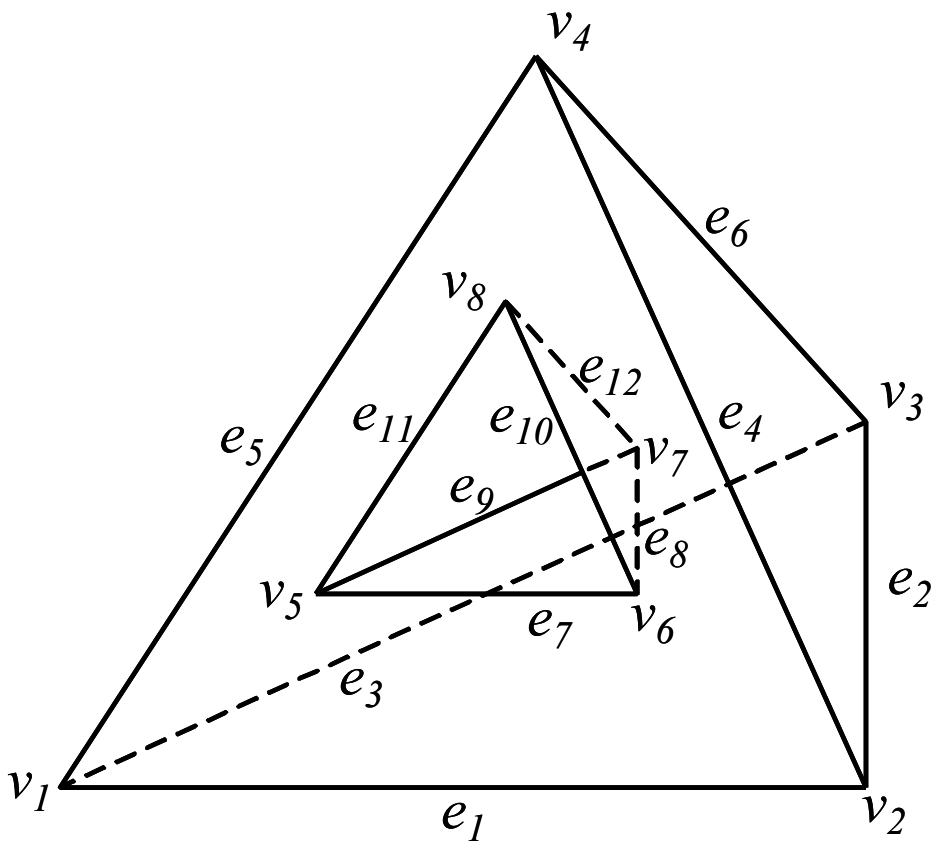
co-edges

Face CCW convention =>

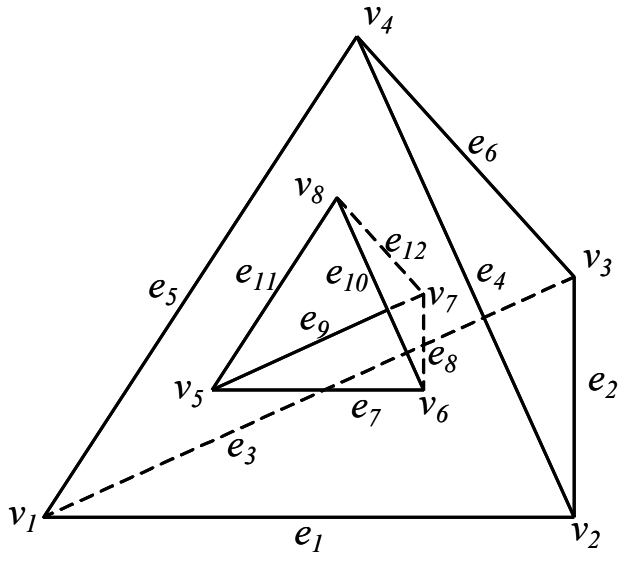
Each edge is once +ve, once -ve



BREP Example



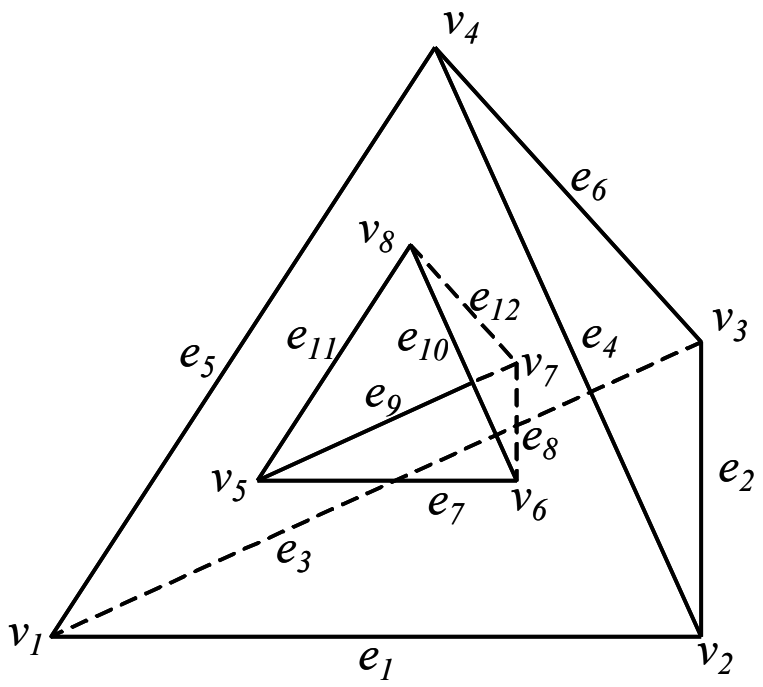
BREP Example



Vertices:

v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

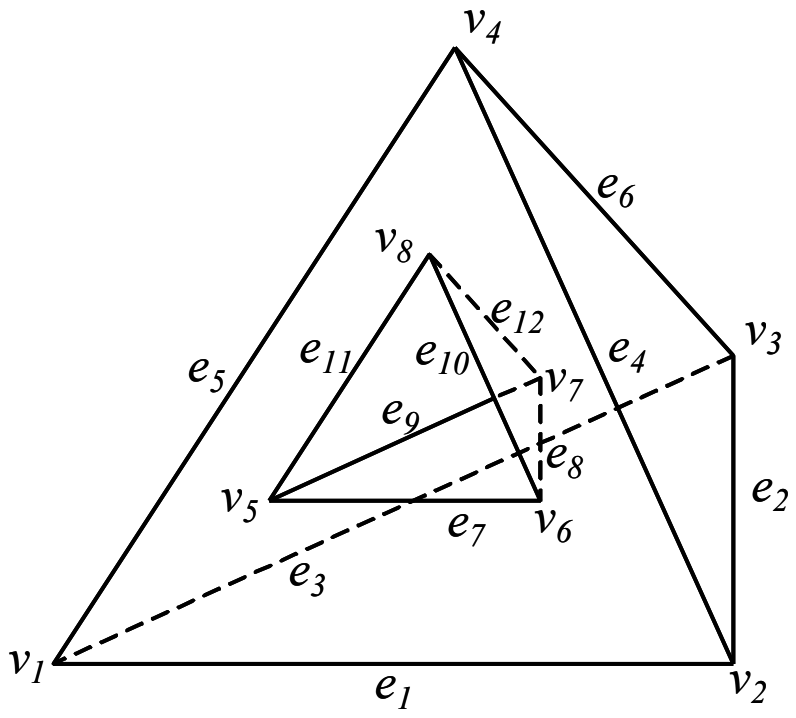
BREP Example..



Edges:

e_1	v_1 v_2
e_2	v_2 v_3
e_3	v_3 v_1
e_4	v_2 v_4
e_5	v_1 v_4
e_6	v_3 v_4
e_7	v_5 v_6
e_8	v_6 v_7
e_9	v_7 v_5
e_{10}	v_6 v_8
e_{11}	v_5 v_8
e_{12}	v_7 v_8

BREP Example…



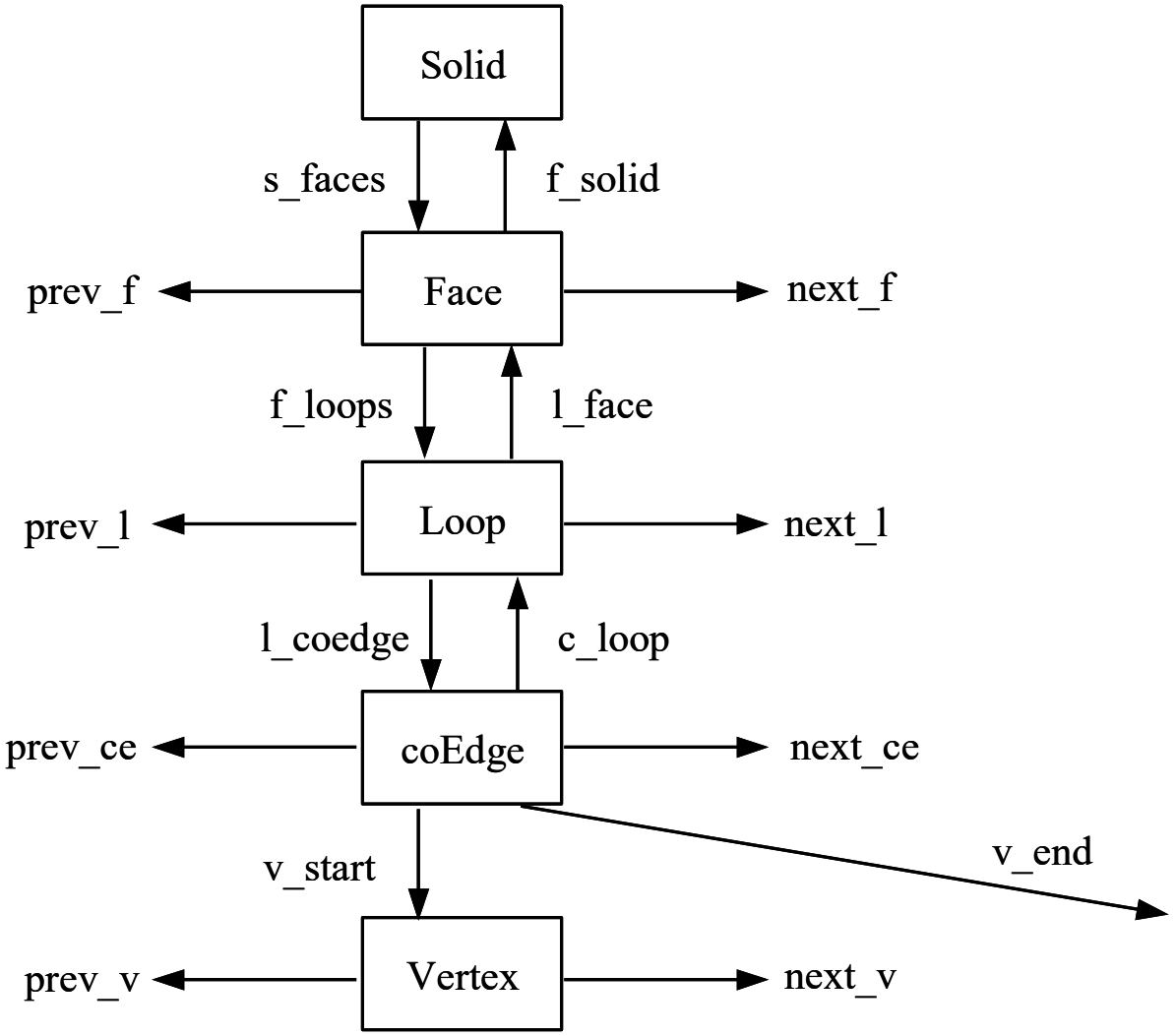
Faces:

f_1	l_1 l_2
f_2	l_3
f_3	l_4
f_4	l_5
f_5	l_6
f_6	l_7
f_7	l_8

Loops:

l_1	$+e_1$ $+e_4$ $-e_5$
l_2	$-e_7$ $+e_{11}$ $-e_{10}$
l_3	$+e_2$ $+e_6$ $-e_4$
l_4	$+e_5$ $-e_6$ $+e_3$
l_5	$-e_1$ $-e_3$ $-e_2$
l_6	$+e_7$ $+e_8$ $+e_9$
l_7	$+e_{10}$ $-e_{12}$ $-e_8$
l_8	$-e_{11}$ $-e_9$ $+e_{12}$

BREP: Winged edge data structure



BREP or CSG ?

Using: CSG is more intuitive

Computing: BREP is more convenient

Modern CAD Systems:

CSG for GUI (feature tree)

BREP for internal storage and API's

BREP: non-polyhedral models?

Same Data Structure, plus

For each edge, store equation

For each curved face, store equation

Why do we need to learn all this ?

(a) To anticipate when an operation will fail

(b) To allow us to write API's

CSG

- https://inversecsg.csail.mit.edu/

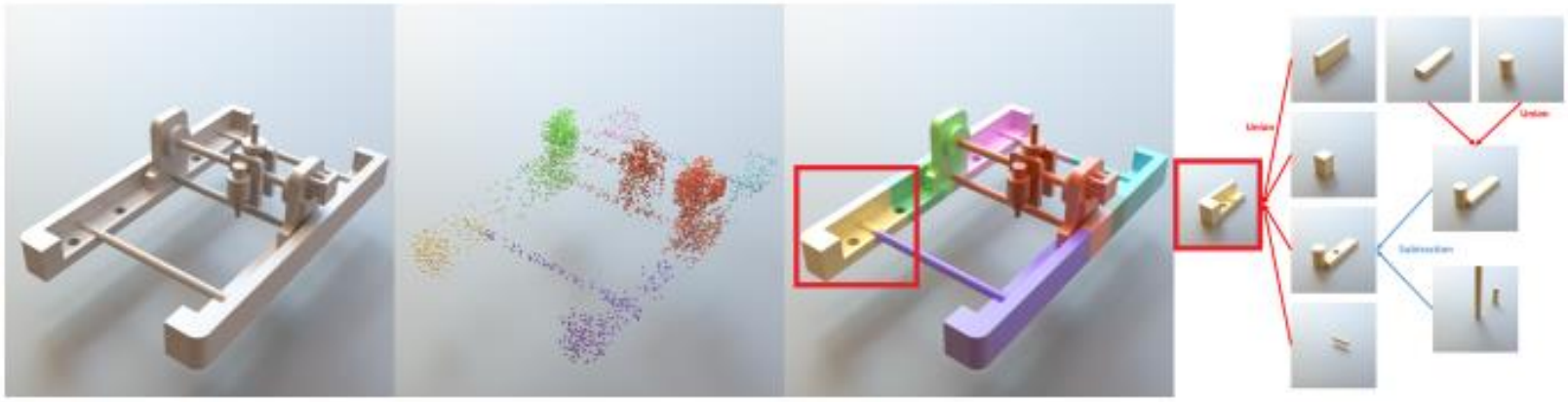


Fig. 1. We provide a system that takes as input a mesh file (left) and outputs its constructive solid geometry (CSG) representation. The three key ideas are a) use of carefully designed point samples to guide a purely discrete search problem (points, middle left), b) a divide-and-conquer algorithm to segment problems to ensure success (colors, middle left), and c) use of program synthesis techniques to solve the hard discrete search problem in each segment. The output CSG structure (middle right) correctly infers over 50 solid primitives and 18 boolean operators. A part of the solution (red box) is extracted for demonstration (right).

Content Generation

- Lightfields and NERFs
- Splatting
- Differential Rendering
- Diffusion Models

Diffusion Models

- From noise to data…

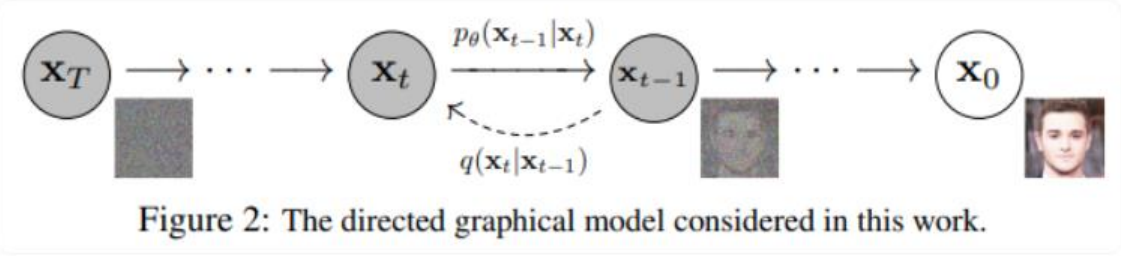
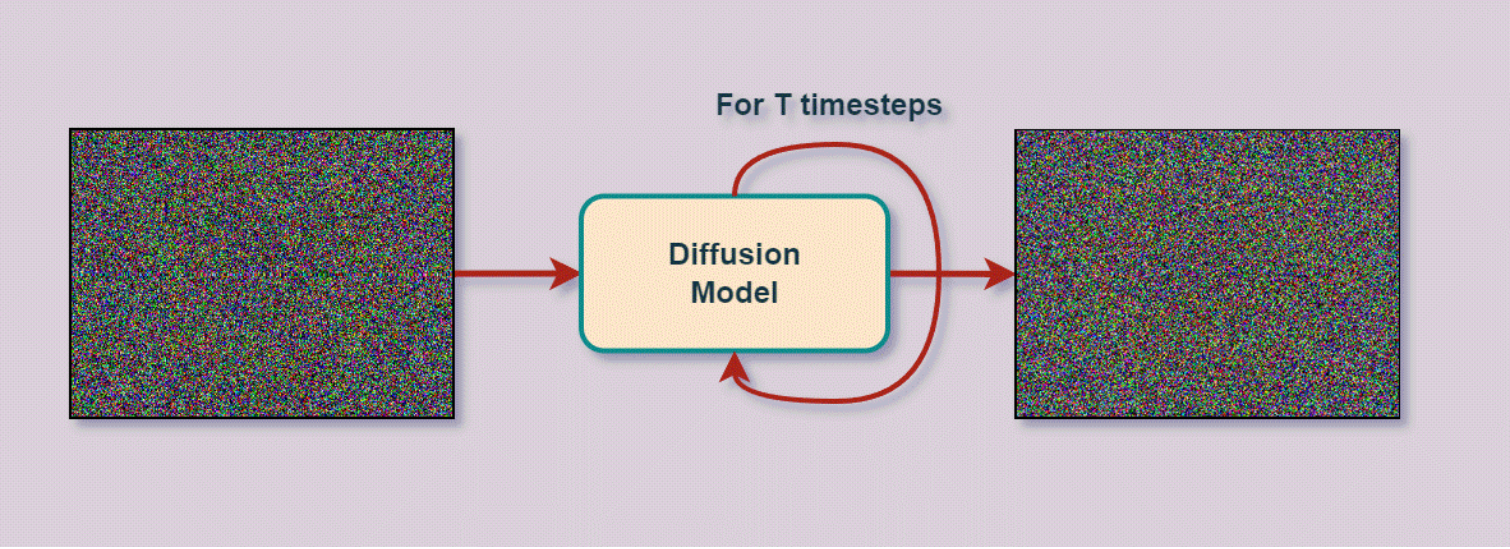


Figure 2: The directed graphical model considered in this work.

- Four popular diffusion models:
  - OpenAI's Dall-E 2
  - Google's Imagen
  - StabilityAI's Stable Diffusion
  - Midjourney

Diffusion Models



[https://learnopencv.com/image-generation-using-diffusion-models/]