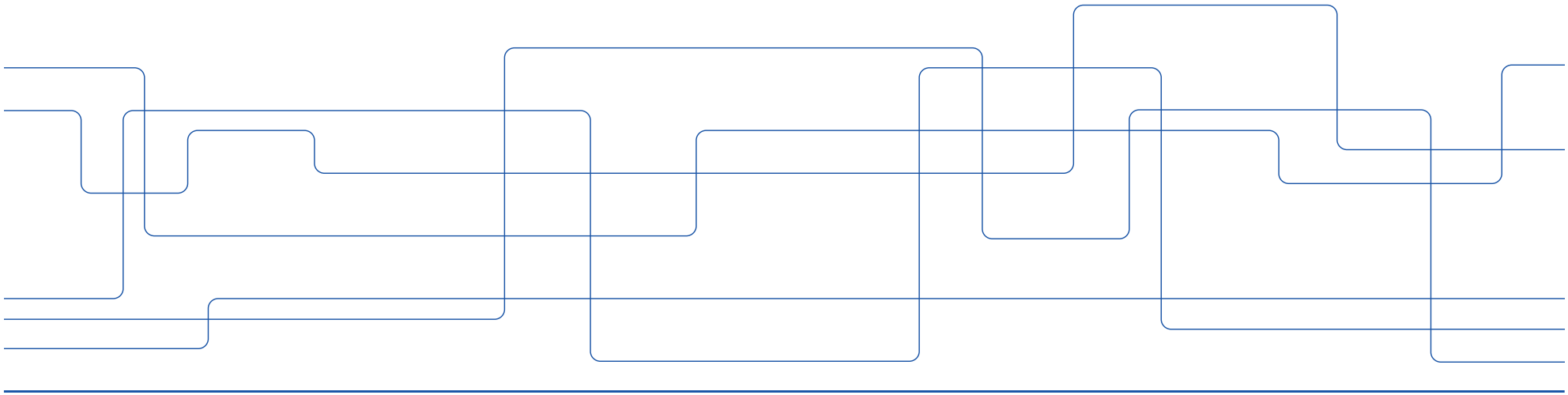
Novel view synthesis

Mårten Björkman



From Lecture 11: Bundle adjustment

- Feature extraction, matching and bundle adjustment using `ColMap`.

From Lecture 11: Bundle adjustment

- Assume you have K camera images with unknown projection matrices P_k and projections q_{ik} of N unknown 3D points Q_i.
- Then you can set up a large system of equations and search for P_k and Q_i by minimizing

$$\min_{\{P_k, Q_i\}} \sum_{k=1}^{K} \sum_{i=1}^{N} d(q_{ik}, f(P_k, Q_i))$$

where

$$d\big(q_{ik}, f(P_k, Q_i)\big) = \left(x_{ik} - \frac{p_k^1 Q_i}{p_k^3 Q_i}\right)^2 + \left(y_{ik} - \frac{p_k^2 Q_i}{p_k^3 Q_i}\right)^2$$

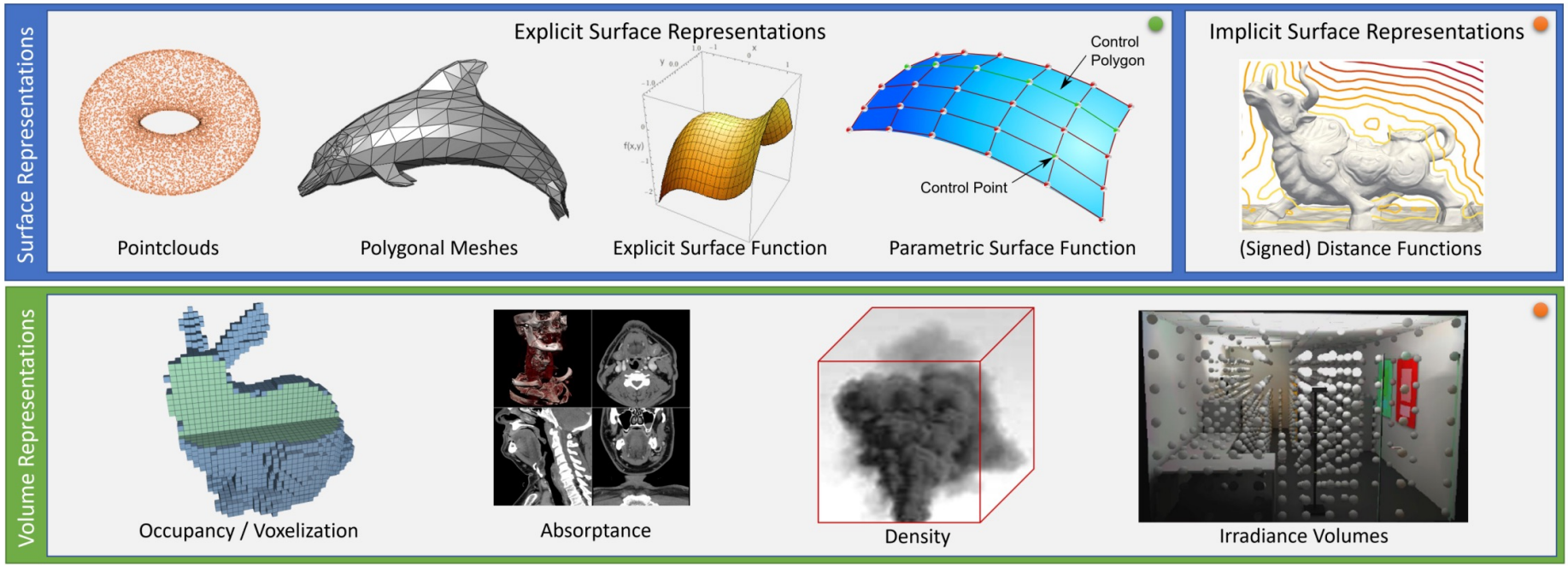
- After initialization by first computing the essential matrices between pairs of camera images, this can be solved iteratively.

ColMap – Structure from motion (SfM)

- By matching not just between two images (lecture 11), but with multiple images and enforcing consistency, a dense depth map per image can be recovered.
  - Photometric consistency (colours are similar between images)
  - Geometric consistency (possible using depths to map from one image to another)
- The 3D reconstruction above is based on 21K images around central Rome.

Schönberger, “Pixelwise view selection for unstructured multi-view synthesis”, ECCV 2016.

Representations of 3D shapes



- There are different kinds of representations for surfaces and volumes

Explicit or implicit representations

- Define a 3D surface based on some 2D surface coordinate system:

$$S_{explicit} = \left\{ \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \middle| \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^2 \right\}$$

- Define a 3D surface based on some level-set function:

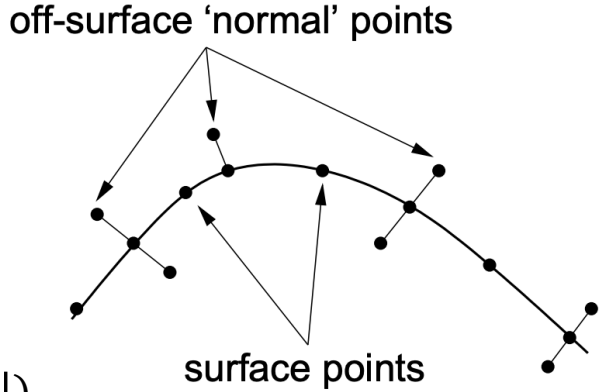
$$S_{implicit} = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \middle| f(x,y,z) = 0 \right\}$$

- Define a 3D volume of some density or opacity function:

$$V = \left\{ f(x,y,z) \middle| \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3 \right\}$$

Implicit surfaces with Radial Basis Functions

- We have a 3D points $\boldsymbol{x}_i$ with the corresponding values f_i and we want to fit a function $f(\boldsymbol{x})$ so that $f(\boldsymbol{x}_i) \approx f_i$.
  - $f_i = 0$, if $\boldsymbol{x}_i$ is on the surface
  - $f_i < 0$, if $\boldsymbol{x}_i$ is inside the object
  - $f_i > 0$, if $\boldsymbol{x}_i$ is outside the object



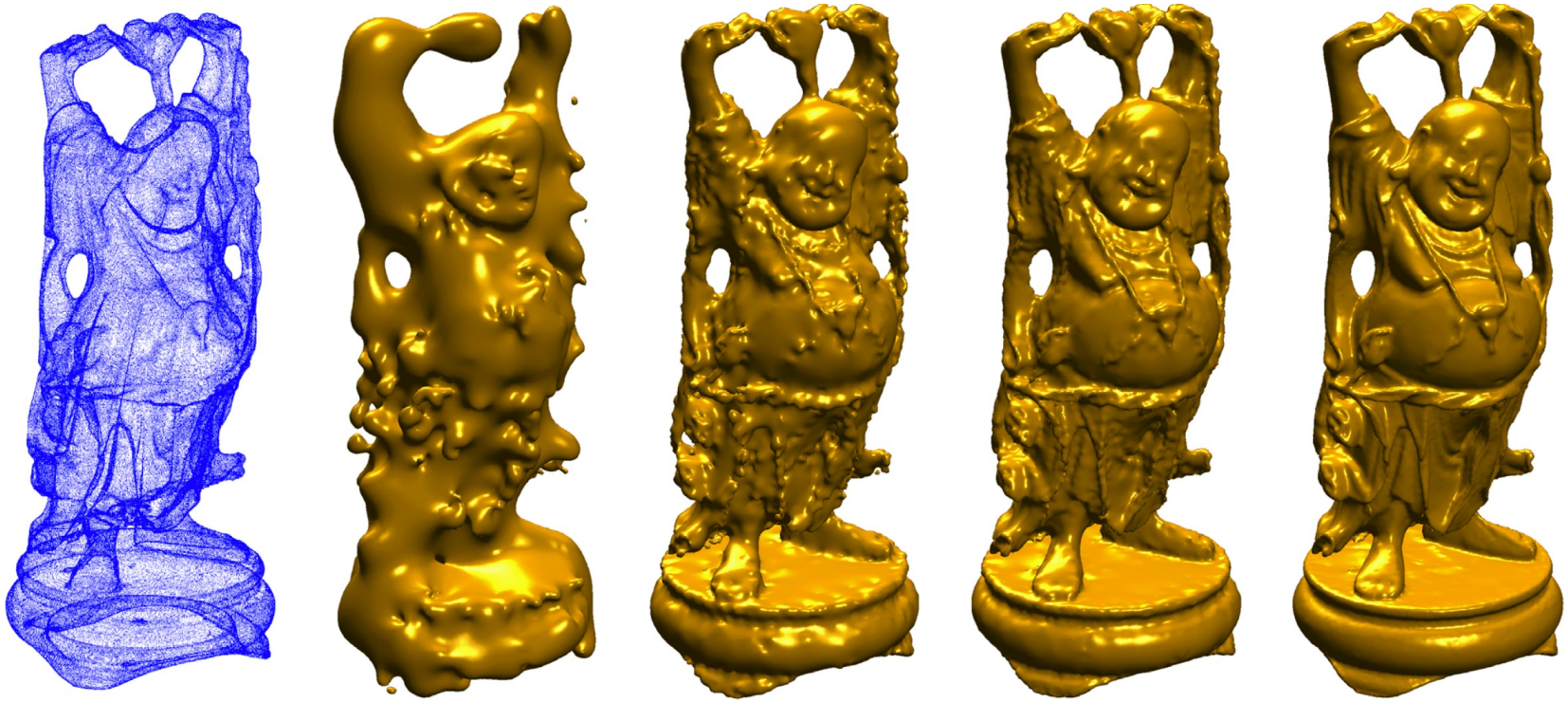
- The function has the following form:

$$f(\boldsymbol{x}) = \sum_{i=0}^{N} w_i \varphi\left(|\boldsymbol{x} - \boldsymbol{x}_i|\right)$$

- The weights, initialized to zero, are found in a greedy manner by identify points $\boldsymbol{x}_i$ with the largest errors.

Carr et al., "Reconstruction and representation of 3D objects with radial basis functions", SIGGRAPH 2001.

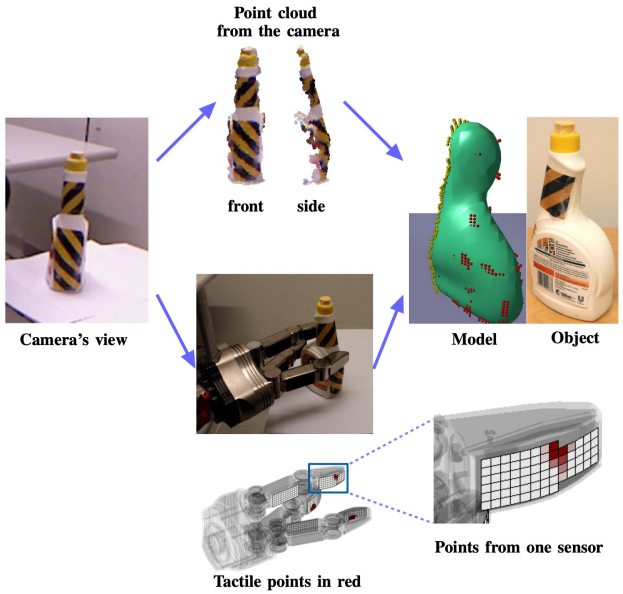
Implicit surfaces with Radial Basis Functions

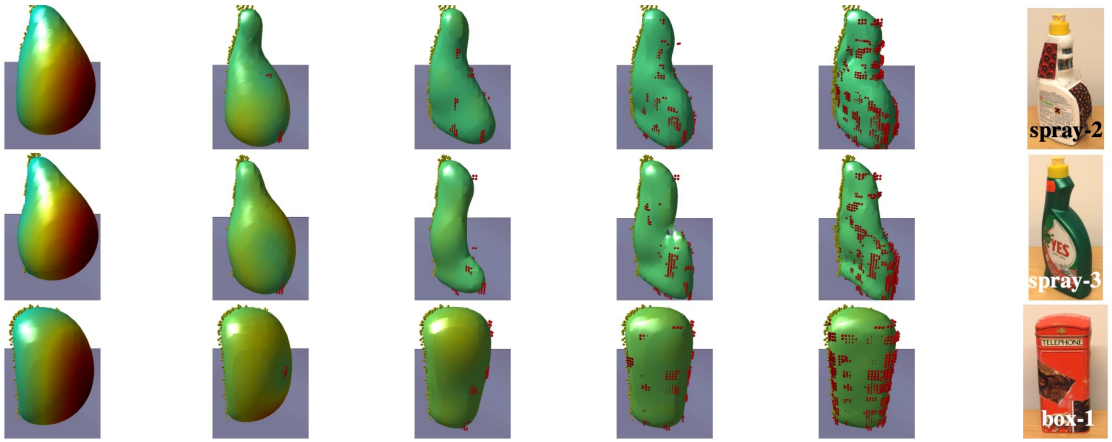


- Gradual improvement for 544K points with 80K non-zero weights.

Implicit surfaces with Gaussian Processes

- Fit positions from visual and tactile measurements to a function modelled with Gaussian Processes.
- The function includes a variance estimate that is used to guide the hand.

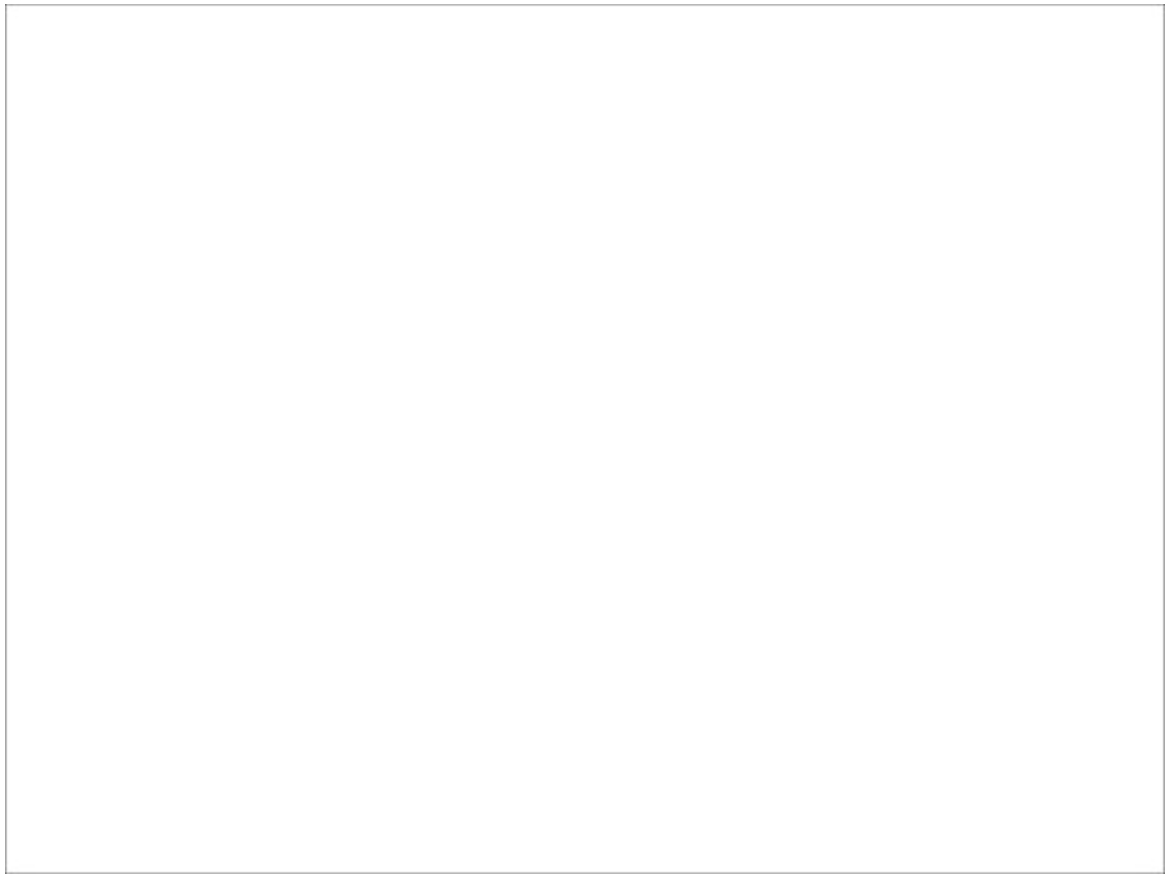




shape estimates for an increasing number of touches

Björkman et al., "Enhancing visual perception of shape through tactile glances", IROS 2013.

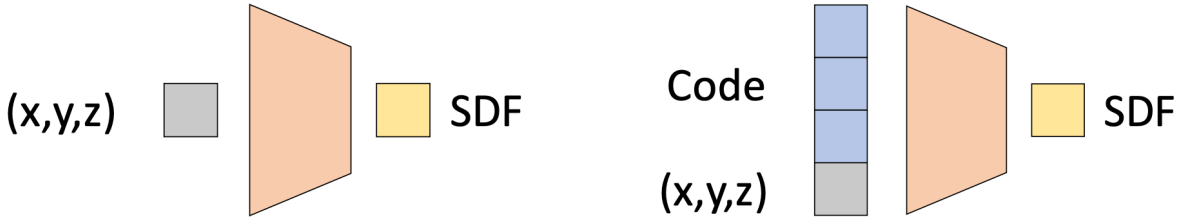
Implicit surfaces with Gaussian Processes



DeepSDF: Learning Signed Distance Functions

- Represent a signed distance function $f(\boldsymbol{x})$ using deep network, but introduce a latent shape code $\boldsymbol{z}$ and model many shapes of the size class at the same time.

$$f(\boldsymbol{z_i}, \boldsymbol{x}) = SDF_i(\boldsymbol{x})$$

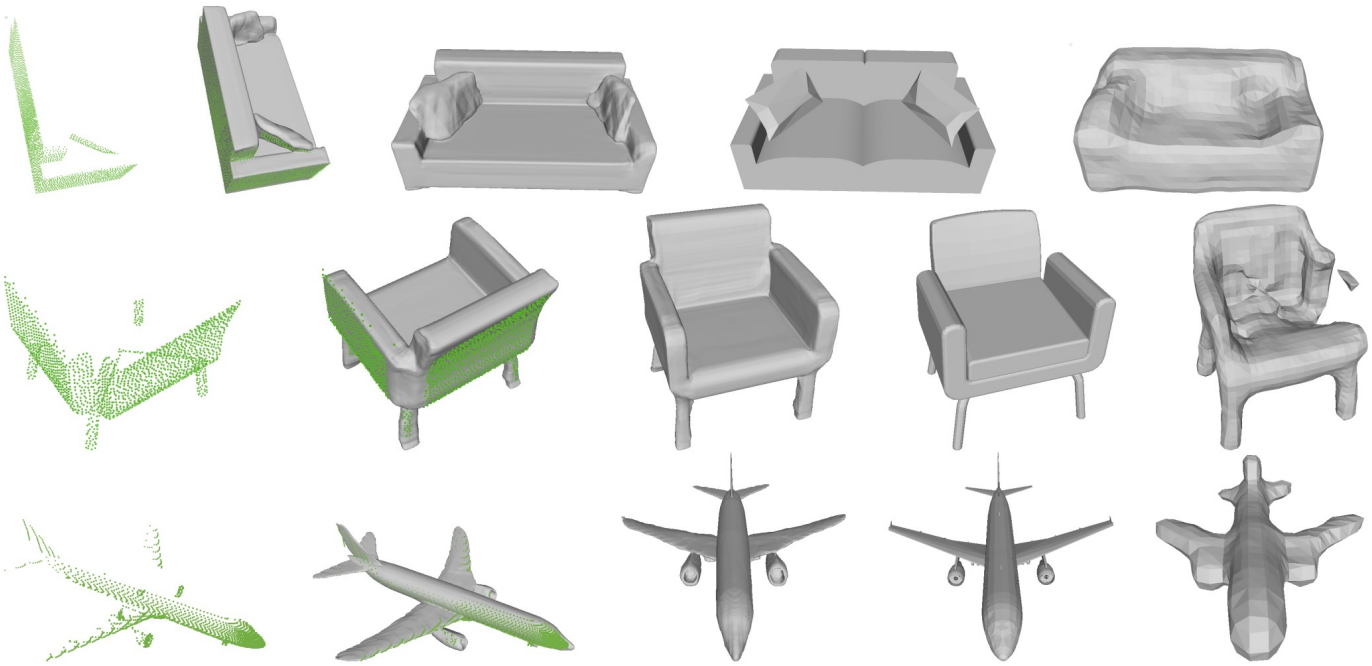


(a) Single Shape DeepSDF **(b)** Coded Shape DeepSDF

- The latent shape code $\boldsymbol{z}$ is learned with an auto-decoder using backpropagation.
- During inference after training, the shape code is estimated, which in turn gives the sign distance function.

Park et al.; ”DeepSDF: Learning continuous signed distance functions for shape representation”, CVPR, 2019.

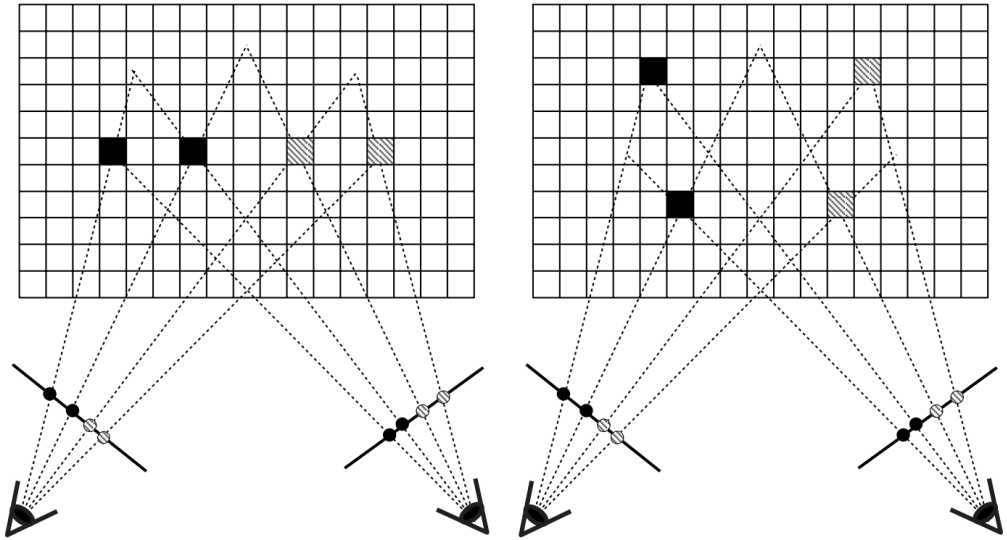
DeepSDF: Learning Signed Distance Functions

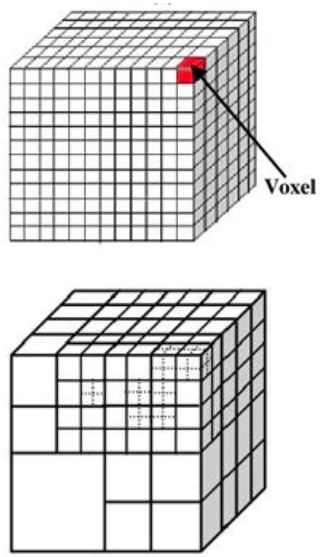


(a) Input Depth (b) Completion (ours) (c) Second View (ours) (d) Ground truth (e) 3D-EPN

- By exploiting the similarities between objects of the same class, the shape can be reconstructing from only partial observations of the shape.

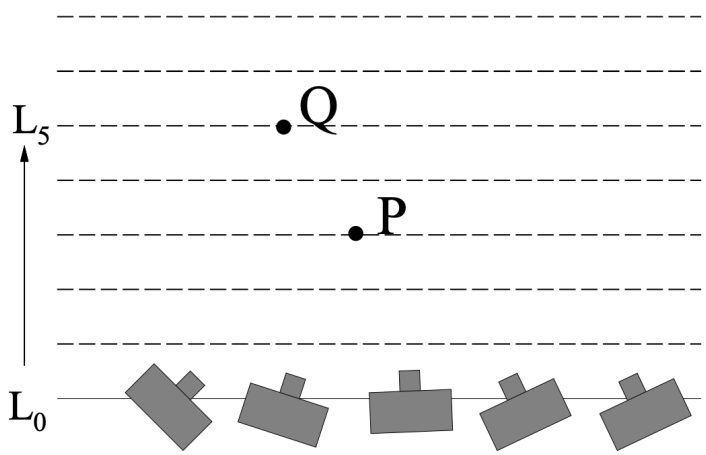
An alternative representation: voxel grids





- Fill in the occupancy and colours from a set of images from known orientations.
- 3D voxel grids can be compactly represented in terms of octrees.
- Problem: ambiguities as multiple shapes can lead to the same projections.

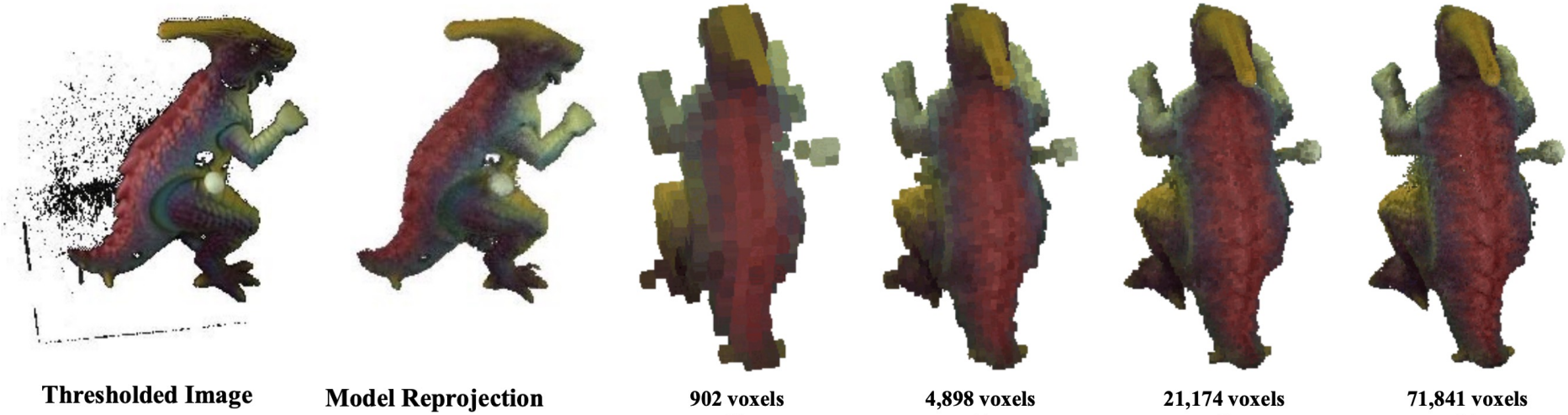
Scene reconstruction by voxel coloring



- Arrange voxels in layers and start filling in voxels from the closest layer.
- Check which *unused* pixels in all cameras each voxel is mapped to.
  - If these pixels have similar colours, the voxel is visible in all cameras.
  - Then set the colour of the voxel and mark the pixels as *used*.

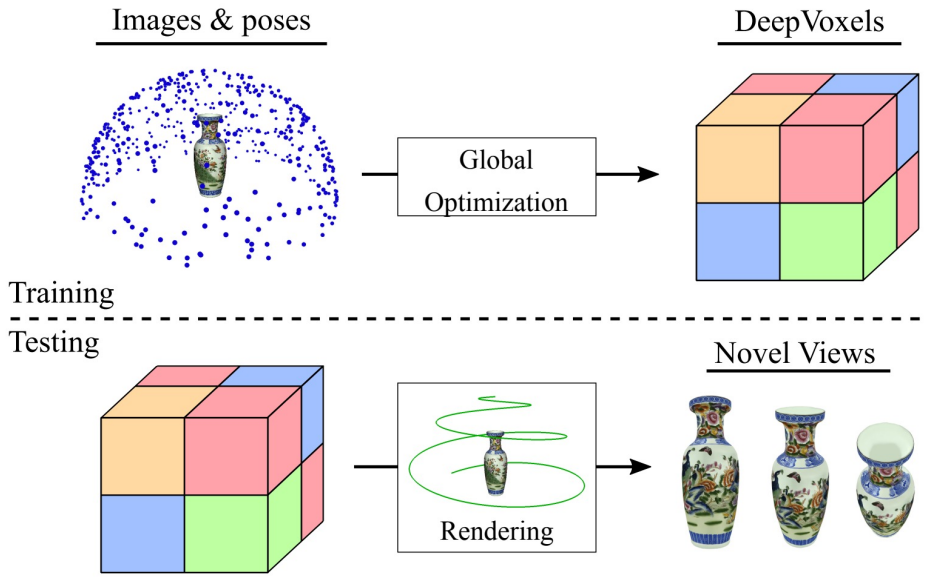
Seitz & Dyer, ”Photorealistic scene reconstruction by voxel coloring”. ICCV 1999.

Scene reconstruction by voxel coloring



- As you increase the grid resolution, the number of voxels increases rapidly.

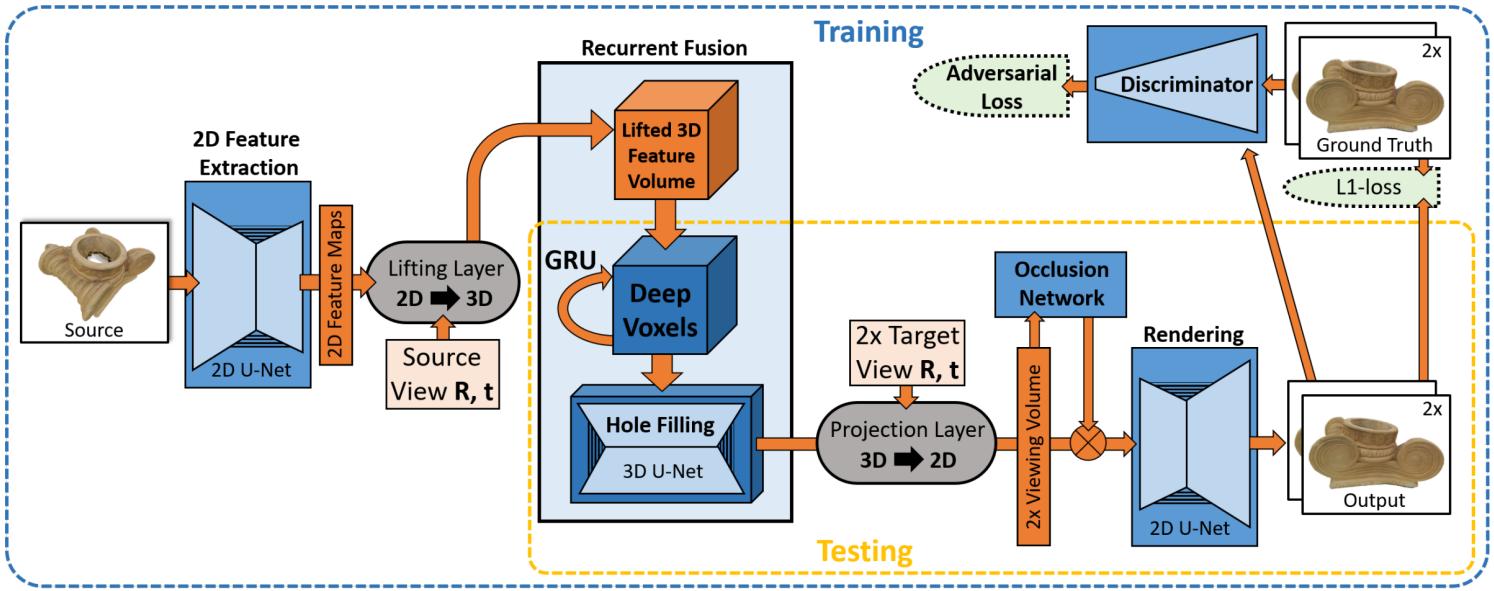
DeepVoxels: Learning 3D Feature Embeddings



- Use deep learning to learn a 3D feature representation that can be projected to create novel views of the object.
- Uses a encoder-decoder like structure with one view as input and two novel views as output, with the 3D feature representation between.

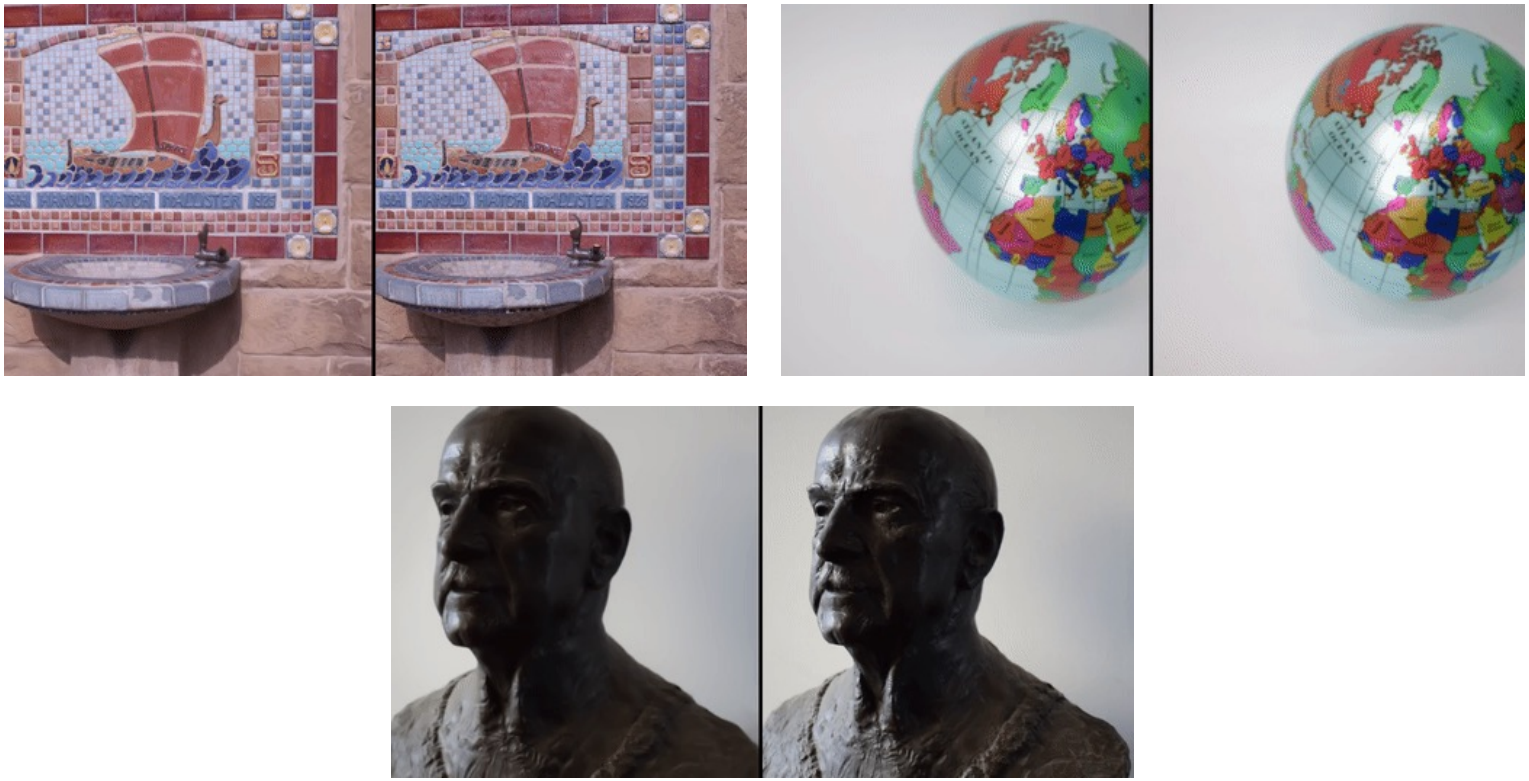
Sitzmann et al, “DeepVoxels: Learning Persistent 3D Feature Embeddings”, CVPR 2019.

DeepVoxels: Learning 3D Feature Embeddings



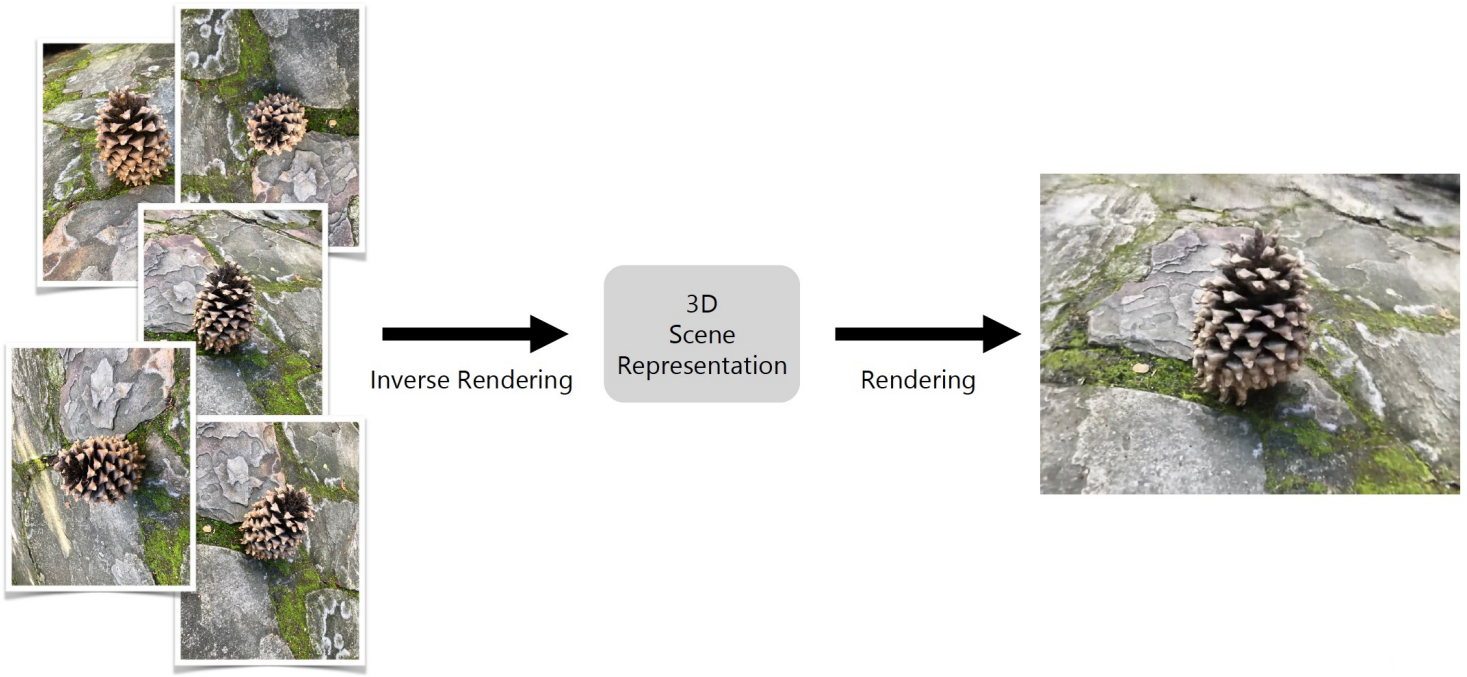
- 2D features are “lifted” to 3D: from the 3D voxels pick features from the image.
- A recurrent network updates once per new image to resolve depth ambiguities.
- An occlusion network predicts the visibility of 3D voxels from new view points.
- 3D features are projected to 2D using visibility weights to produce new images.

DeepVoxels: Learning 3D Feature Embeddings



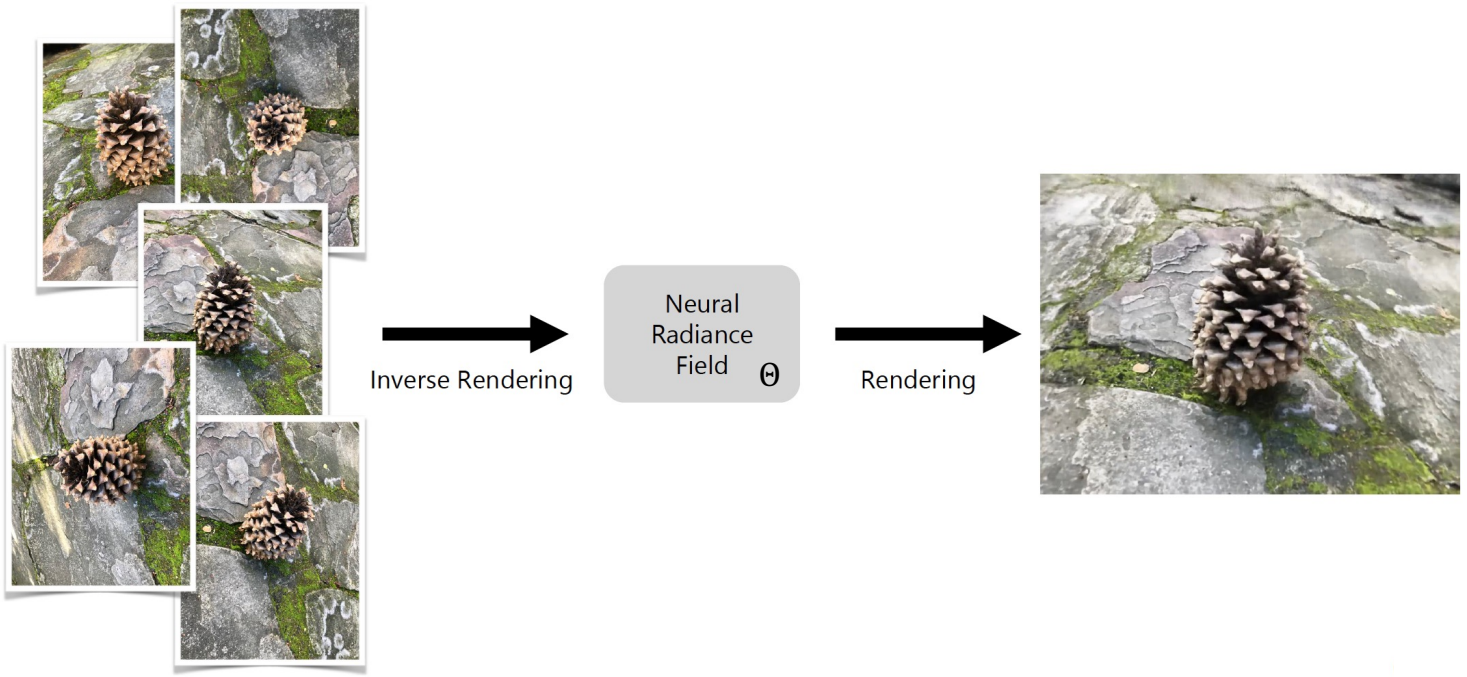
Novel views and closest ground truth views of three scenes.

Computer vision as inverse rendering



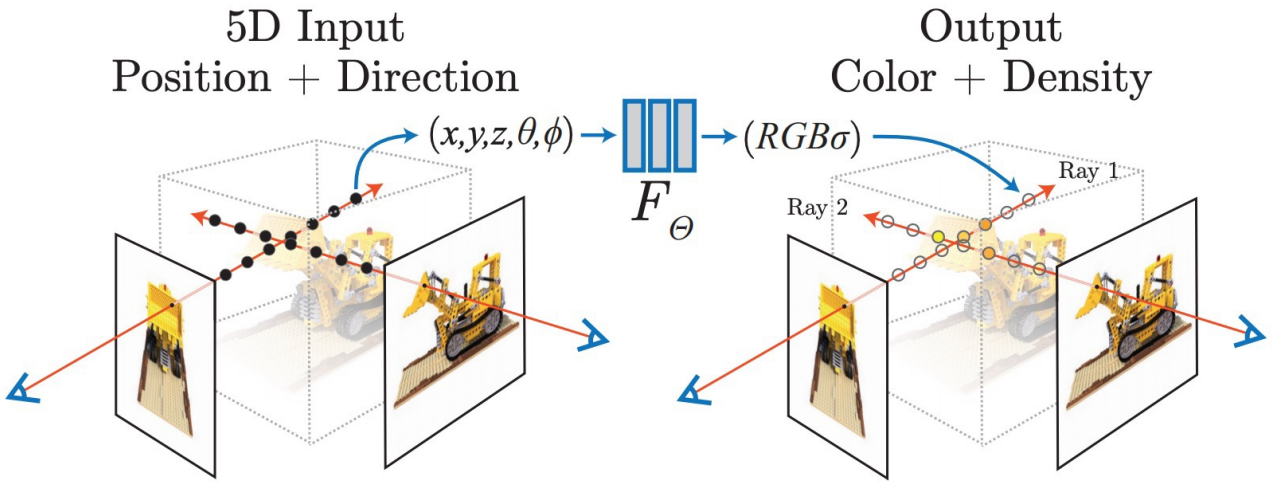
- You can view computer vision as the inverse problem of computer graphics.

Neural Radiance Fields (NeRFs) for inverse rendering



- But now the model is represented as a neural network with parameters Θ.
- We can view it as an encoder-decoder structure that can be trained.

Volume rendering in NeRF



- Most modern NeRF-like methods have a structure similar to the above.
- Input: position $\boldsymbol{x} = (x, y, z)$ and direction $\boldsymbol{d} = (\theta, \phi)$
- Output: colour $\boldsymbol{c} = (R, G, B)$ and density or opacity σ
- Using the learned network you can render images from any direction.

Mildenhall, ”NeRF: Representing scenes as neural radiance fields for view synthesis”, Commun. ACM, 2021.

Naïve implementation produces blurry results

NeRF (Naive)

NeRF (with positional encoding)

Slide credit: Jon Barron

Positional encoding

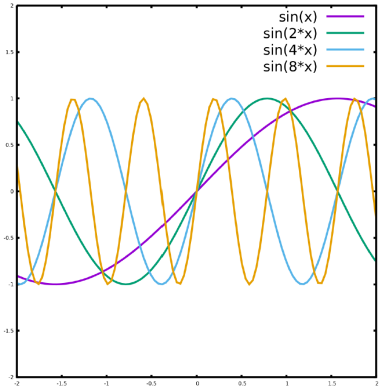
- Neural networks (and many other ML methods) have the tendency to learn the low-frequency components of functions first.
- This causes problems when the high-frequency content of the signal is equally important, or even more important.
- To avoid this, methods often use positional encoded representations

$$\gamma(\boldsymbol{x}) = [\cos(M\boldsymbol{x}), \sin(M\boldsymbol{x})]$$

where

$$M = [I \quad 2I \quad 2^2 I \quad \cdots \quad 2^{L-1} I]$$

- The input of the network is thus $(6L_x + 4L_d)$-dimensional, instead of $(3 + 2)$-dimensional, e.g. $L_x = 10, L_d = 4$.



"Positional encoding" of a number $\boldsymbol{x}$

Vaswani et al., "Attention is All you Need", NeurIPS, 2017.

Volume rendering is trivially differentiable

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

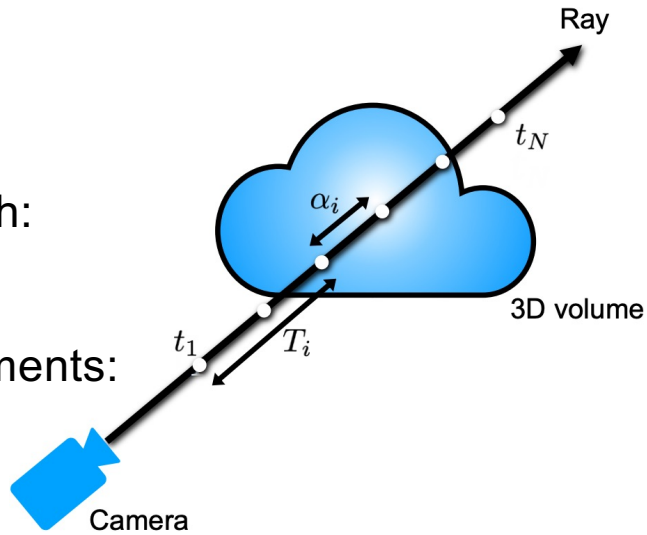
$$\boldsymbol{c} \approx \sum_{i=1}^{N} T_i \, \alpha_i \boldsymbol{c}_i$$

- How much light does ray segment i contribute with:

$$\alpha_i = 1 - e^{-\sigma_i |t_i - t_{i-1}|}$$

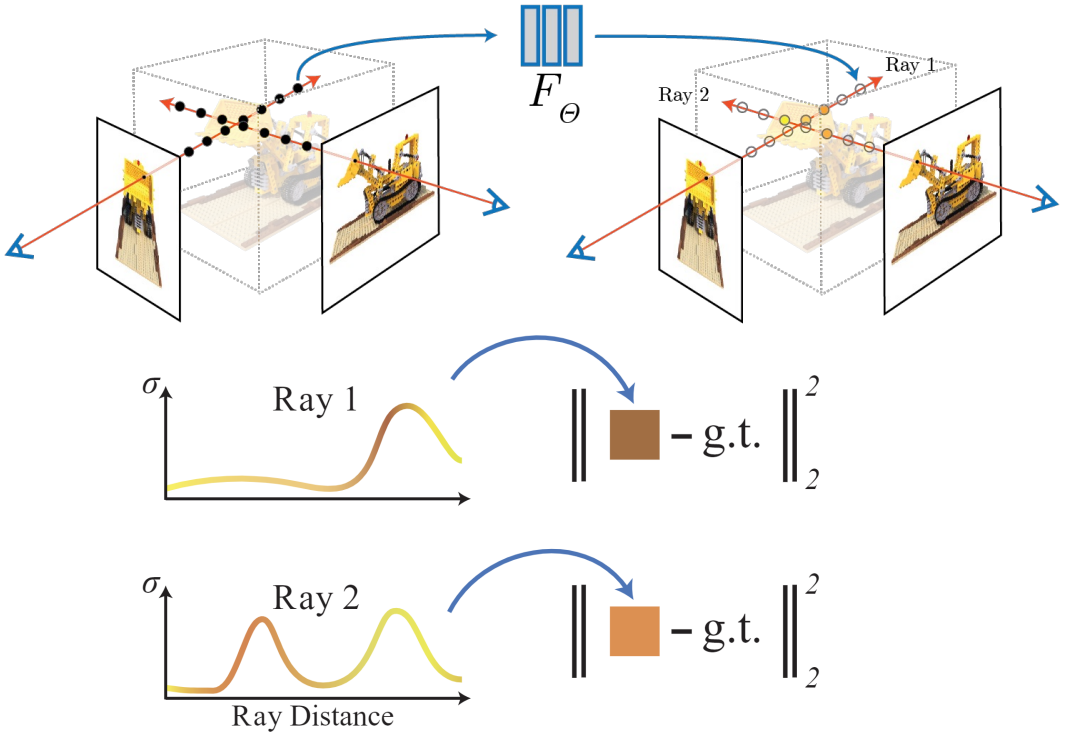
- How much light will not be blocked by earlier segments:

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



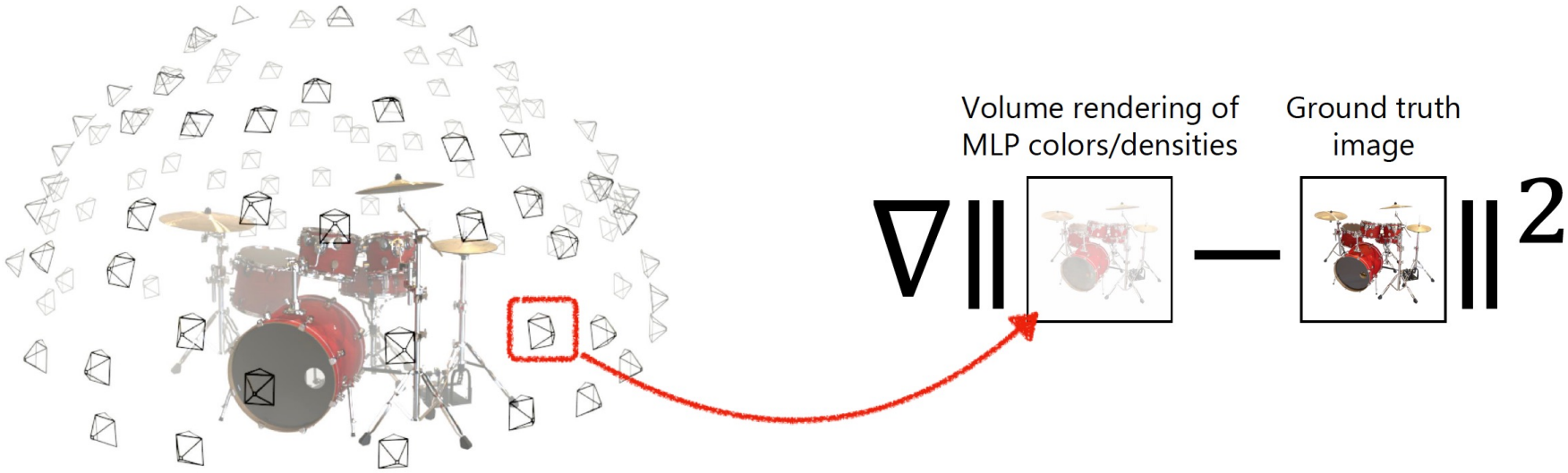
Slide credit: Jon Barron

Putting it all together



Aggregate predicted colours along each rays, using the predicted densities.

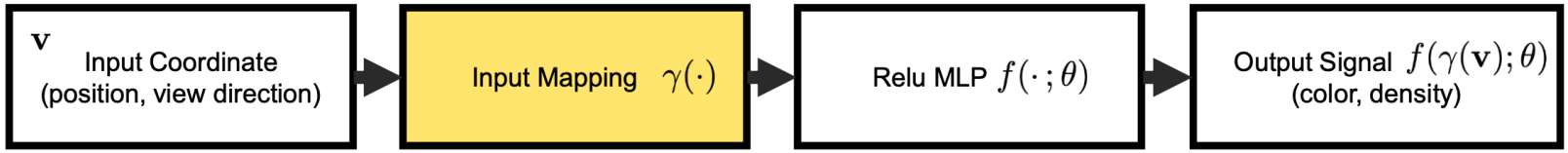
Training the NeRF



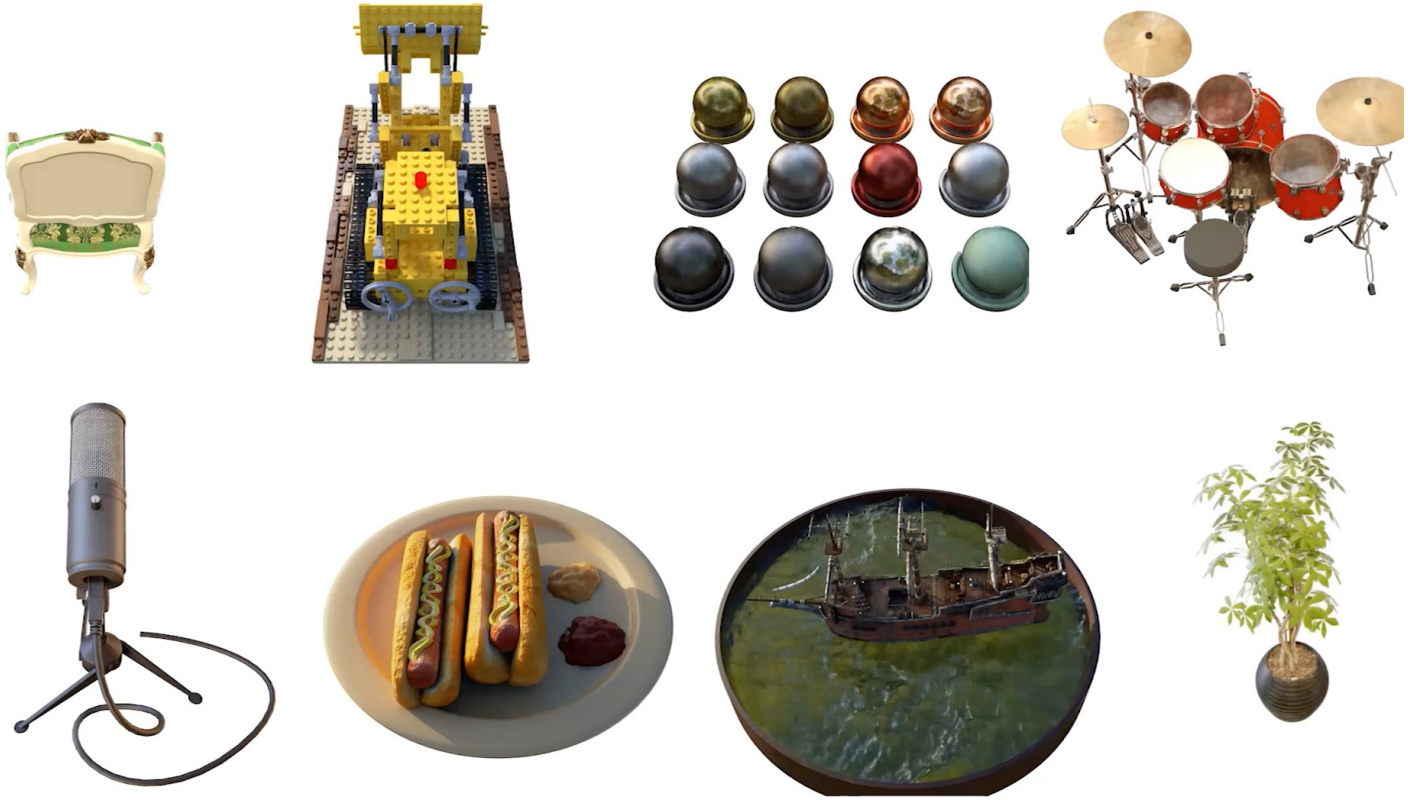
Train network using gradient descent to reproduce all input views of the scene.

NeRF summary

- Represent the scene as volumetric coloured “fog”.
- Store the fog colour and density at each point as an MLP mapping 3D position $\boldsymbol{x}$ and direction $\boldsymbol{d}$ to colour $\boldsymbol{c}$ and density σ.
- Render an image by shooting rays through the fog for each pixel.
- Optimize MLP parameters by rendering to a set of known viewpoints and comparing to ground truth images.

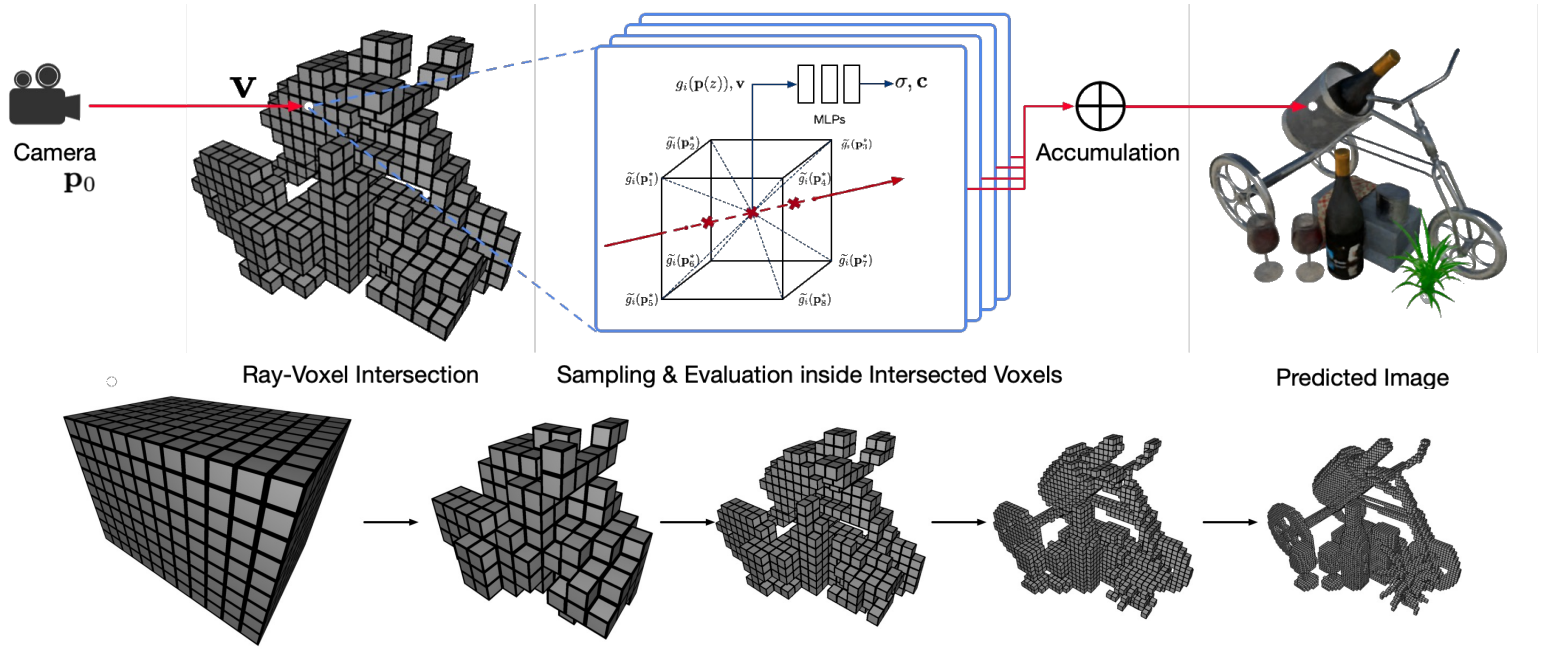


NeRF results: synthetic data



NeRF results: real world data

Neural sparse voxel fields



- Comeback of the voxels: each voxel has a small multi-layer perceptron (MLP).
- Progressive training from coarse to fine voxels resolutions.
- Rendering speed is about 10 times faster than NeRF.

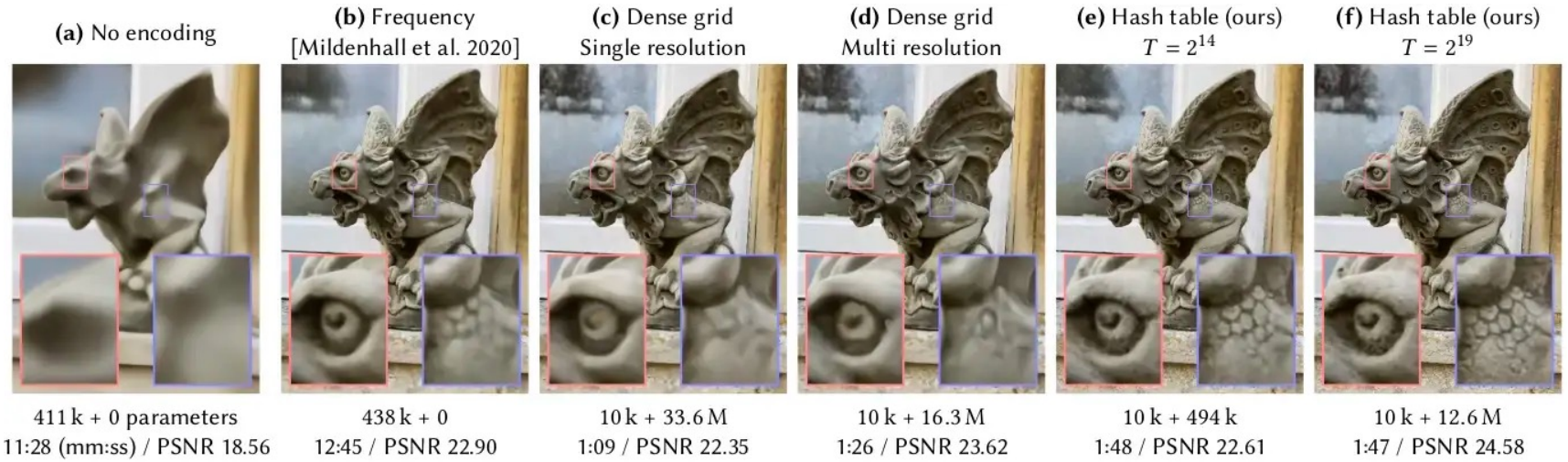
Liu et al., “Neural Sparse Voxel Fields”, NeurIPS 2020.

Neural sparse voxel fields

Interactive editing

NSVF

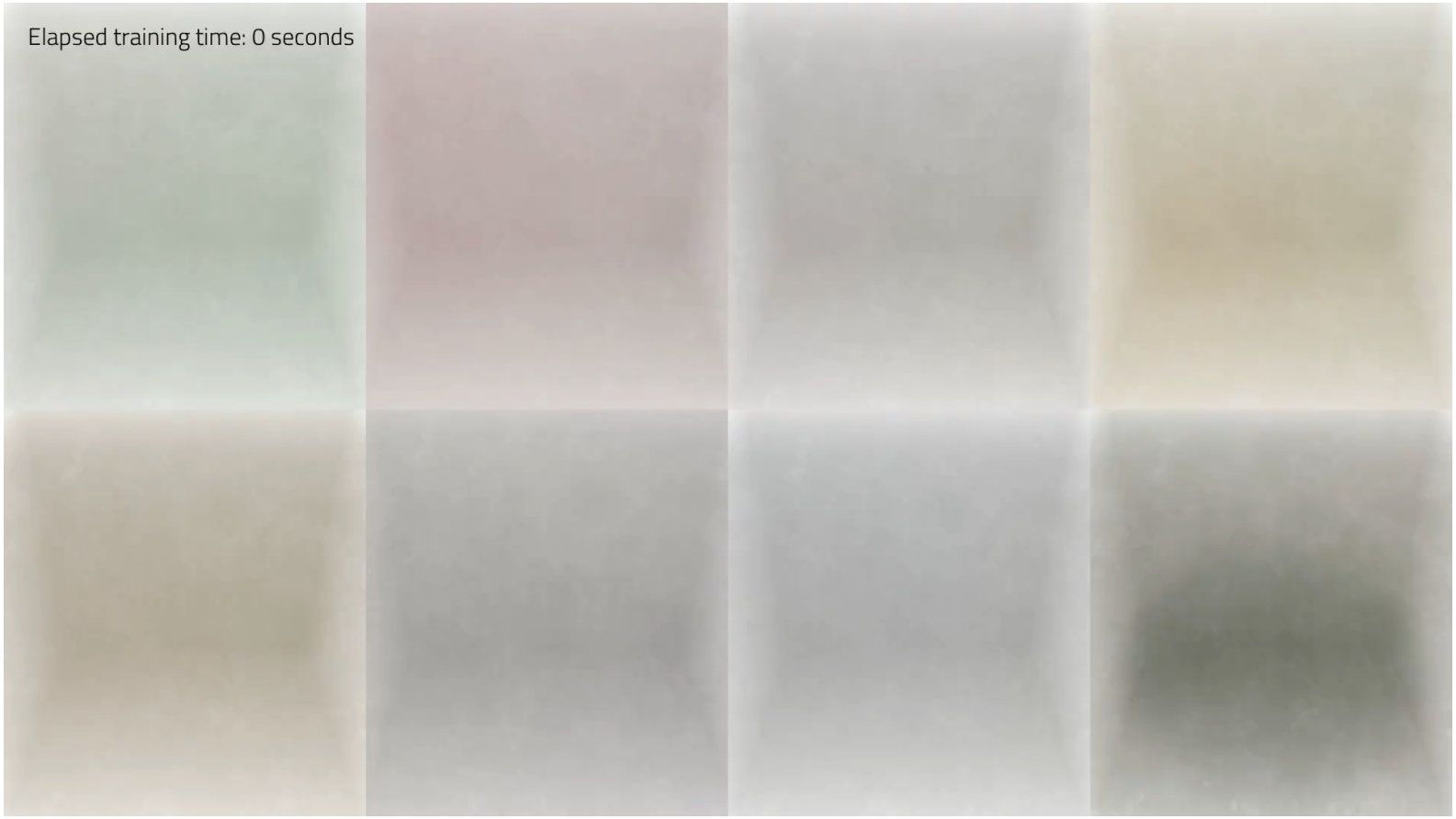
Instant neural graphics primitives



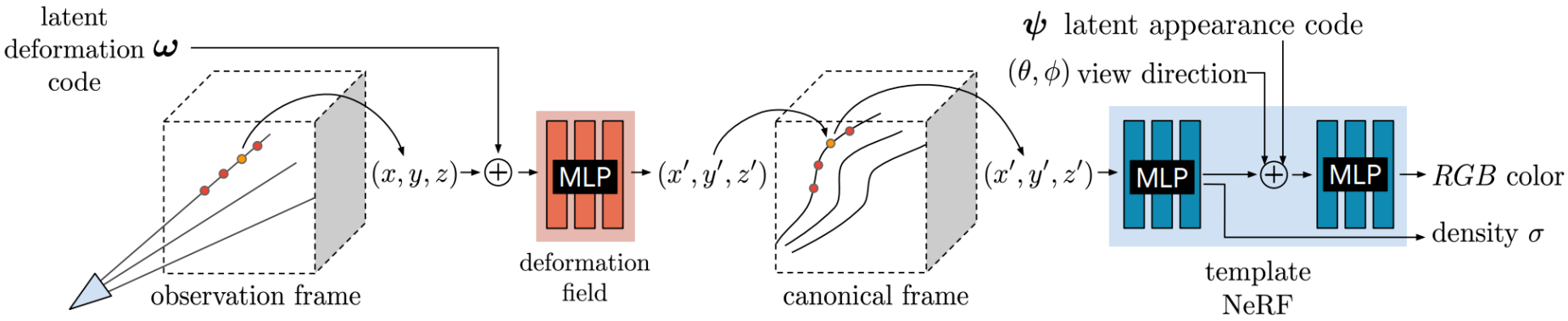
- Idea: let voxels in a multi-resolution grid share MLPs, indexed through hashing.
- Since the surface covers only a fraction of all voxels, collisions are few.
- This allows for a much finer grid, if there is enough details to be represented.
- Training is exceptionally fast compared to earlier methods.

Müller et al., "Instant neural graphics primitives with a multiresolution hash encoding.", *SIGGRAPH,* 2022.

Instant neural graphics primitives



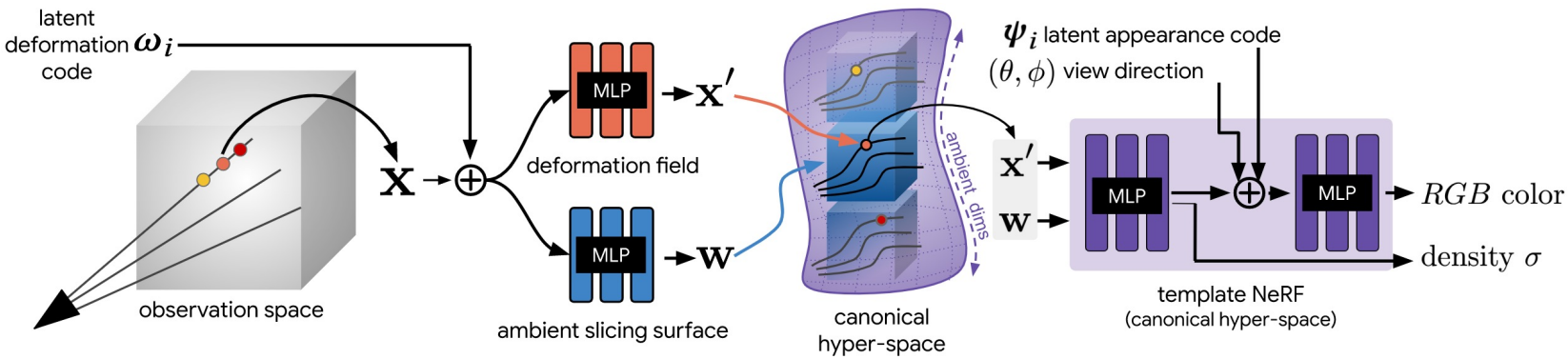
Nerfies: Deformable NeRFs



- How to change the scene over time?
- Instead of deforming the scene, deform the light rays.
  - A deformation code ω is used to deform rays to a canonical representation.
  - The canonical representations is trained similarly to a normal NeRF.
  - An appearance code ψ allows changes in illumination, etc.

Park et al., "Nerfies: Deformable neural radiance fields", ICCV 2021.

HyperNeRF: Topologically varying NeRFs



- Just bending the rays does not allow any topological changes.
- Solution:
  - Use a canonical representation that is 2 dimensions higher.
  - Use an extra MLP to generate a slicing code $\mathbf{w}$ to cut the canonical representation.
- This allows for much more flexibility, including topological changes.

Park et al., “HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields”, SIGGRAPH 2021.

Nerfies versus HyperNeRF

(a) Input Video (b) Nerfies (c) HyperNeRF

As can be seen the canonical representation is much more stable.

Examples of topological changes

What about the exam?

- What about the exam?
  - It will be held on campus!
  - Saturday 13 January 2023, 09:00-14:00
- Three kinds of questions
  - C: ~~Concept questions~~ (in the online quiz)
  - P1: Easier problem questions
  - P2: More complex questions
- Exam registration is needed. If not registered, contact EECS service center.
- Allowed tools: calculator and mathematical handbook (e.g. Beta)

And the online quiz?

- We will have the online quiz tomorrow and the day after.
  - It will become available in Canvas between 15.00 and 19.00.
  - Stay online in Zoom during the quiz.
  - Select your date of choice in the calendar.
- We will have the first catch-up sessions for both labs and quiz after the exam.
  - But probably more sessions around the re-exam in April.
- If you failed the quiz (the first time), attend the exam anyway.
  - There will be more opportunity to finish the quiz.

Labs and exercises

- Laboratory exercises:
  - Reread what you did. What were you supposed to have learned?
  - It will help you on both practical and theoretical parts of the exam.
- Exercise sessions:
  - Go through problems from exercise sessions and earlier exams!
  - Likely that something similar is on the exam.

AND, please fill in “Course evaluation form”!

Summary of good questions

- What is novel view synthesis?
- What kind of surface and volume representations exist?
- What is a sign distance function?
- What is a voxel grid?
- How does volume rendering work?
- What are the major components of NeRF?
- How can you make NeRFs dependent on time?
- What should I buy the kids for Christmas?

Recommended reading

- Szeliski: Chapters 12.7, 13.5 and 14.6