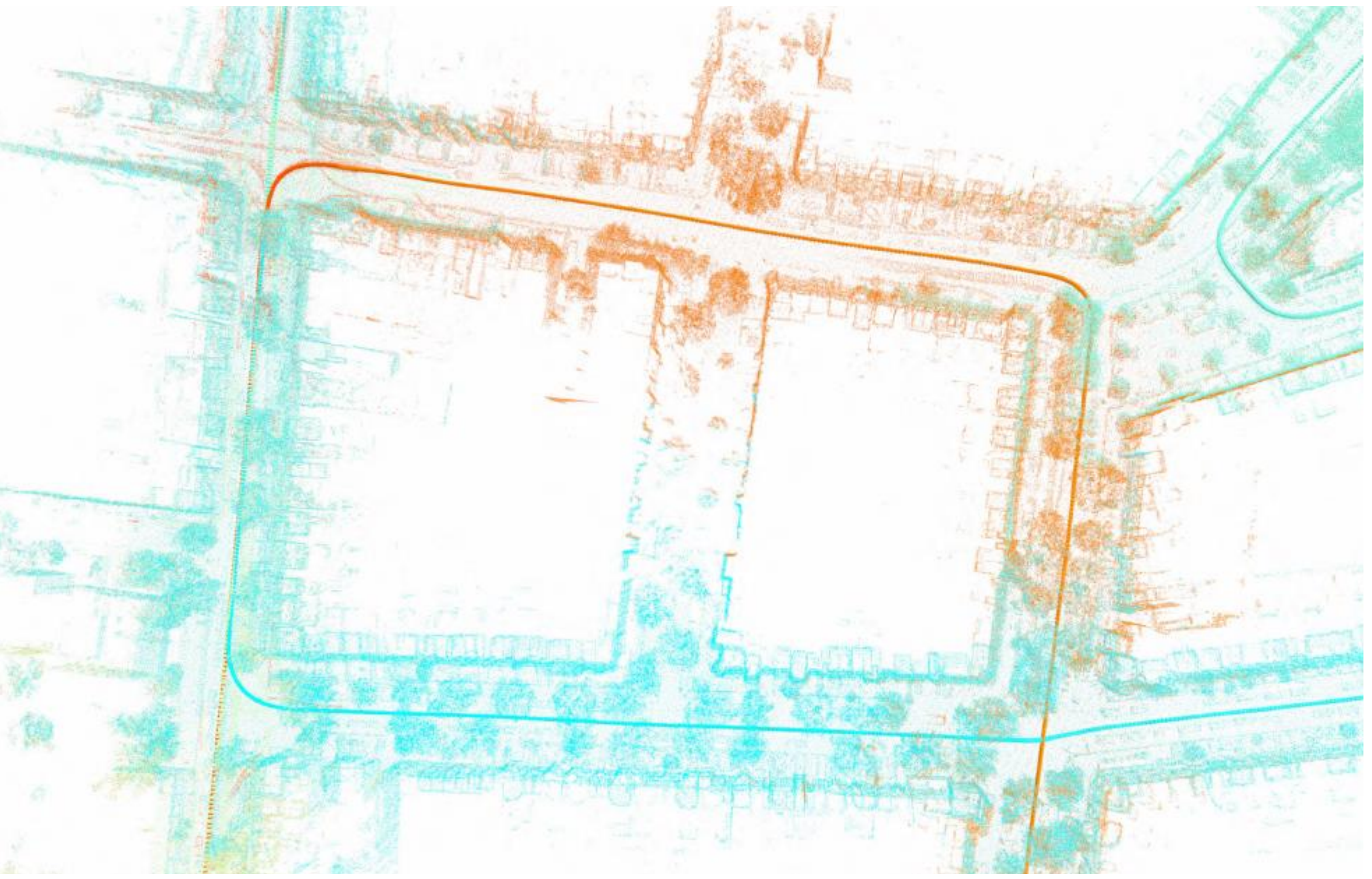
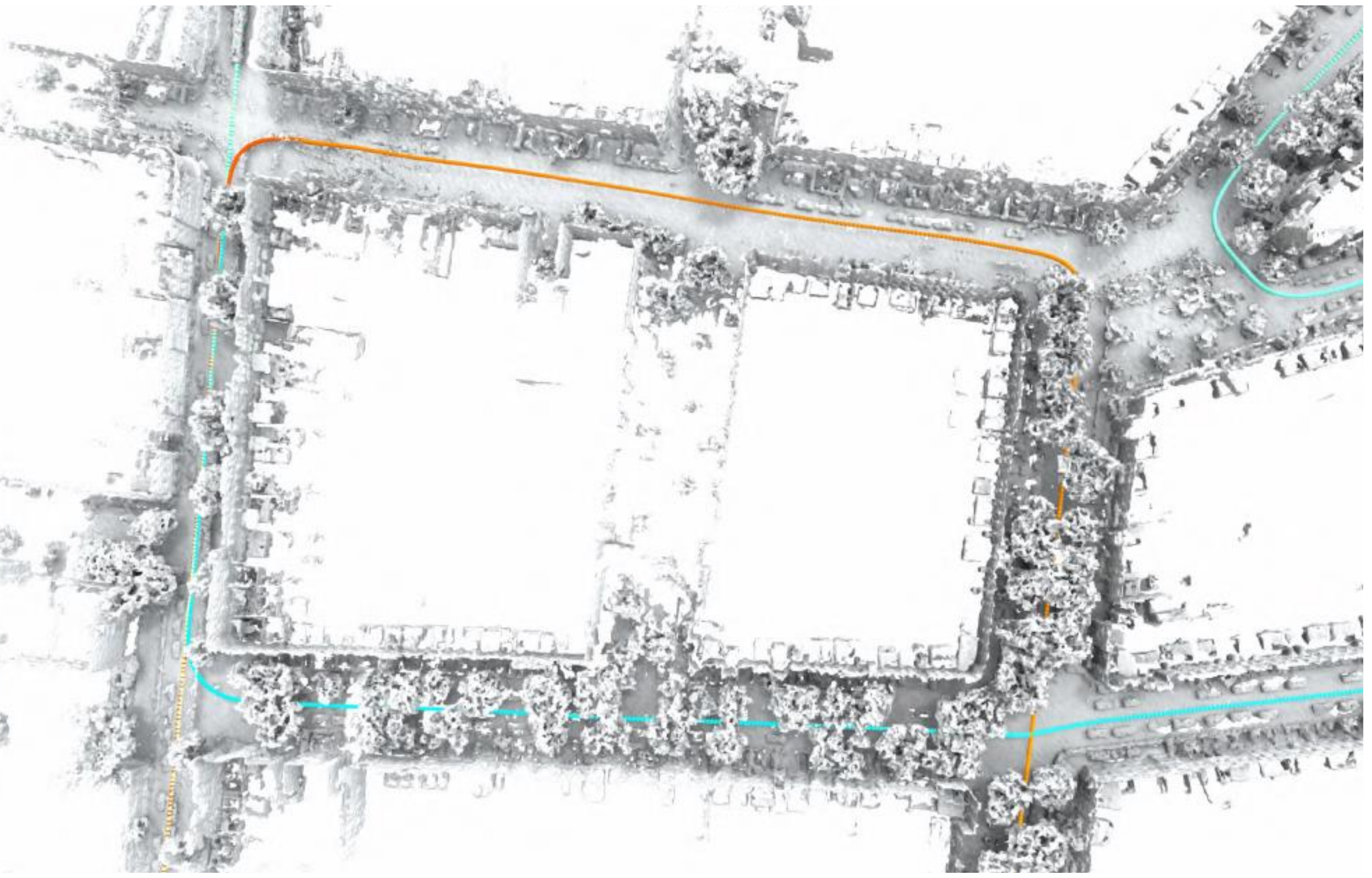
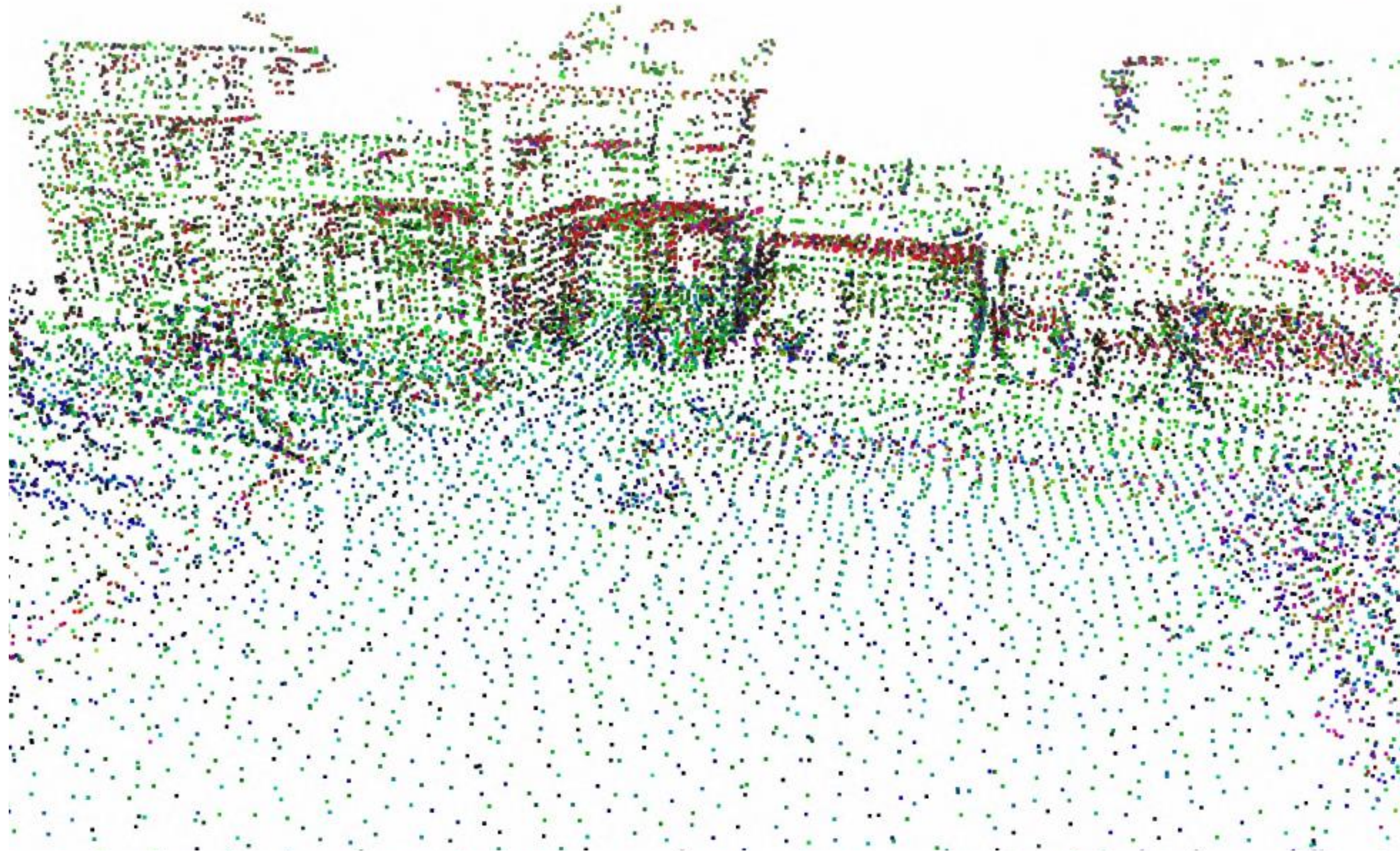




# Signed Distance Function and 3D Dense SLAM

Yulun Tian
(yut034@ucsd.edu)

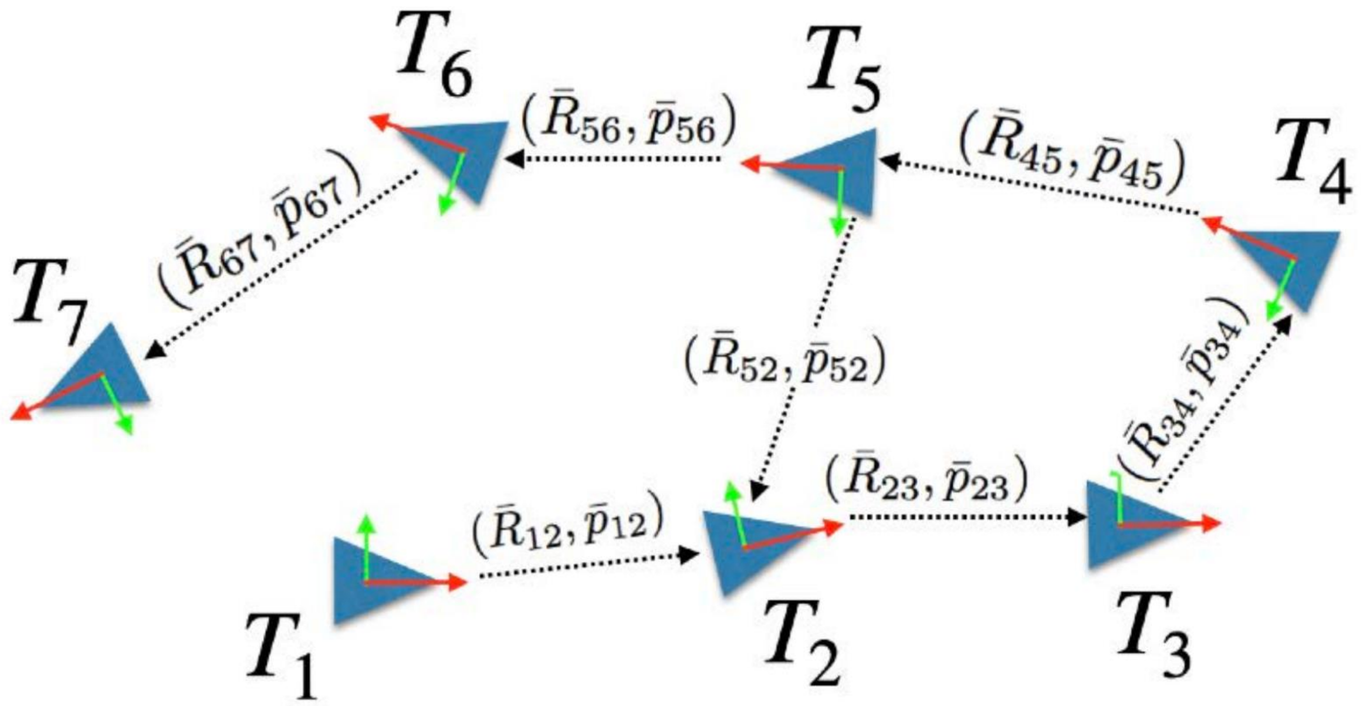
Image credit: Pan et al., "PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency," IEEE T-RO, 2024.

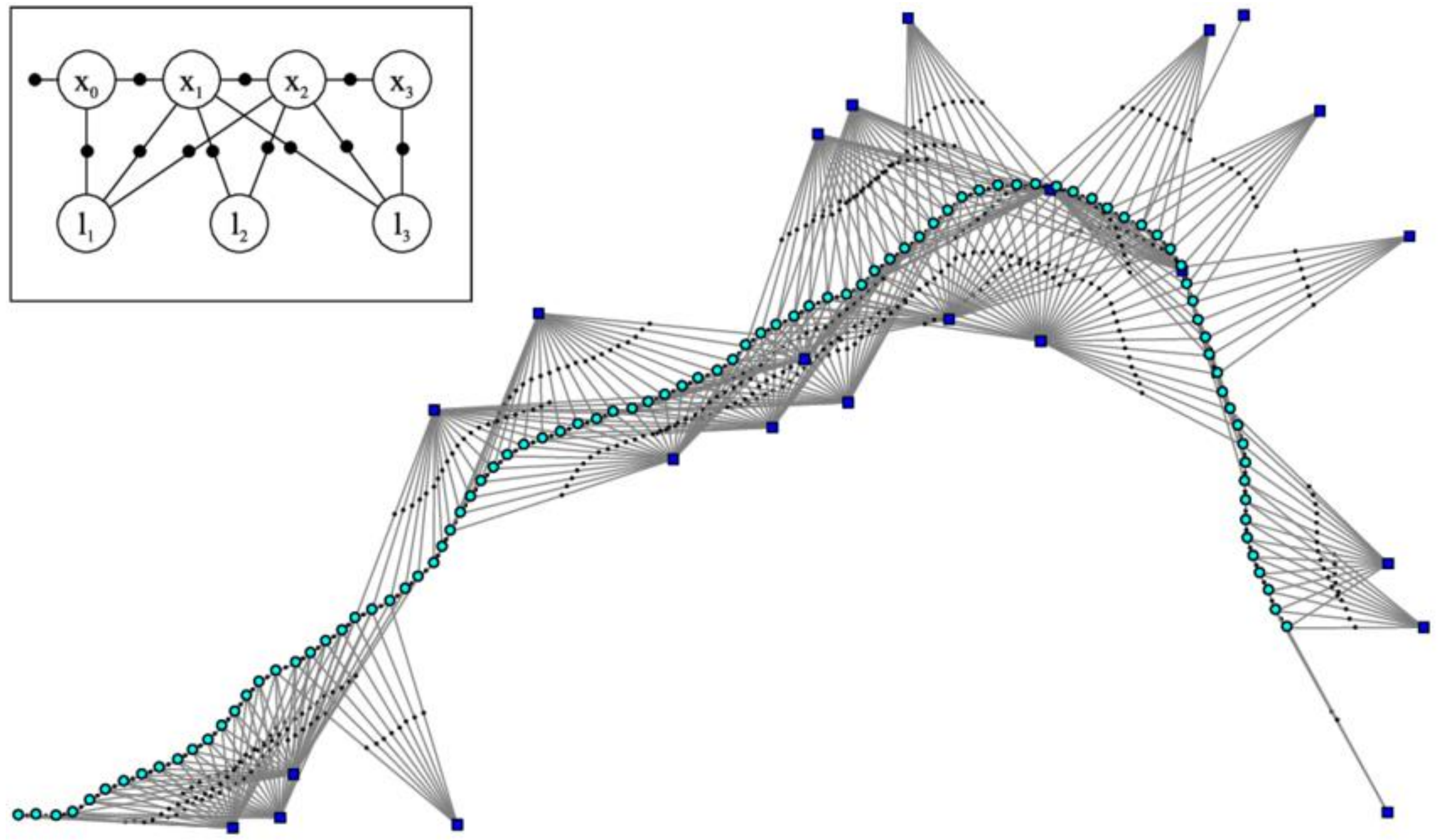
EXISTENTIAL ROBOTICS LABORATORY

# Sparse vs. Dense World Representations

- Sparse: e.g., pose graphs, landmarks

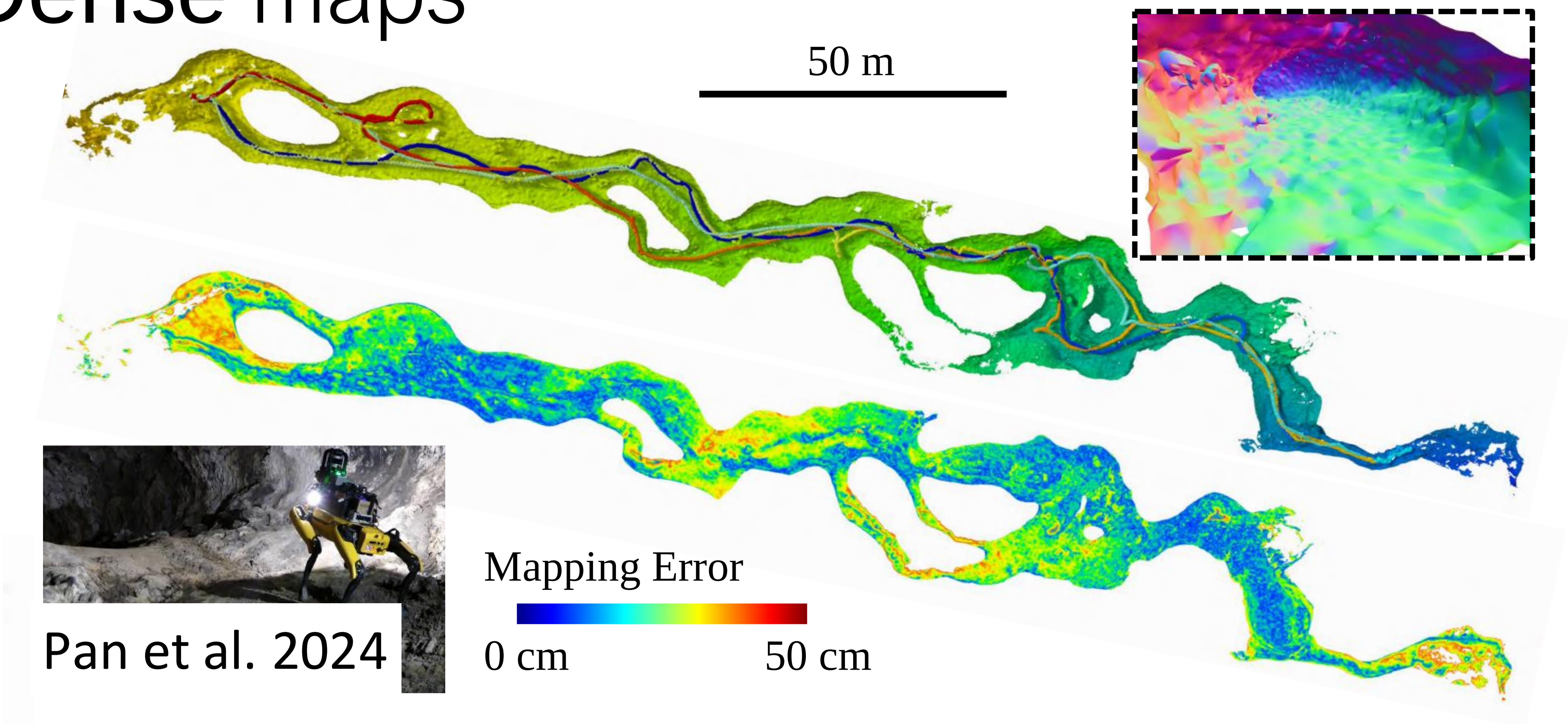
▶ **Pose graph**

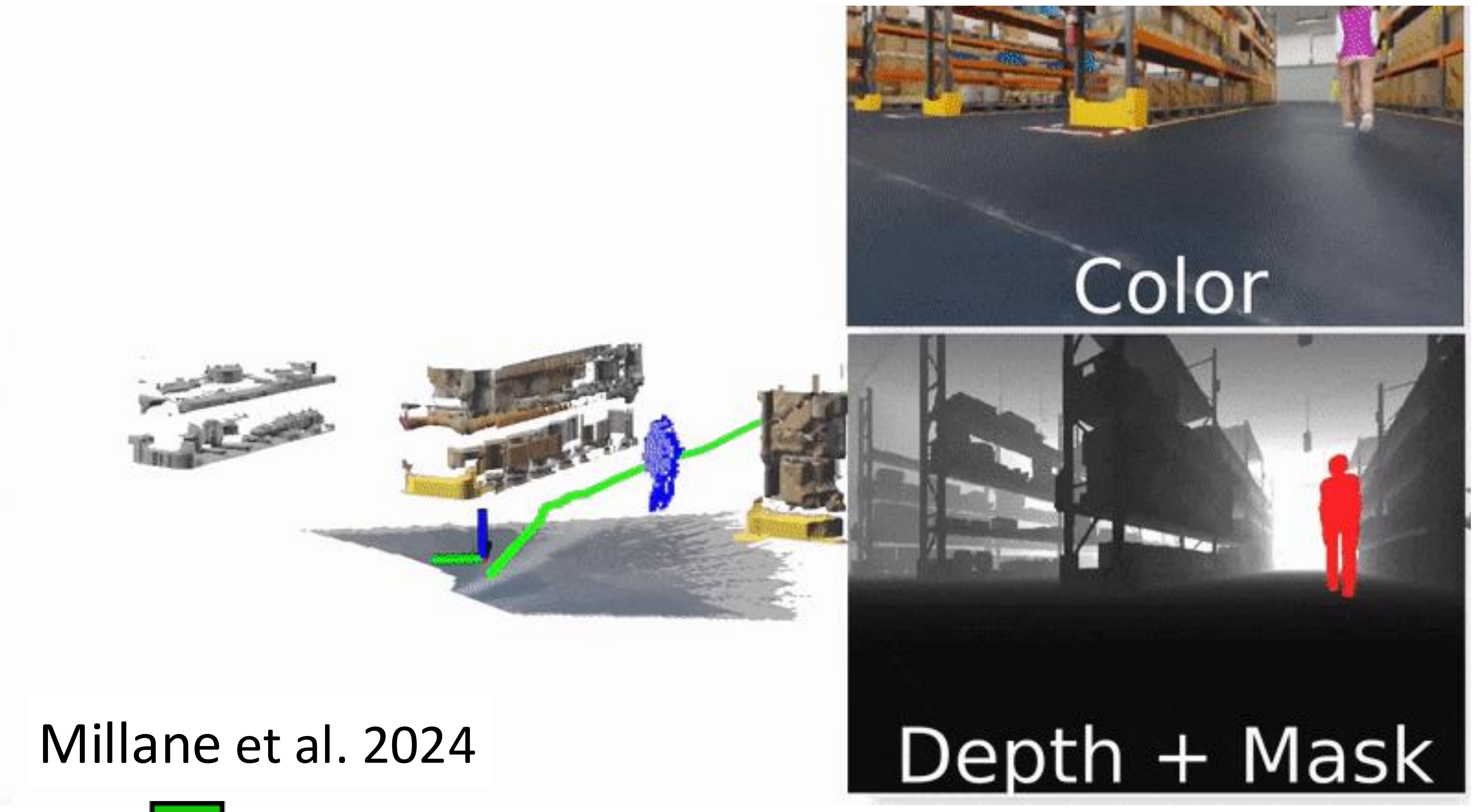



✅ Good for state estimation (e.g., localization)

- **Dense** maps



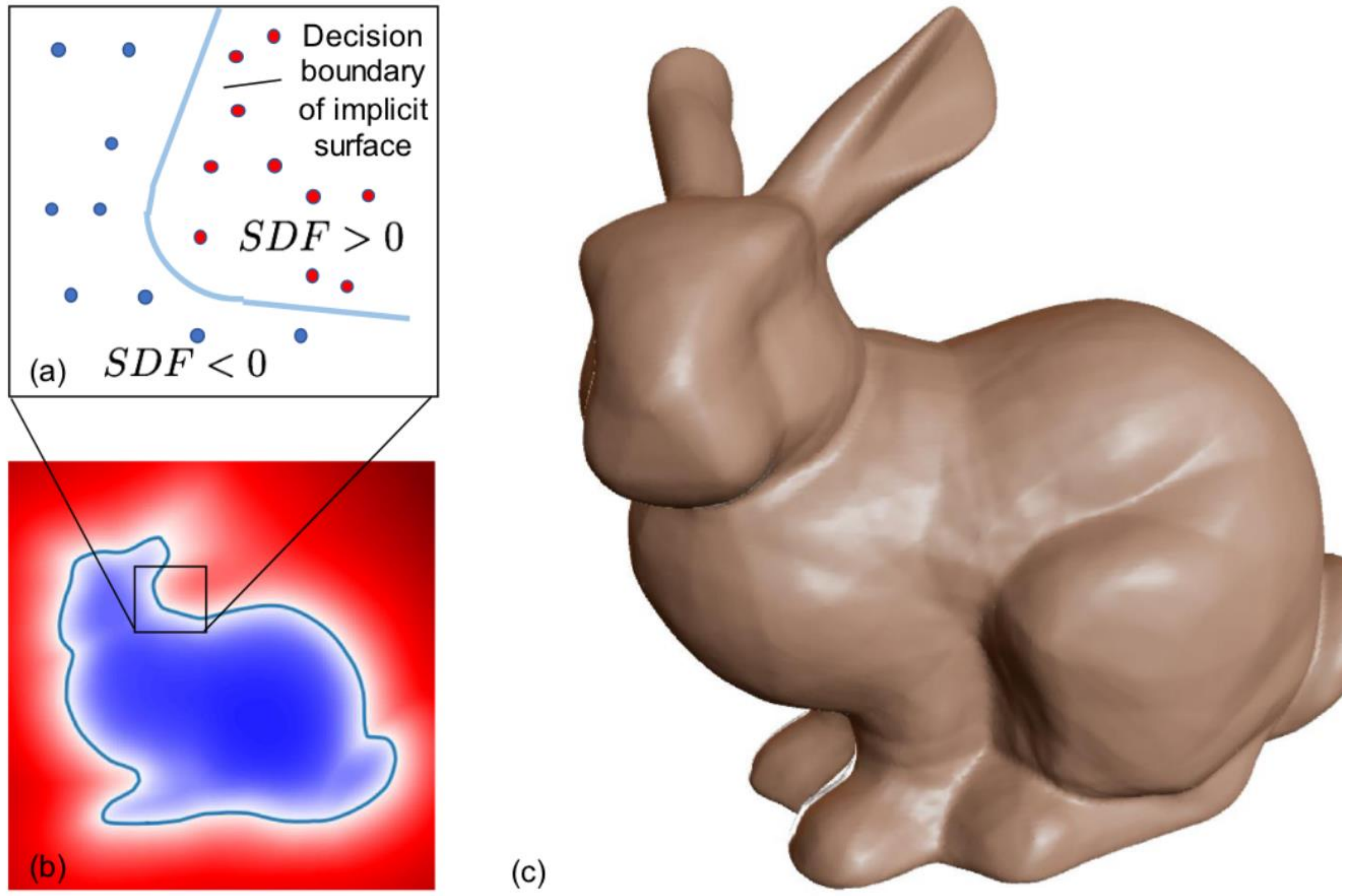




✅ Higher fidelity reconstruction

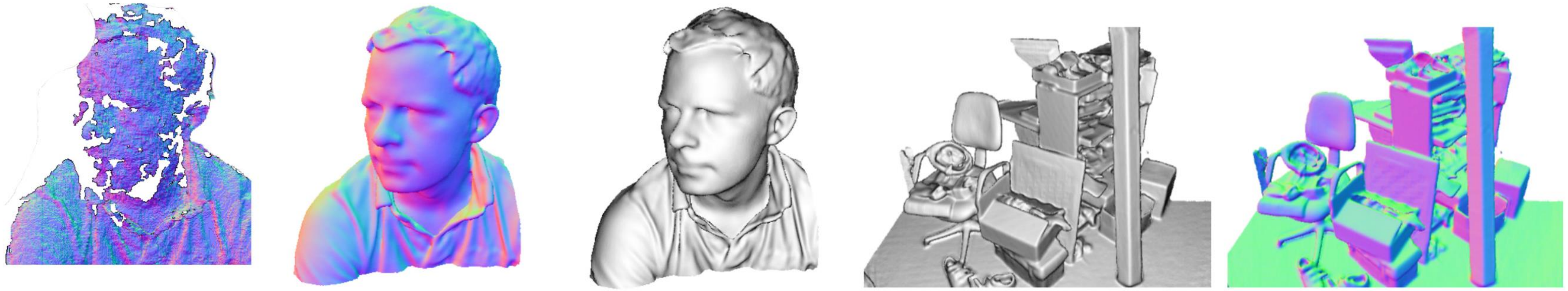
🤔 Real-time?

# Today's Lecture

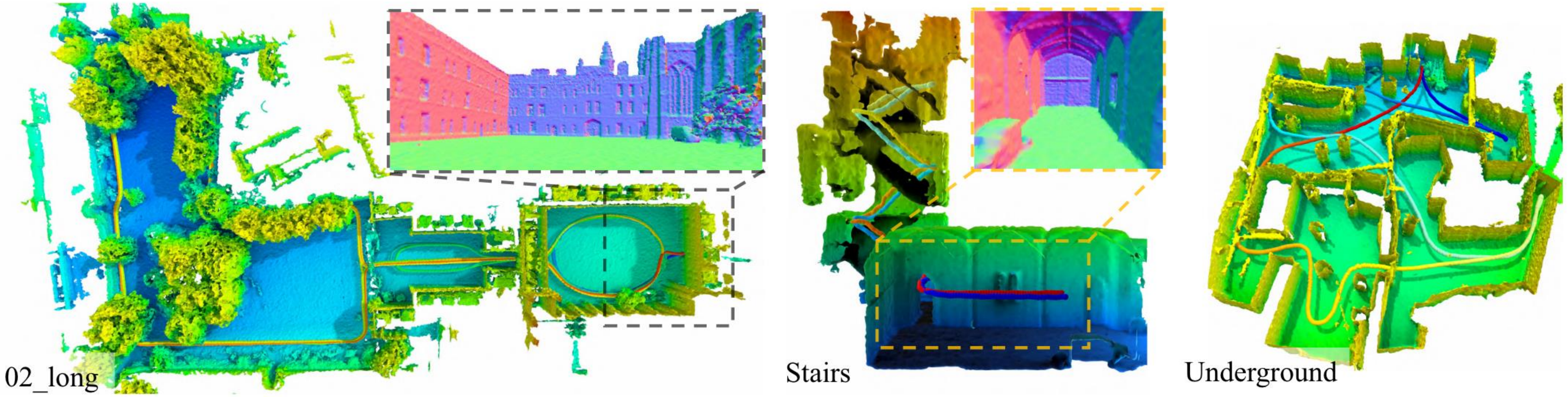
- Dense signed distance function (SDF) representation and properties
- Basics of 3D dense SLAM using SDF
- Recent advancements to improve SDF-based SLAM




Park et al. 2019

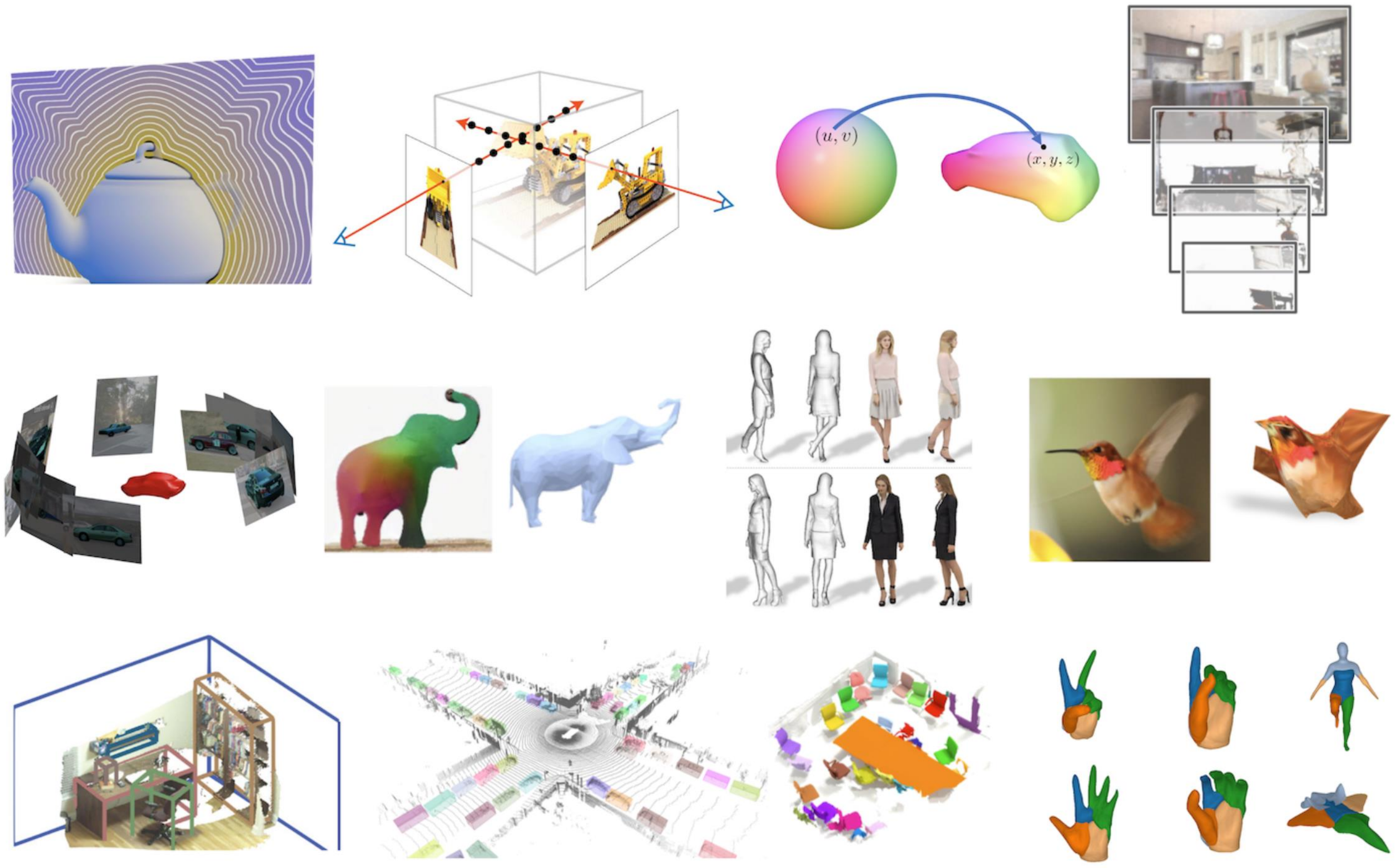


Newcombe et al. 2011




Pan et al. 2024

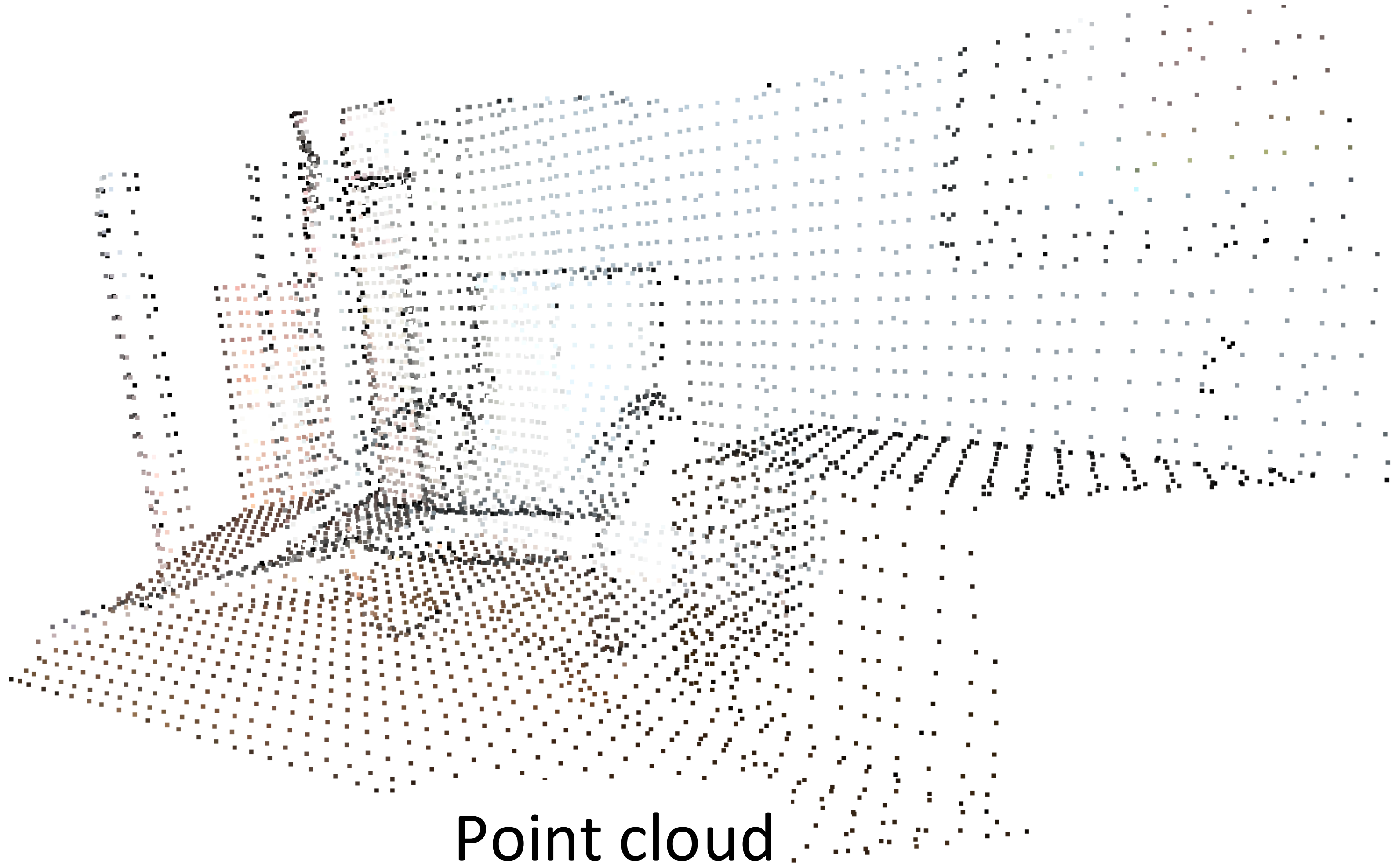
# The Land of 3D Representations



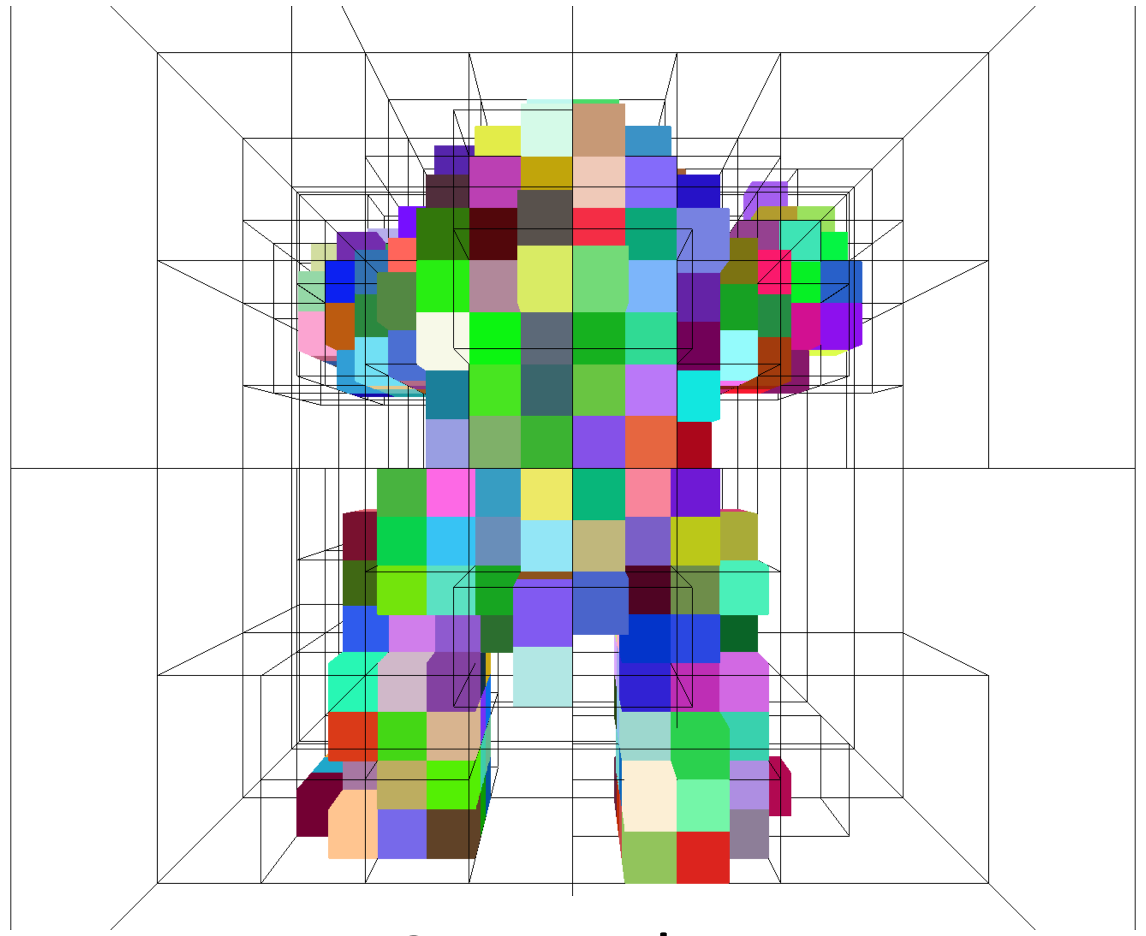

Picture credit: Shubham Tulsiani

# The Land of 3D Representations

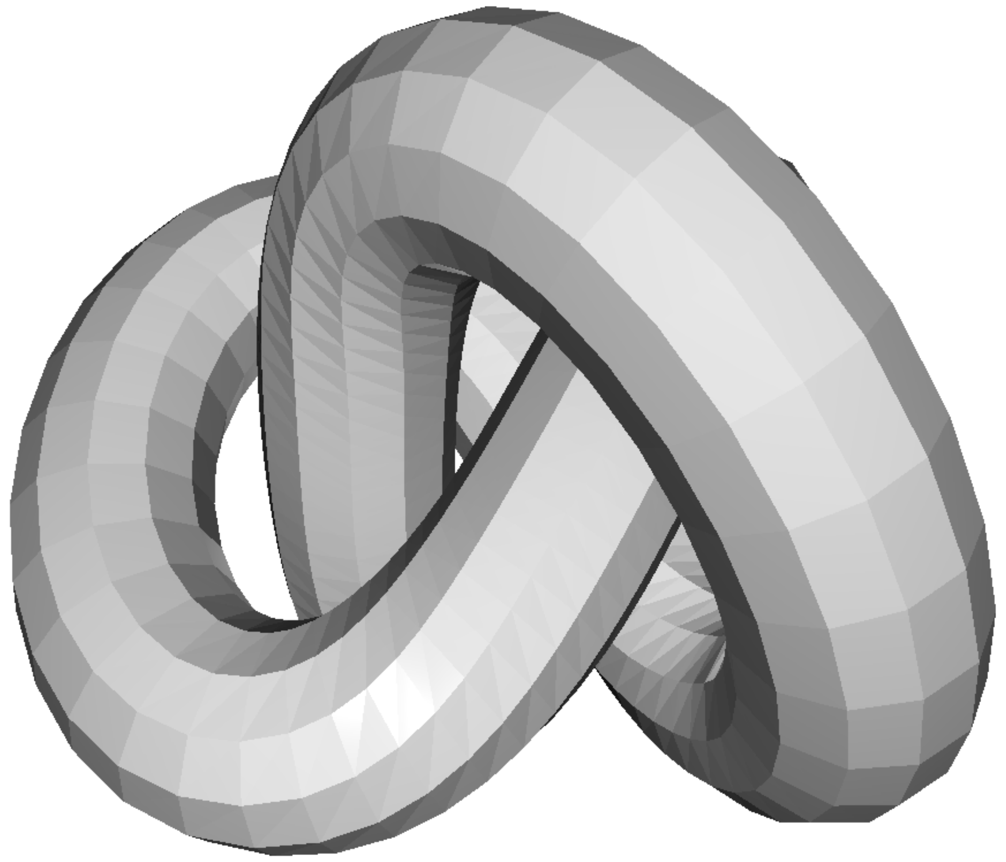
- **Surface** representations
- **Volumetric** representations



Point cloud



3D voxels



Meshes



An **image** that represents how far each pixel **p** is $D[\mathbf{p}] \in \mathbb{R}^+$

Depth images

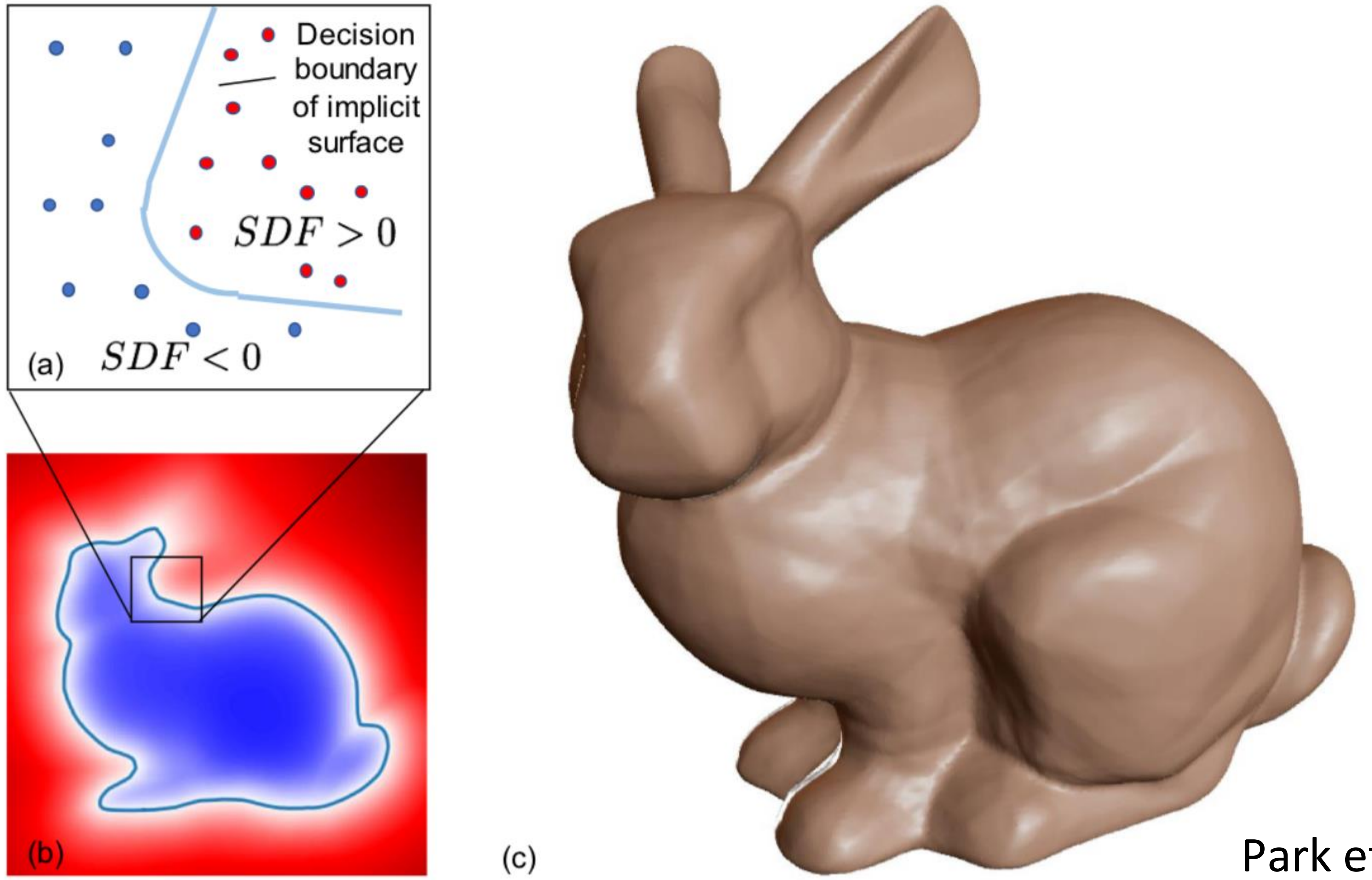
Distance functions

Today's lecture

# Signed Distance Function (SDF)

**Definition 1.** The *signed distance function* (SDF) of a set $\mathcal{O} \subset \mathbb{R}^n$ is a function $f : \mathbb{R}^n \to \mathbb{R}$ that measures the signed distance from a point $p \in \mathbb{R}^n$ to the set boundary $\partial\mathcal{O}$, defined as:

$$f_{\mathrm{SDF}}(p; \mathcal{O}) \triangleq \begin{cases} \min_{y \in \partial\mathcal{O}} \|p - y\|_2 , & p \notin \mathcal{O}, \\ -\min_{y \in \partial\mathcal{O}} \|p - y\|_2 , & p \in \mathcal{O}. \end{cases} \tag{1}$$




Park et al. 2019

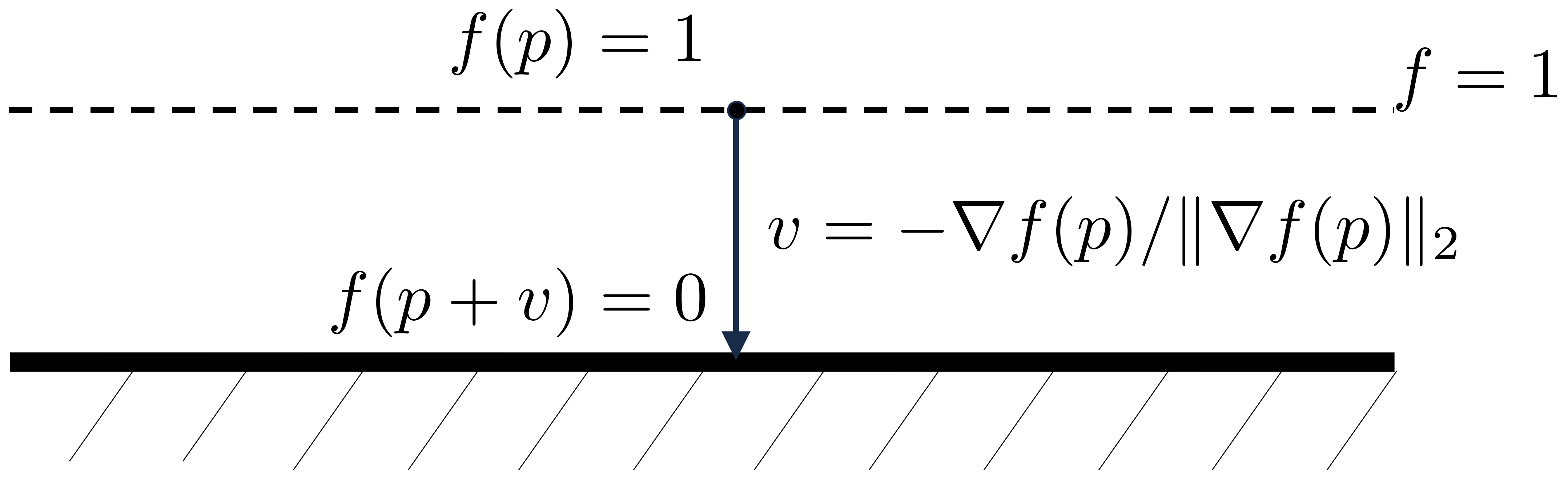
# The Eikonal Property of SDF

- Suppose the SDF is differentiable at a point $p$. Then its gradient satisfies

$$\|\nabla_p f_{\mathrm{sdf}}(p;\mathcal{O})\|_2 = 1.$$

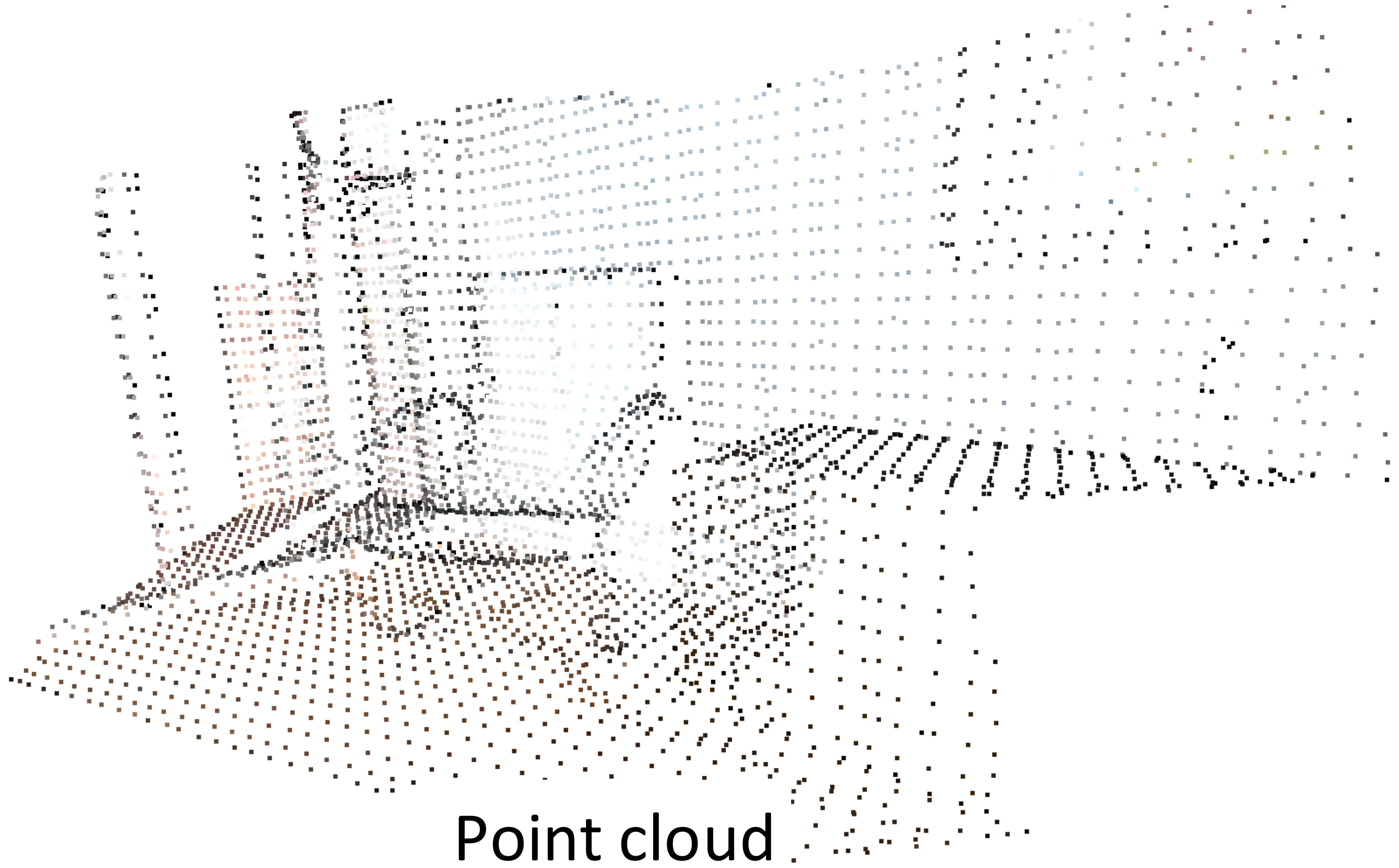
- "*Distance changes at one meter per meter*."

$$\begin{aligned} \cancelto{0}{f(p+v)} &\approx f(p) + v^\top \nabla f(p) \\ &= \cancelto{1}{f(p)} - \|\nabla f(p)\|_2 . \end{aligned}$$

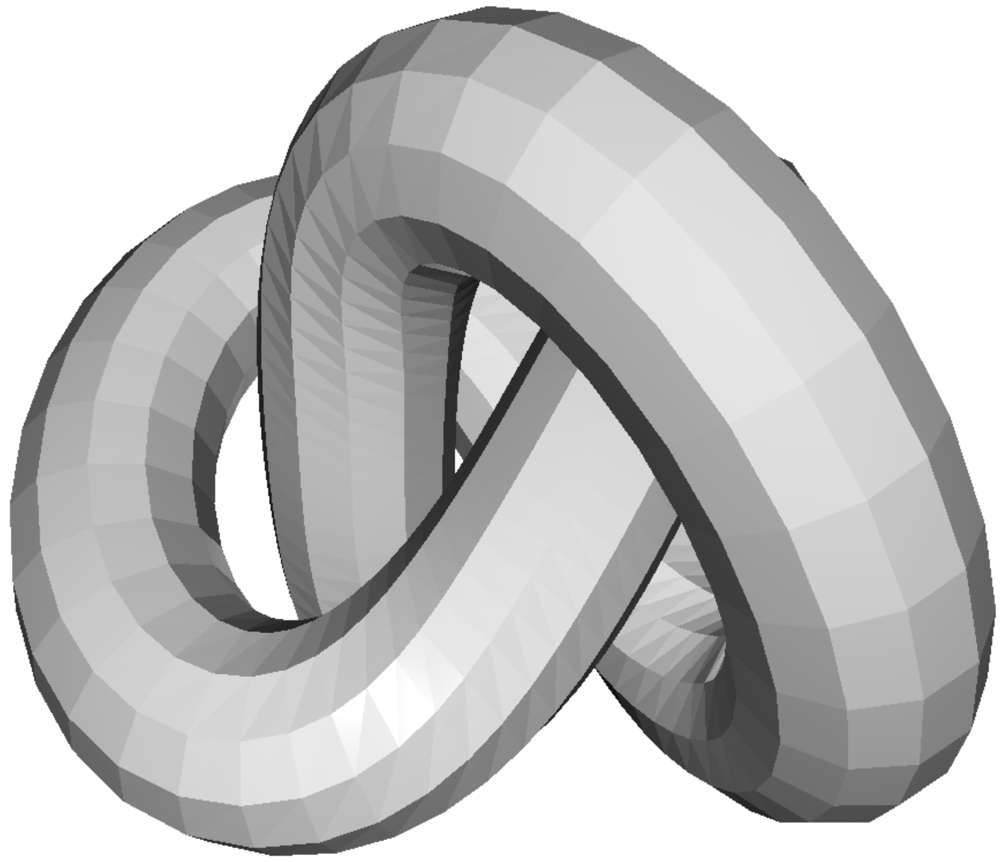


# From SDF to Voxels

- **Surface** representations



Point cloud



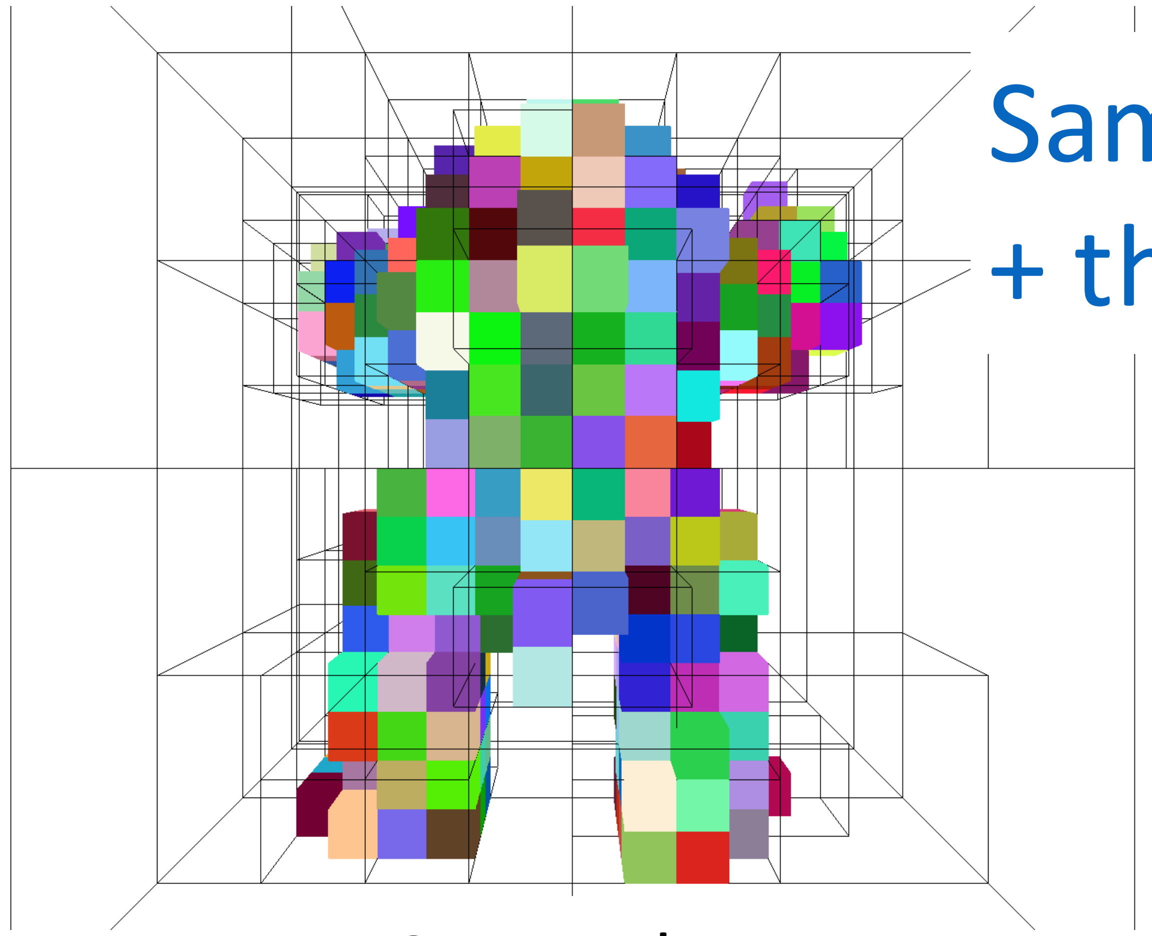
Meshes

credits: Paul Bourke

An **image** that represents how far each pixel **p** is $D[\mathbf{p}] \in \mathbb{R}^+$

Depth images

- **Volumetric** representations



3D voxels

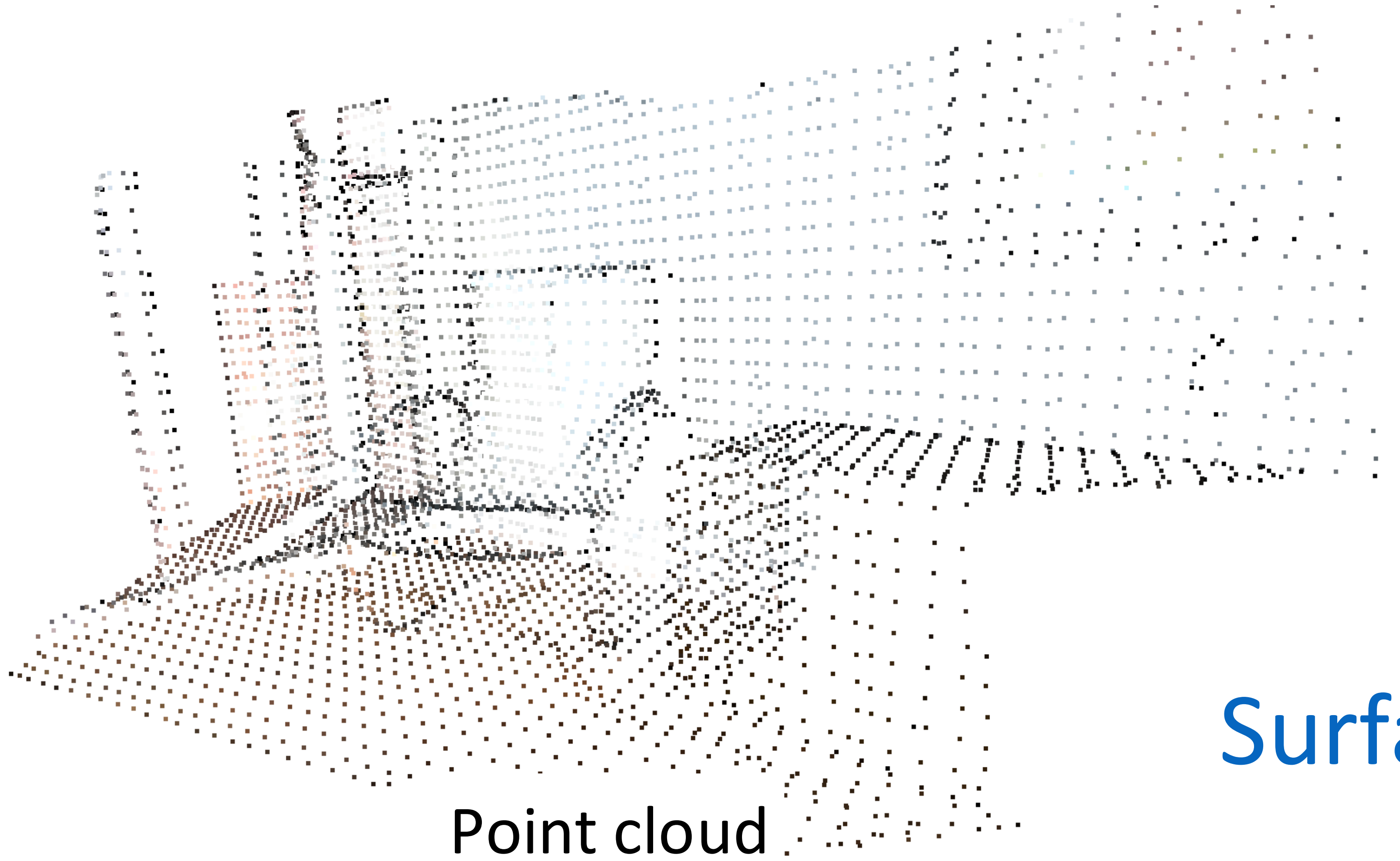
Sampling
+ thresholding

Distance functions

Today's lecture

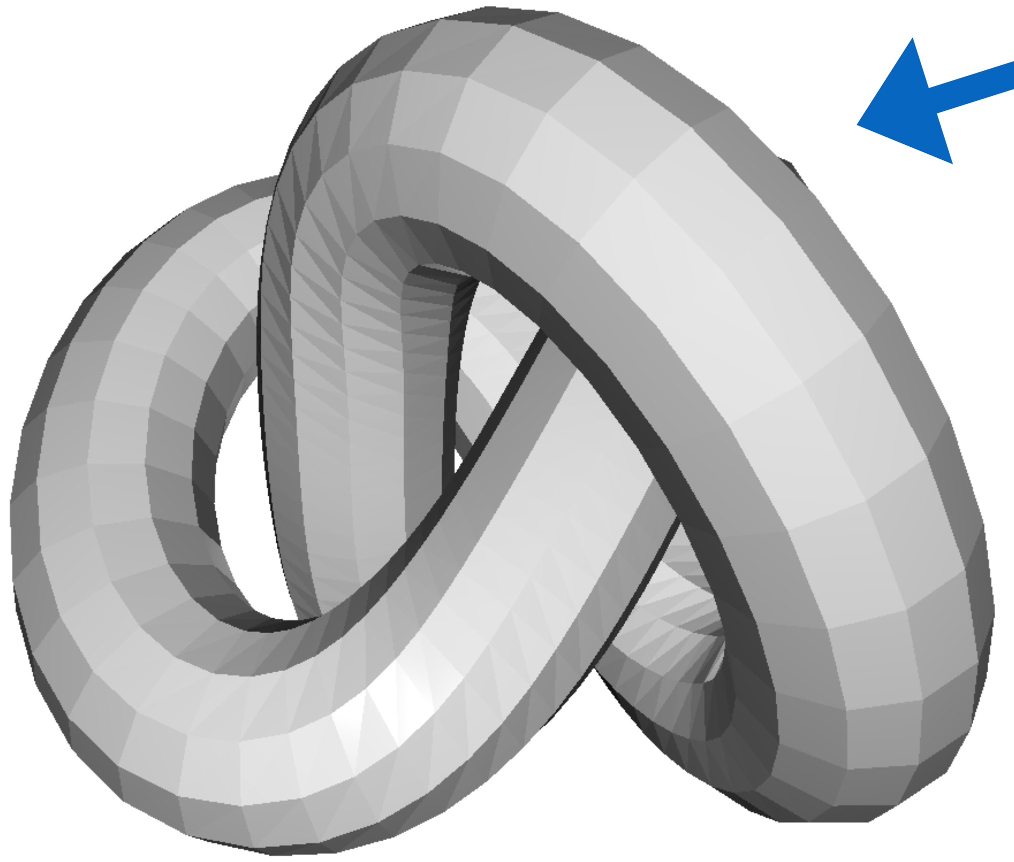
# From SDF to Meshes

- **Surface** representations



Point cloud

Surface reconstruction



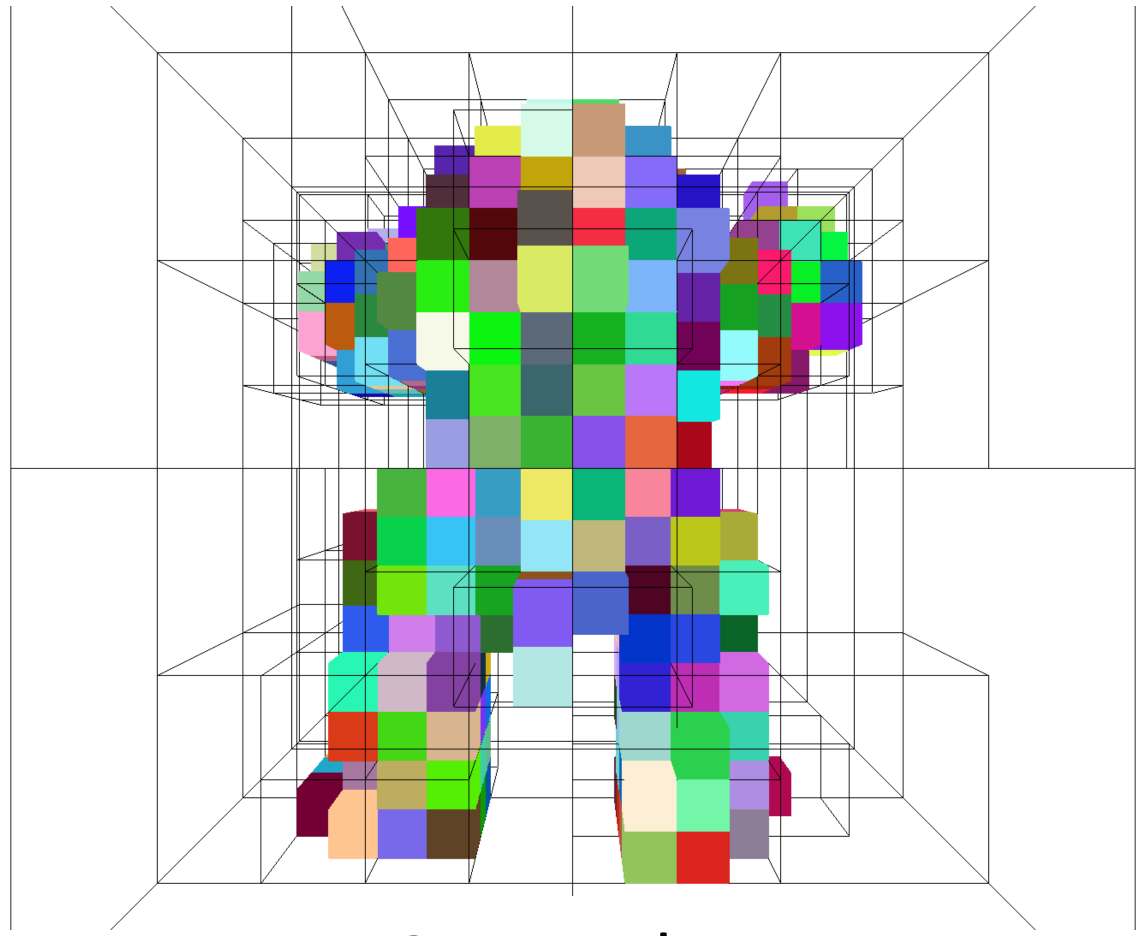
credits: Paul Bourke

Meshes

An **image** that represents how far each pixel **p** is $D[\mathbf{p}] \in \mathbb{R}^+$

Depth images

- **Volumetric** representations

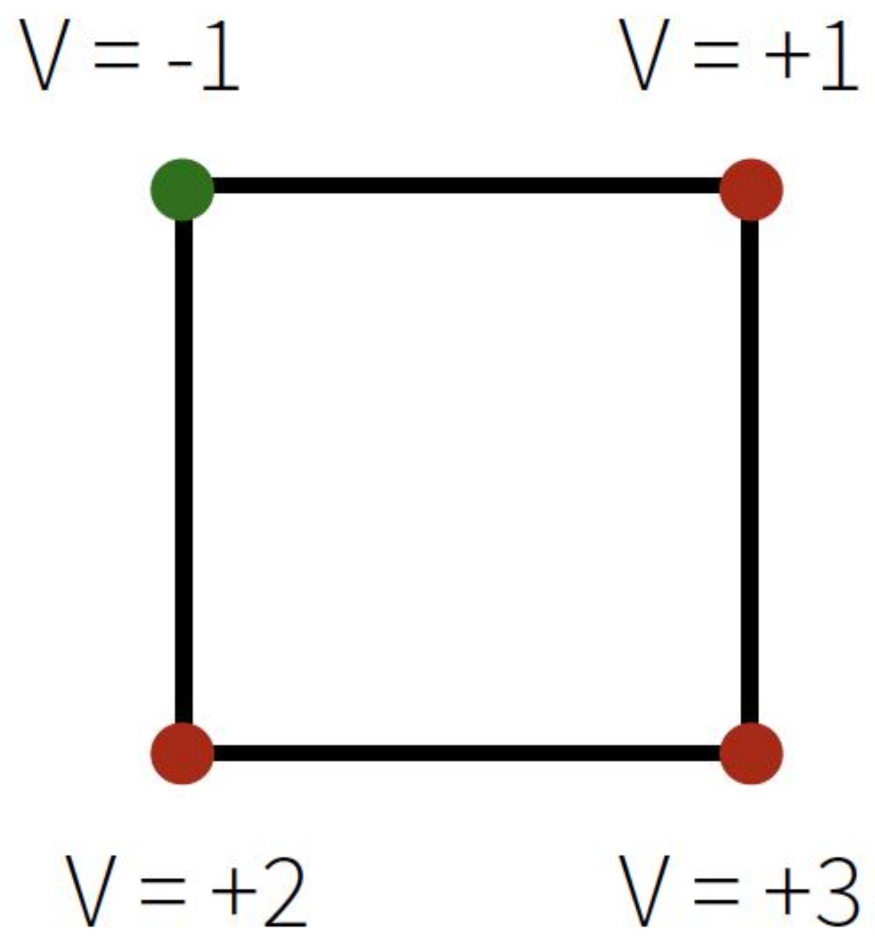


3D voxels

Distance functions

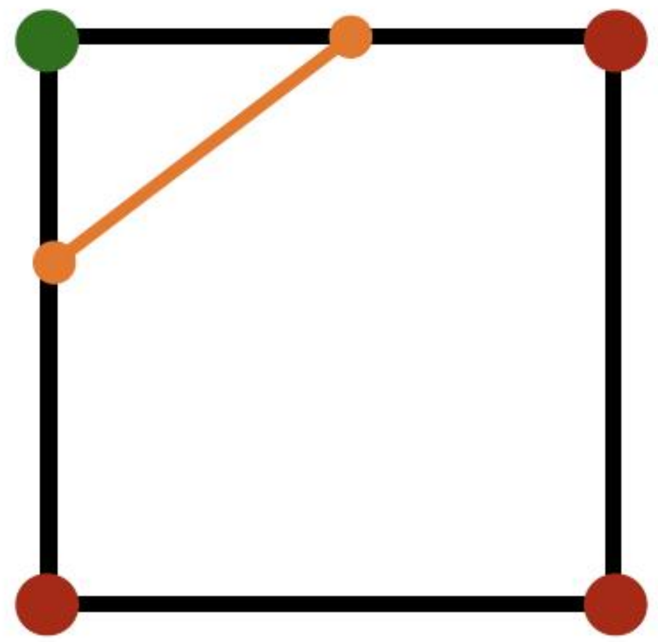
Today's lecture

# Surface Reconstruction in 2D: Single Grid Cell




Green — inside, red — outside

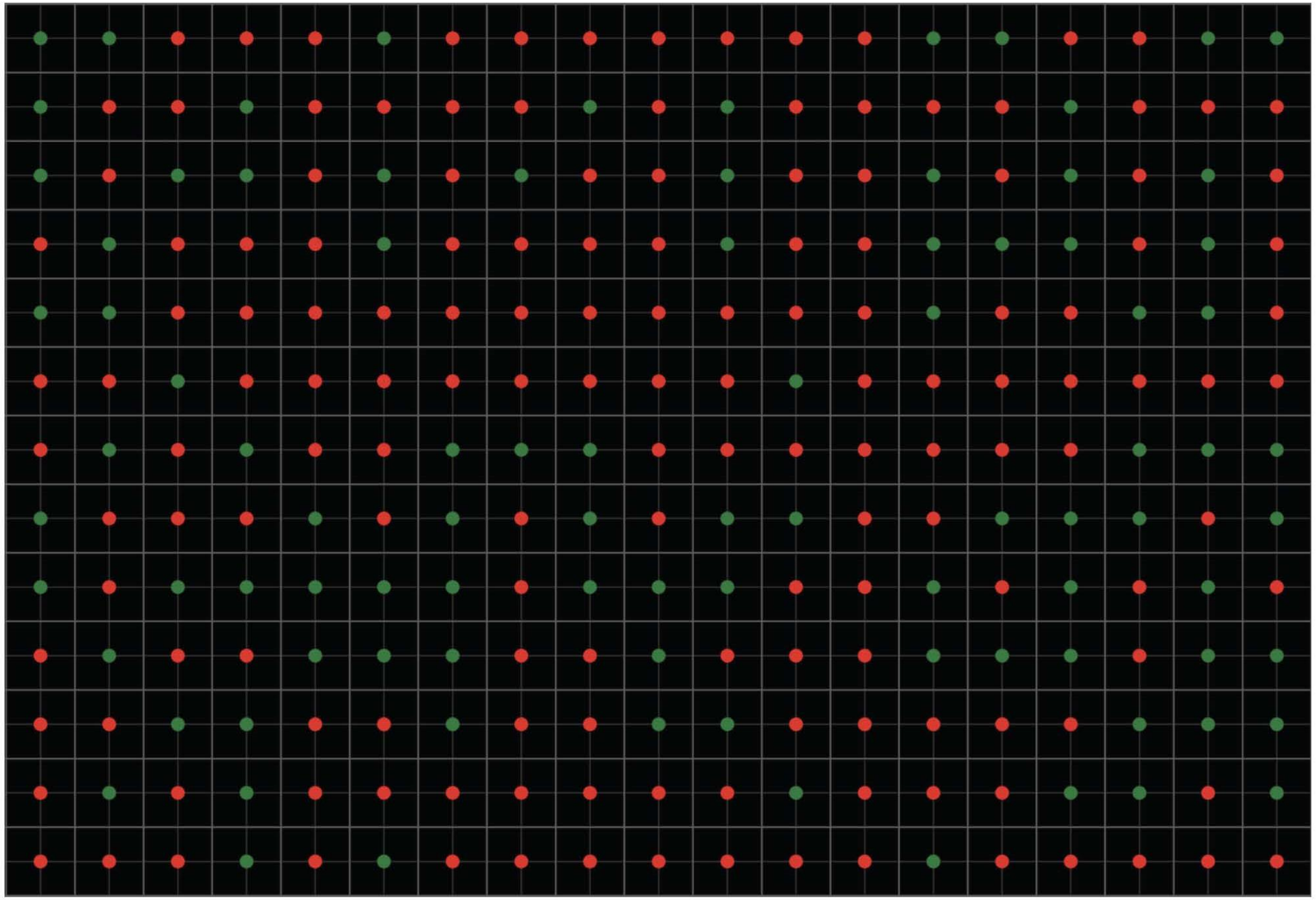
Where should the surface boundary be?



Location of boundary determined by vertex values (differentiable!)

Image credit: Shubham Tulsiani

# Surface Reconstruction in 2D: Marching Square



For each cell:
- Use lookup table to draw contours

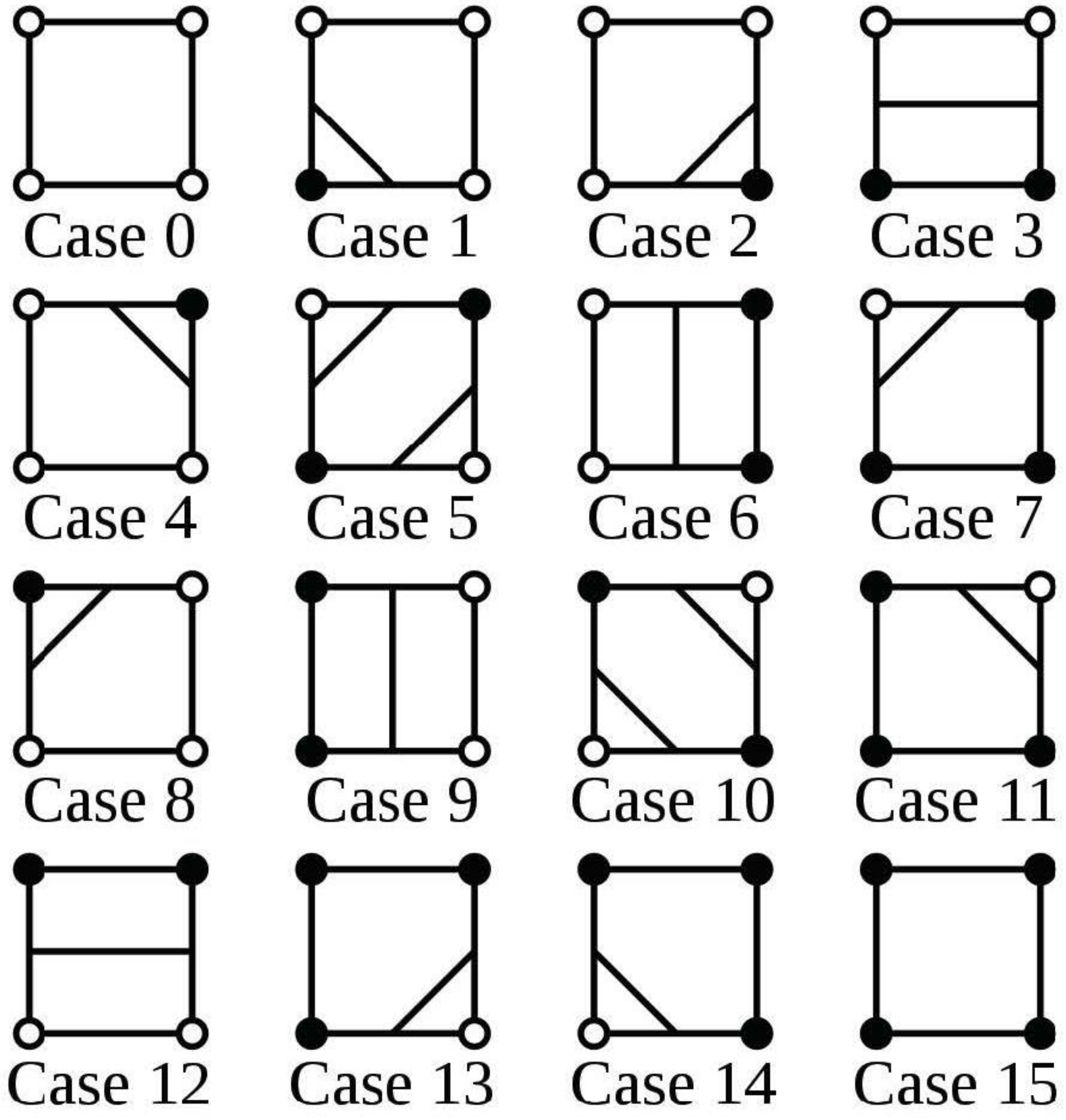
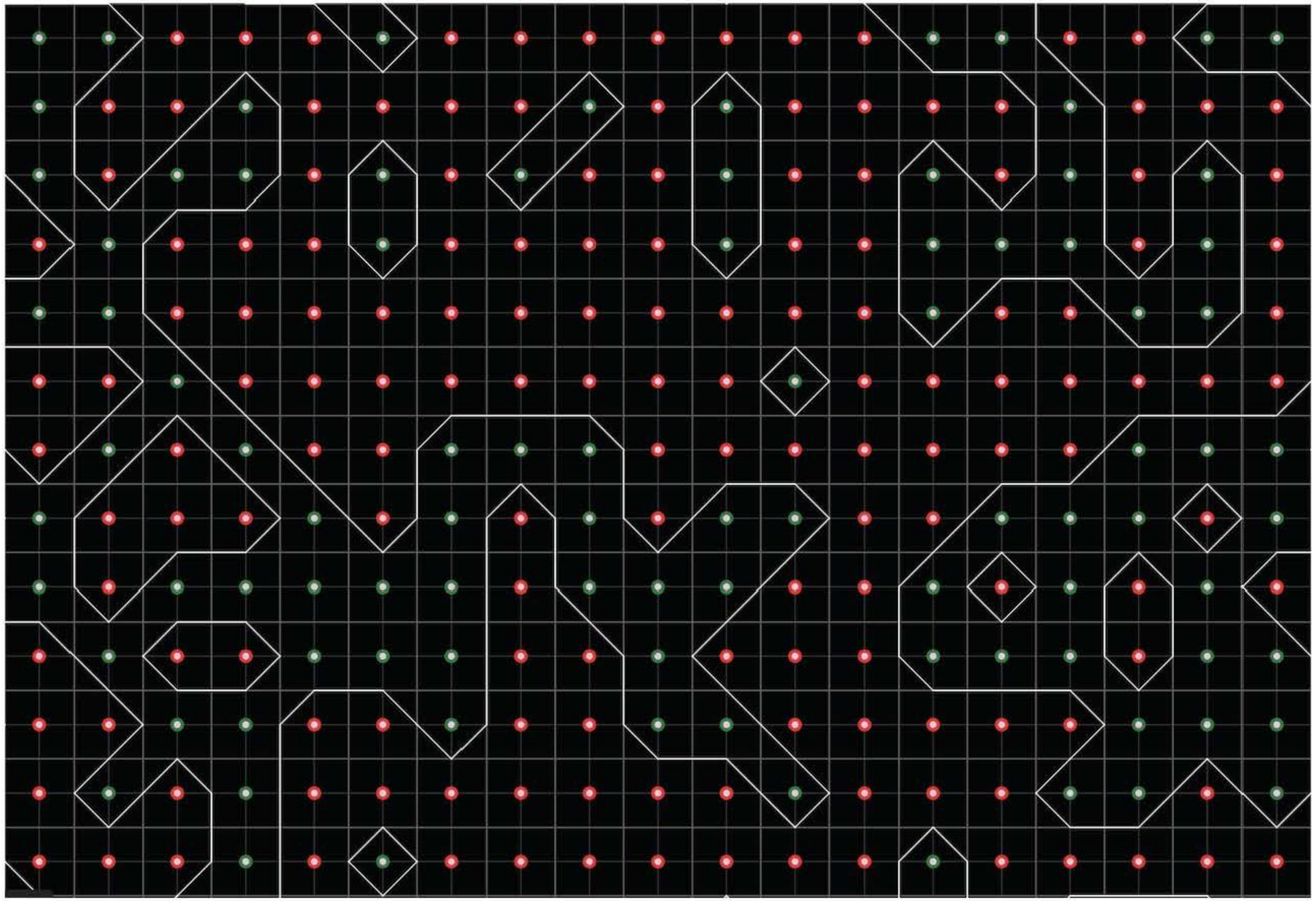



Image credits: David Ramalho

# Surface Reconstruction in 2D: Marching Square



For each cell:

- Use lookup table to draw contours

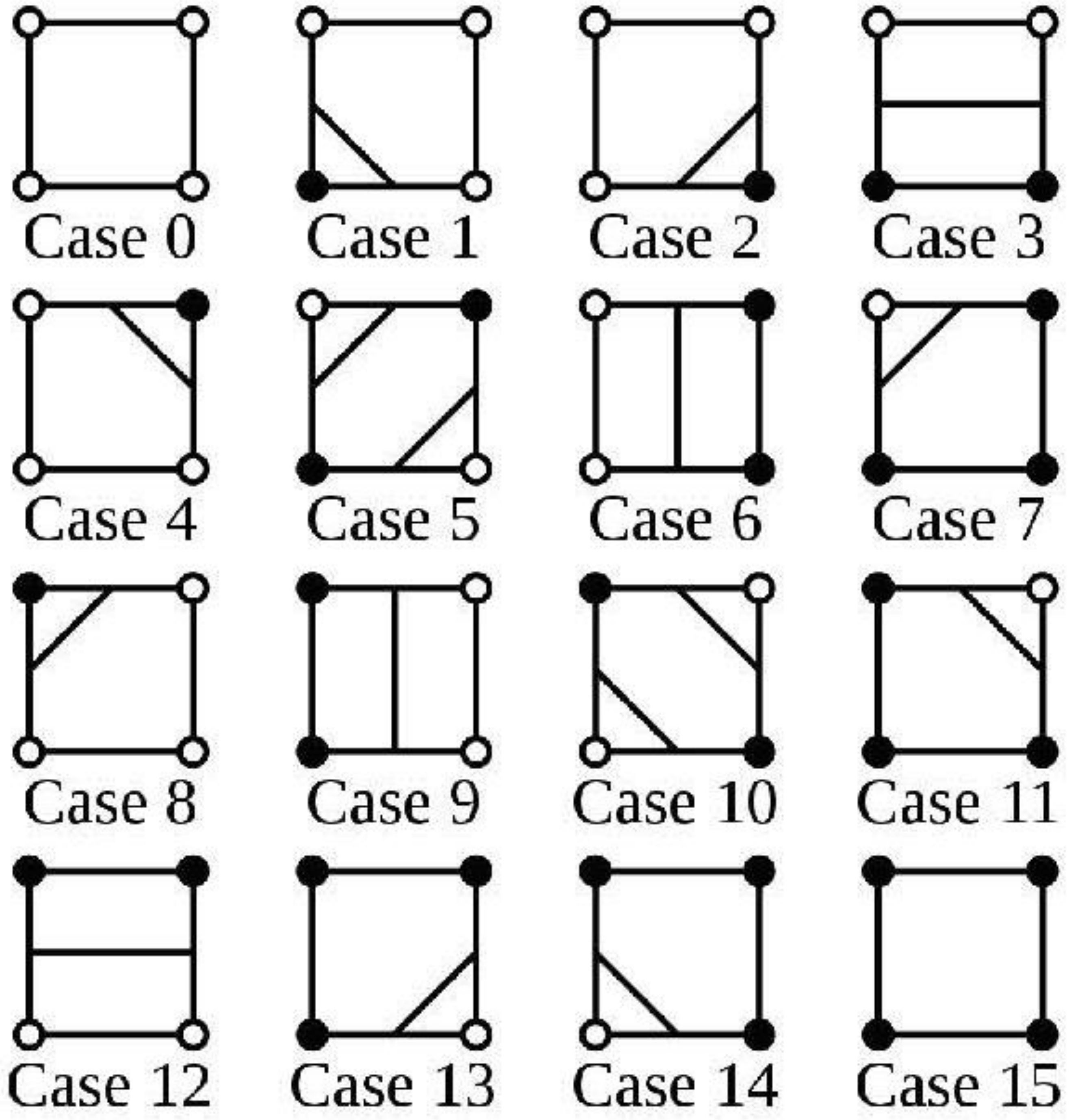
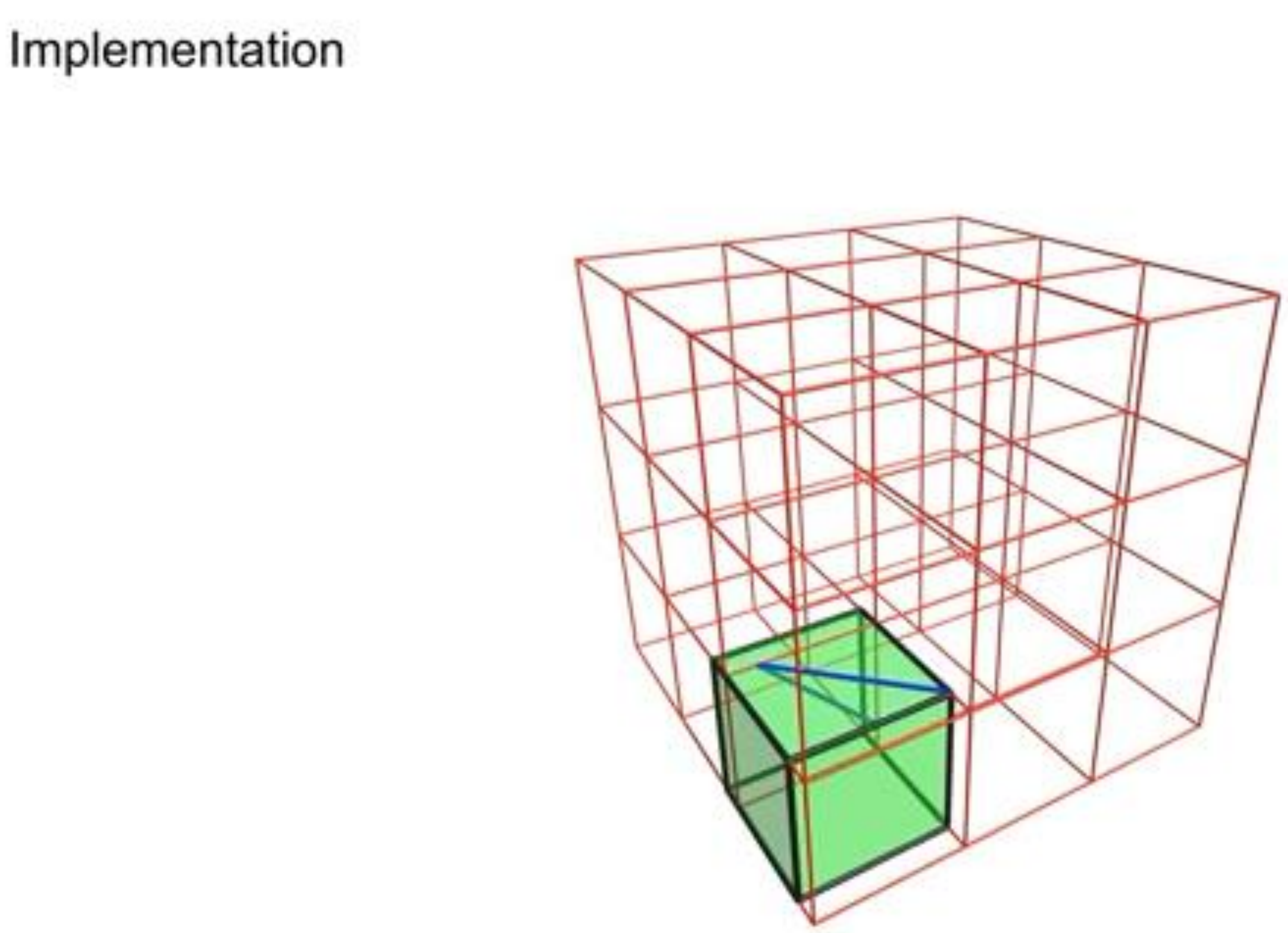



Image credits: David Ramalho

# Surface Reconstruction in 3D: Marching Cube




credit: [www.youtube.com/@algorithmsvisualized9025](www.youtube.com/@algorithmsvisualized9025)

## PyMCubes

`PyMCubes` is an implementation of the marching cubes algorithm to extract iso-surfaces from volumetric data. The volumetric data can be given as a three-dimensional `NumPy` array or as a Python function `f(x, y, z)`.

`PyMCubes` also provides functions to export the results of the marching cubes in a number of mesh file formats.

## Installation

Use `pip`:

```
$ pip install --upgrade PyMCubes
```

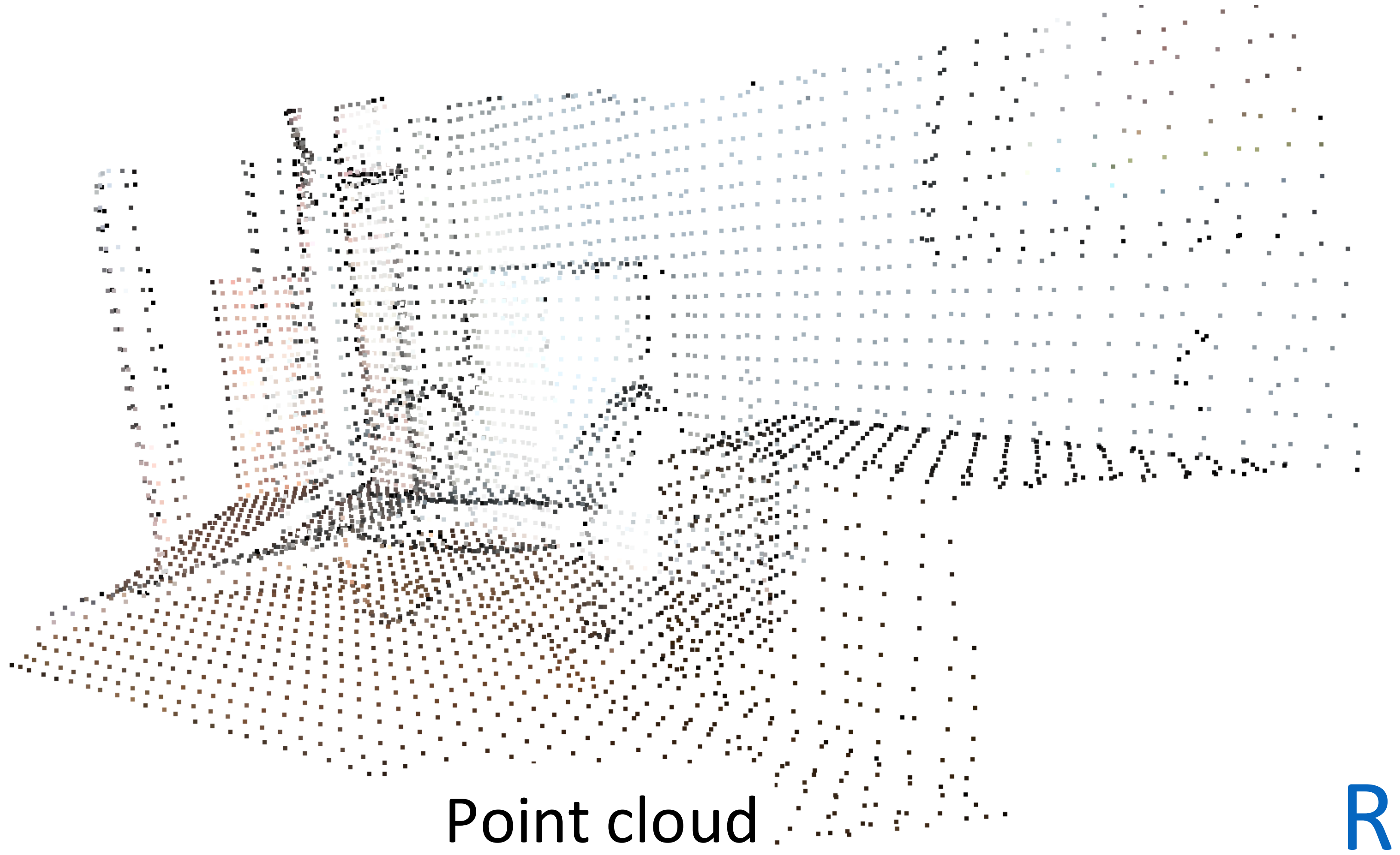
## Example

The following example creates a `NumPy` volume with spherical iso-surfaces and extracts one of them (i.e., a sphere) with `mcubes.marching_cubes`. The result is exported to `sphere.dae`:

```
>>> import numpy as np
>>> import mcubes

# Create a data volume (30 x 30 x 30)
>>> X, Y, Z = np.mgrid[:30, :30, :30]
>>> u = (X-15)**2 + (Y-15)**2 + (Z-15)**2 - 8**2

# Extract the 0-isosurface
>>> vertices, triangles = mcubes.marching_cubes(u, 0)

# Export the result to sphere.dae
>>> mcubes.export_mesh(vertices, triangles, "sphere.dae", "MySphere")
```

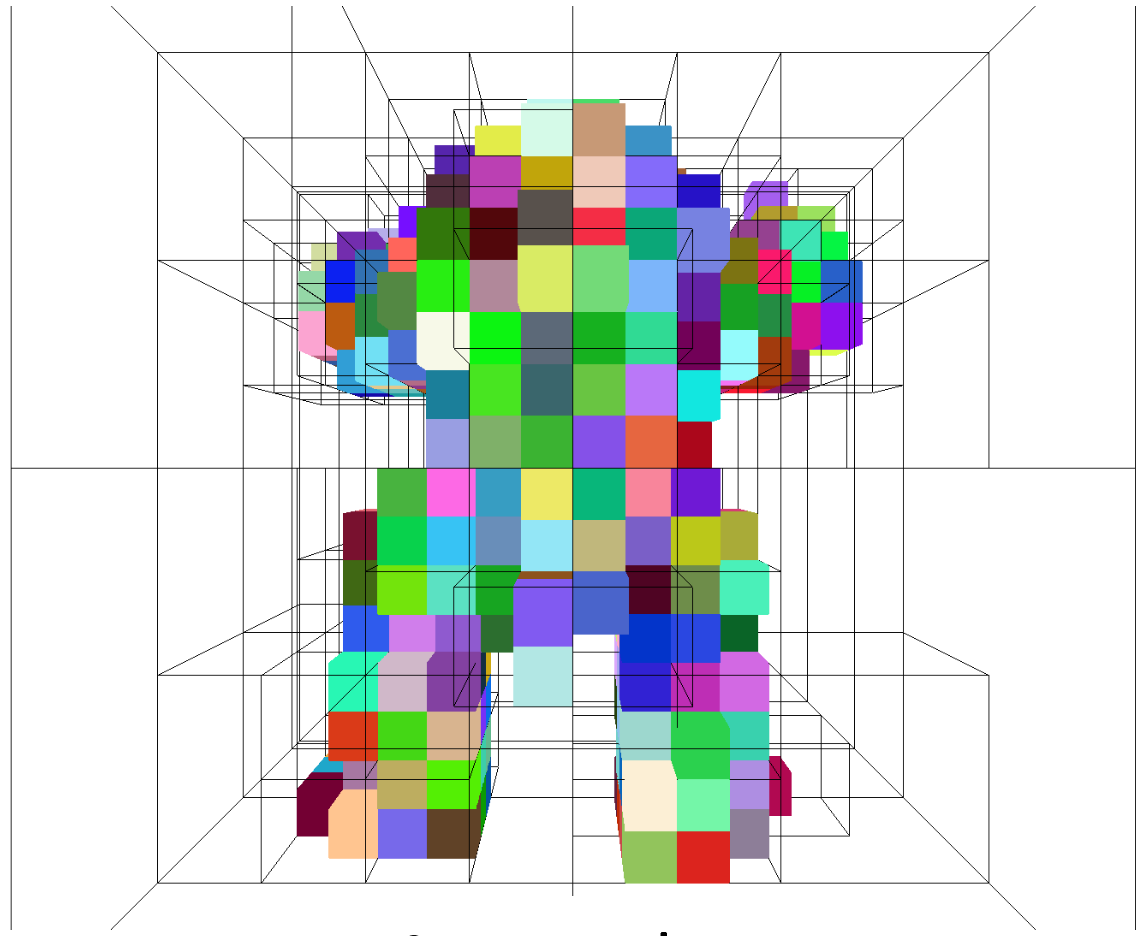
# From SDF to Depth Images

- **Surface** representations
- **Volumetric** representations



Point cloud

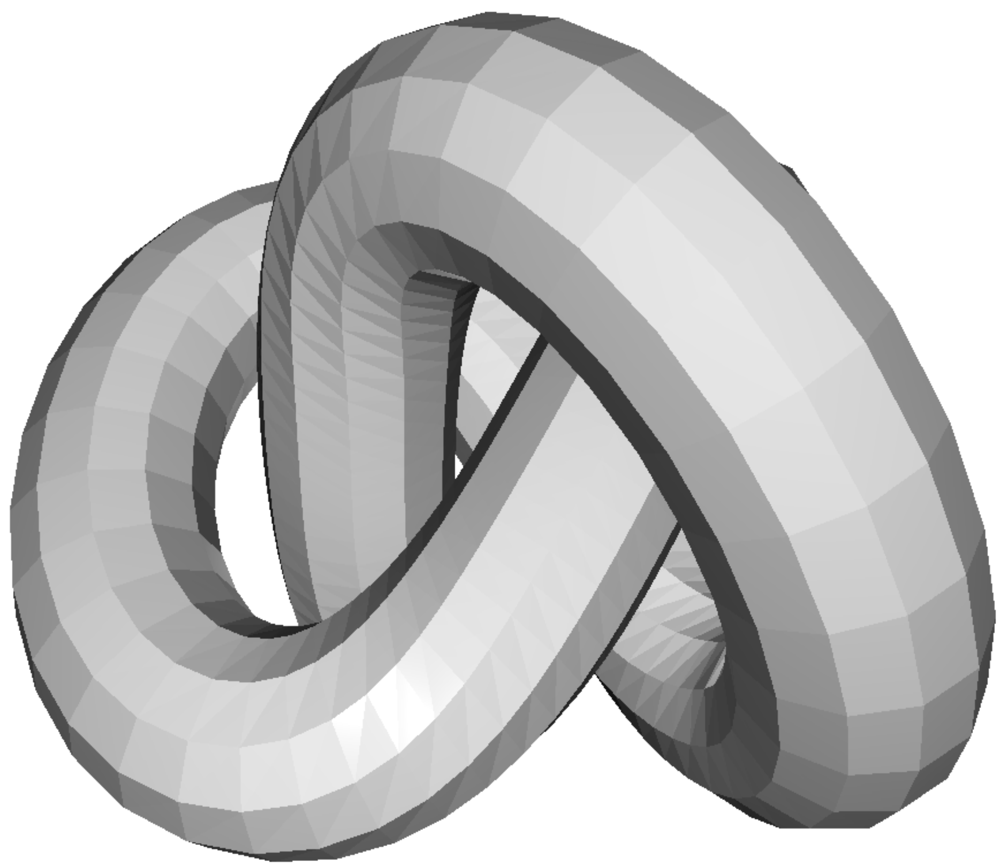
Rendering



3D voxels

Distance functions

Today's lecture



credits: Paul Bourke

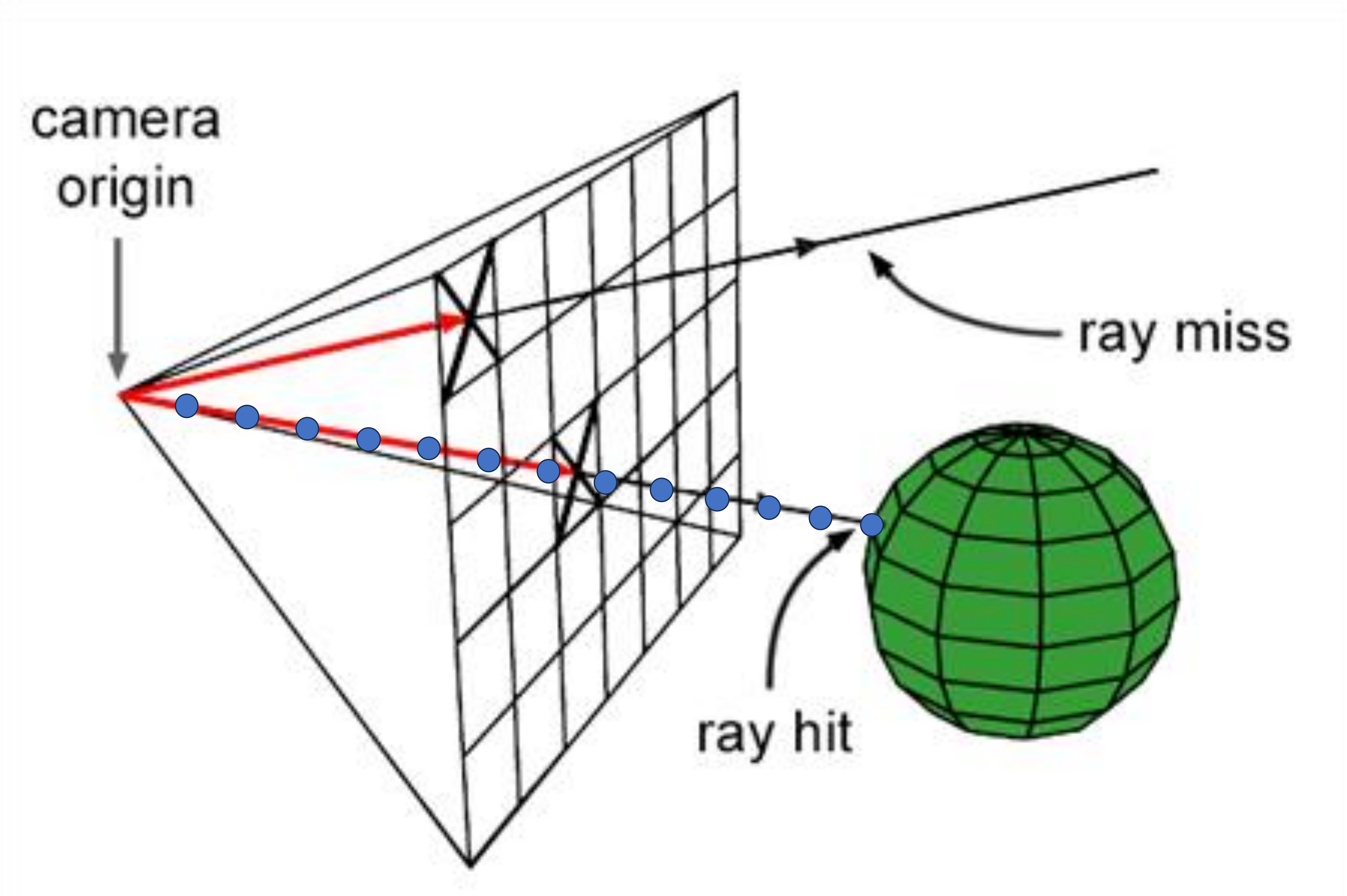
Meshes

An **image** that represents how far each pixel **p** is

$D[\mathbf{p}] \in \mathbb{R}^+$

Depth images

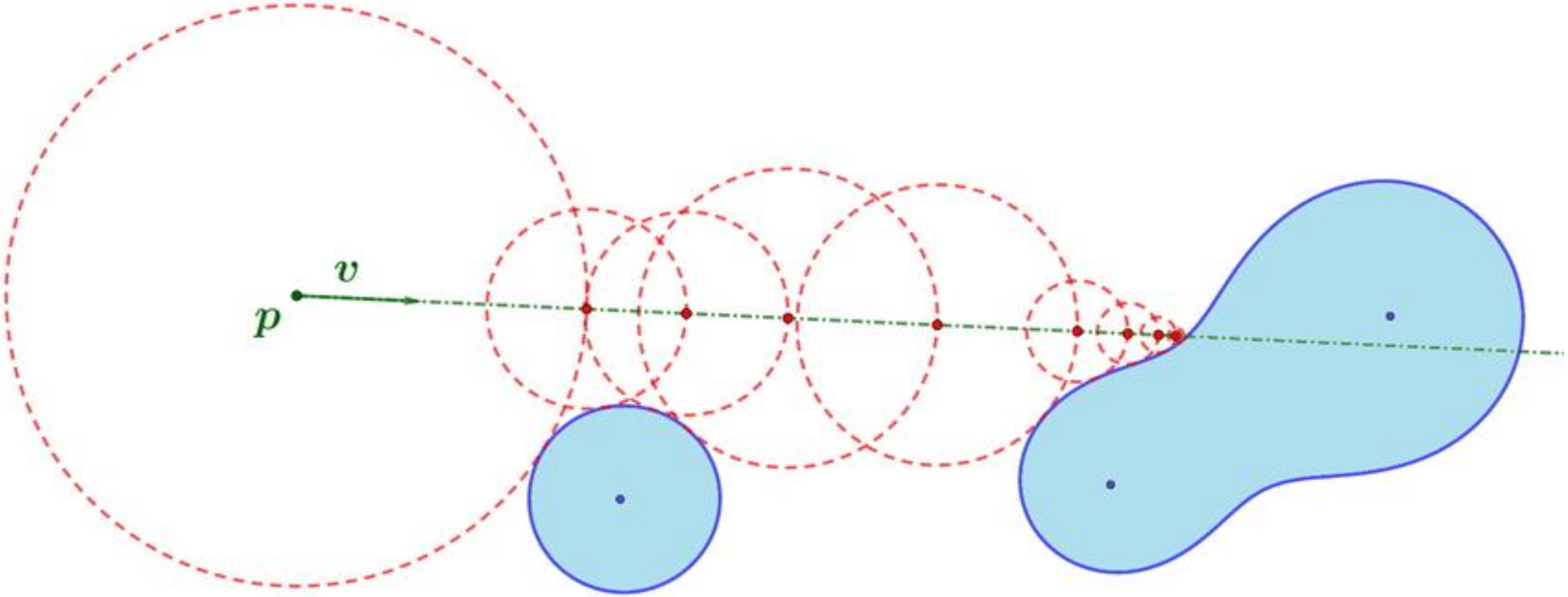
# Rendering: Ray Marching




🤔 Can we use the property of SDF to speed up ray marching?

# Rendering: Sphere Tracing

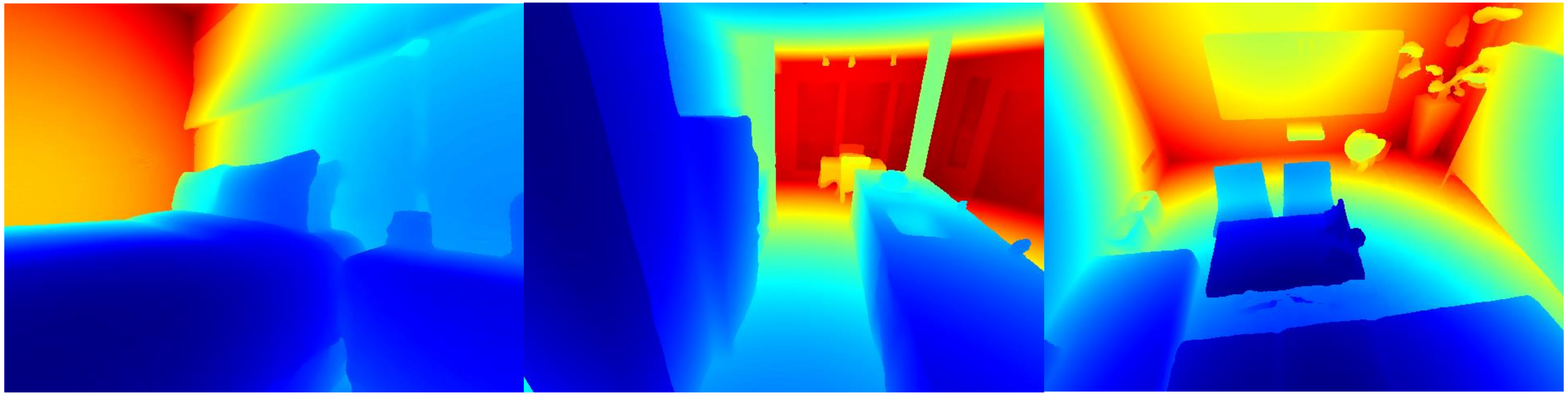
- Key idea: SDF at any point gives minimum step size!
- Further from surface ➔ larger step size ➔ faster rendering



# Rendering: Sphere Tracing

- Key idea: SDF at any point gives minimum step size!
- Further from surface ➔ larger step size ➔ faster rendering

Input viewpoints (shown as RGB images)

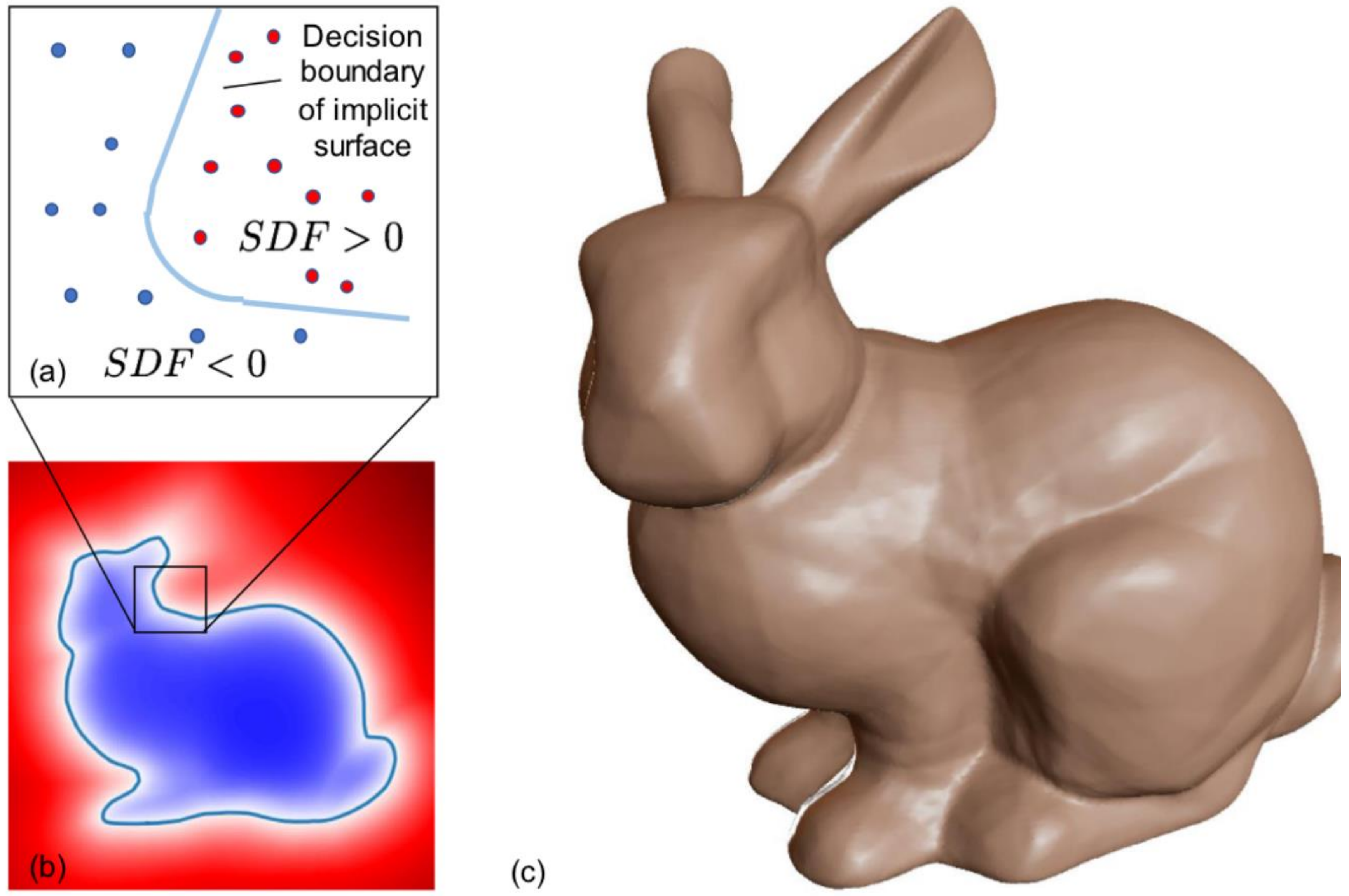


Depth from sphere tracing

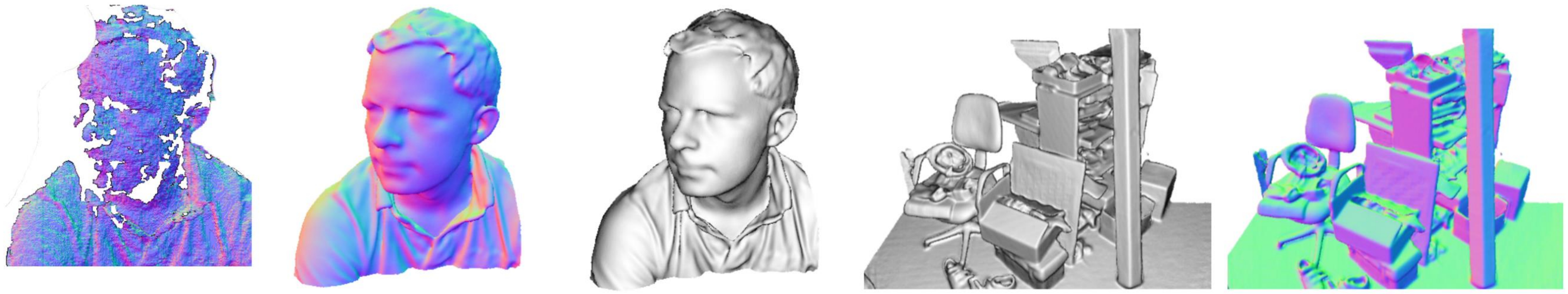
Image credit: Zhirui Dai

# Today’s Lecture

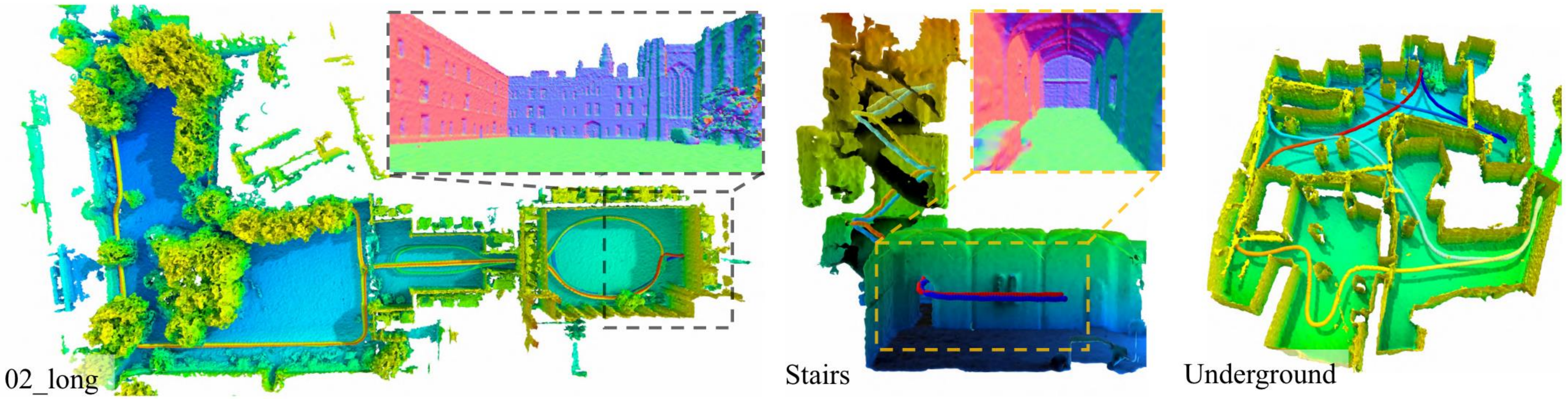
- Dense signed distance function (SDF) representation and properties
- Basics of 3D dense SLAM using SDF
- Recent advancements to improve SDF-based SLAM




Park et al. 2019



Newcombe et al. 2011




Pan et al. 2024

# SLAM with Dense Representation

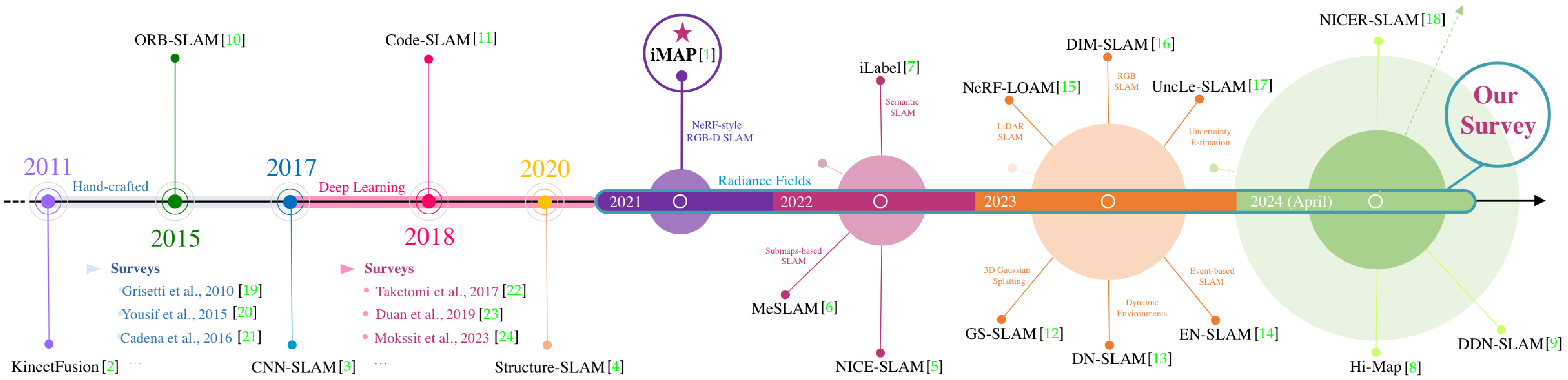
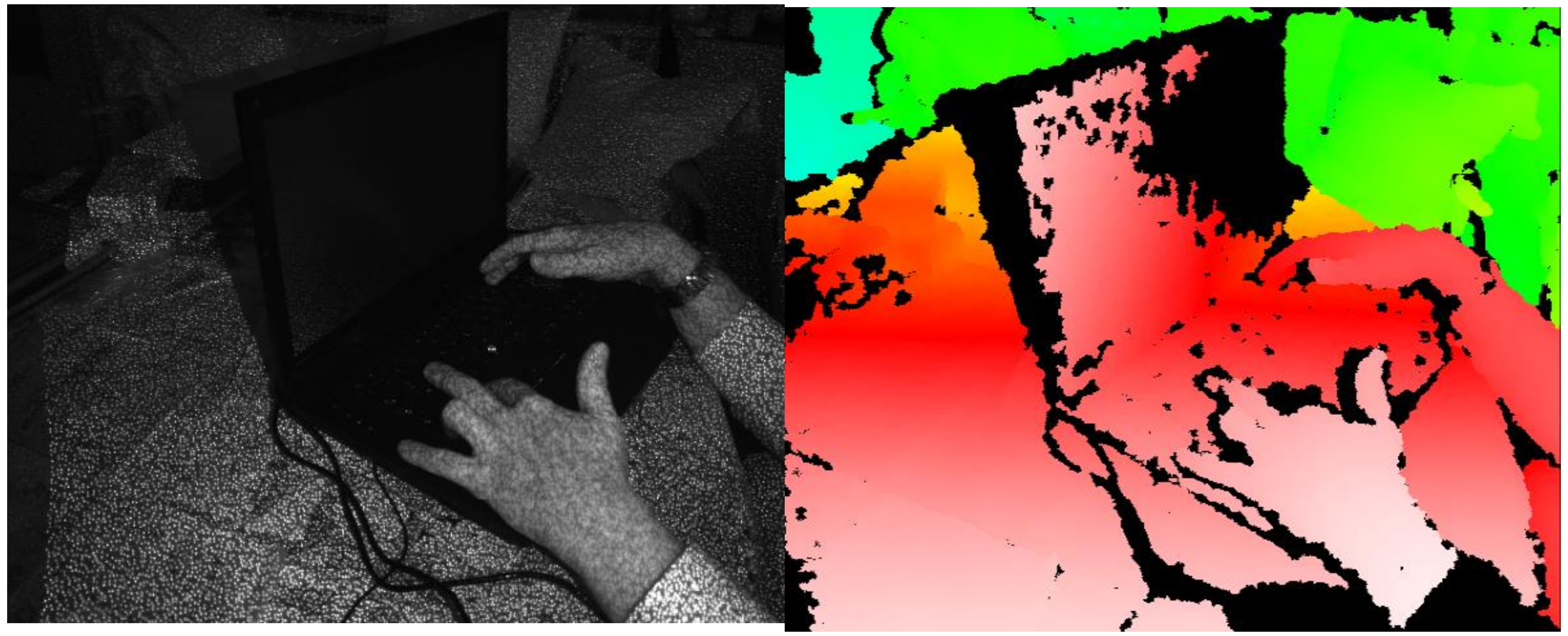



Fig. 1: **Timeline SLAM Evolution.** This timeline begins by illustrating the transition from hand-crafted to deep learning techniques, featuring key surveys from both eras. In 2021, a pivotal shift focuses on radiance-field-based SLAM systems, marked by iMap [1]. The circles on the right side of the figure represent key papers for each year, with size indicating publication volume. The outer circle for 2024 signals a projected surge, highlighting the growing interest in NeRF and 3DGS-inspired SLAM.

Image credit: Tosi et al. “How NeRFs and 3D Gaussian Splatting are Reshaping SLAM: a Survey”.

# Where it began: KinectFusion (2011)

**Context**: Kinect RGB-D camera released by Microsoft in 2010.

**KinectFusion: Real-Time Dense Surface Mapping and Tracking***

Richard A. Newcombe, Imperial College London · Shahram Izadi, Microsoft Research · Otmar Hilliges, Microsoft Research · David Molyneaux, Microsoft Research, Lancaster University · David Kim, Microsoft Research, Newcastle University · Andrew J. Davison, Imperial College London · Pushmeet Kohli, Microsoft Research · Jamie Shotton, Microsoft Research · Steve Hodges, Microsoft Research · Andrew Fitzgibbon, Microsoft Research

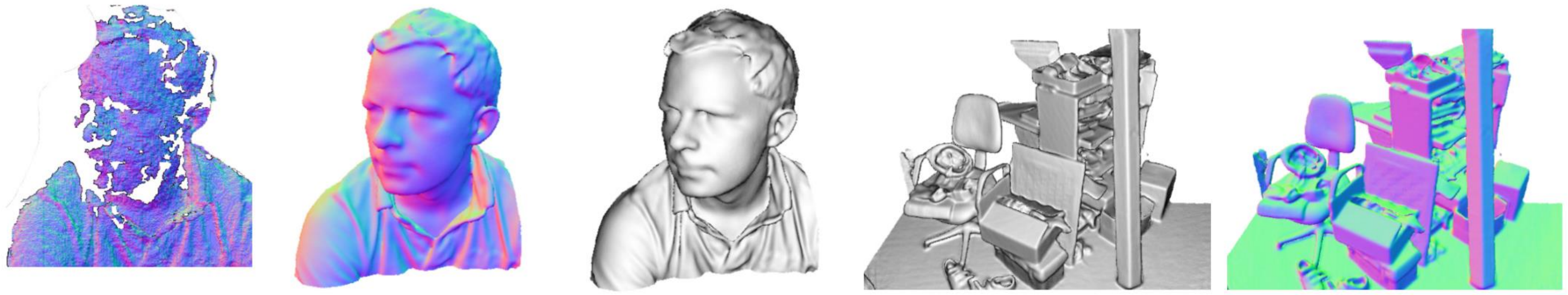


Figure 1: Example output from our system, generated in real-time with a handheld Kinect depth camera and no other sensing infrastructure. Normal maps (colour) and Phong-shaded renderings (greyscale) from our dense reconstruction system are shown. On the left for comparison is an example of the live, incomplete, and noisy data from the Kinect sensor (used as input to our system).

**Kinectfusion**: Real-time dense surface mapping and tracking

RA Newcombe, S Izadi, O Hilliges… - 2011 10th IEEE …, 2011 - ieeexplore.ieee.org

We present a system for accurate real-time mapping of complex and arbitrary indoor scenes in variable lighting conditions, using only a moving low-cost depth camera and commodity …

☆ Save  99 Cite  Cited by 5474  Related articles  All 21 versions

# KinectFusion (2011)

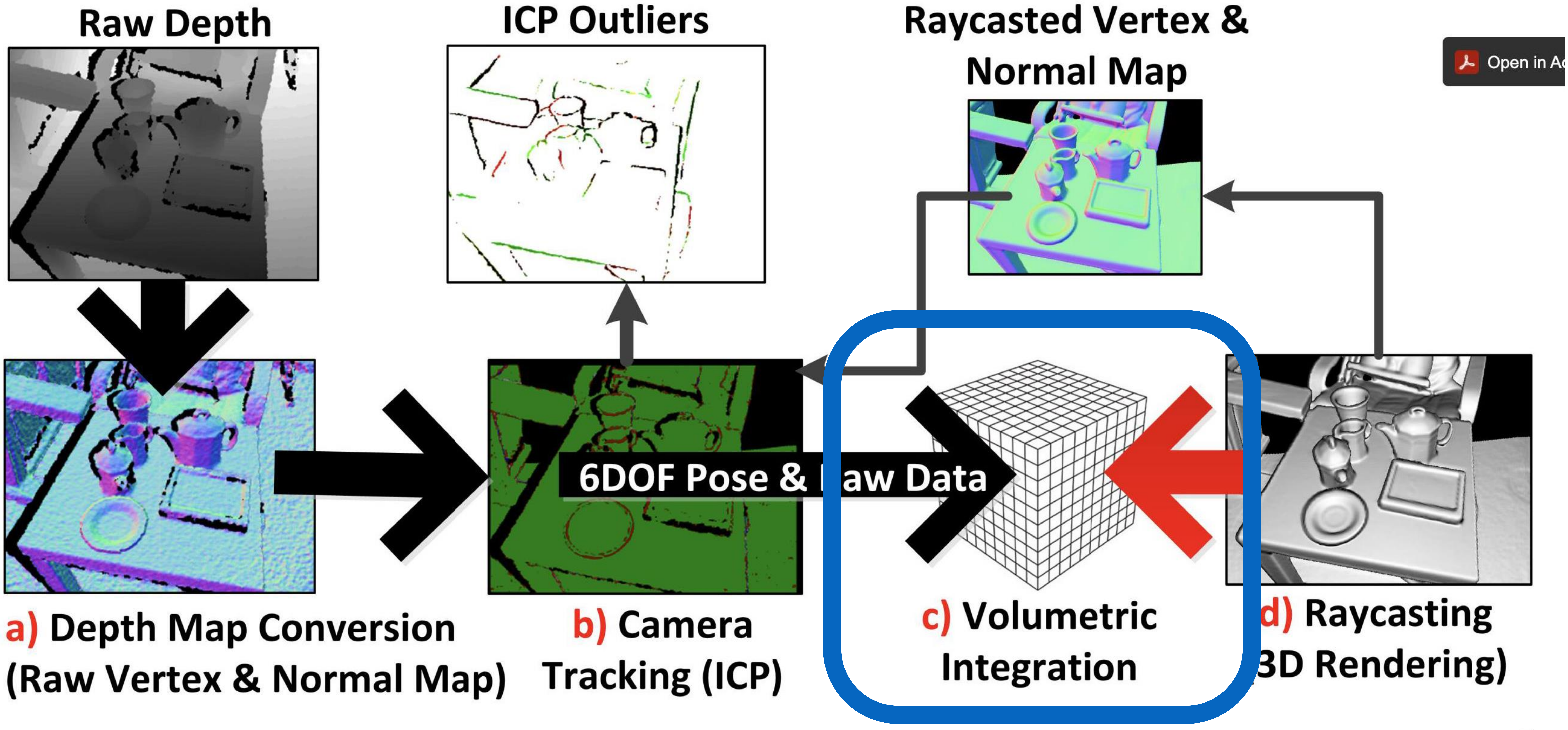



Image credit: Newcombe

# Truncated Signed Distance Function (TSDF)

- KinectFusion uses truncated SDF (TSDF)
- Better noise handling and faster computation

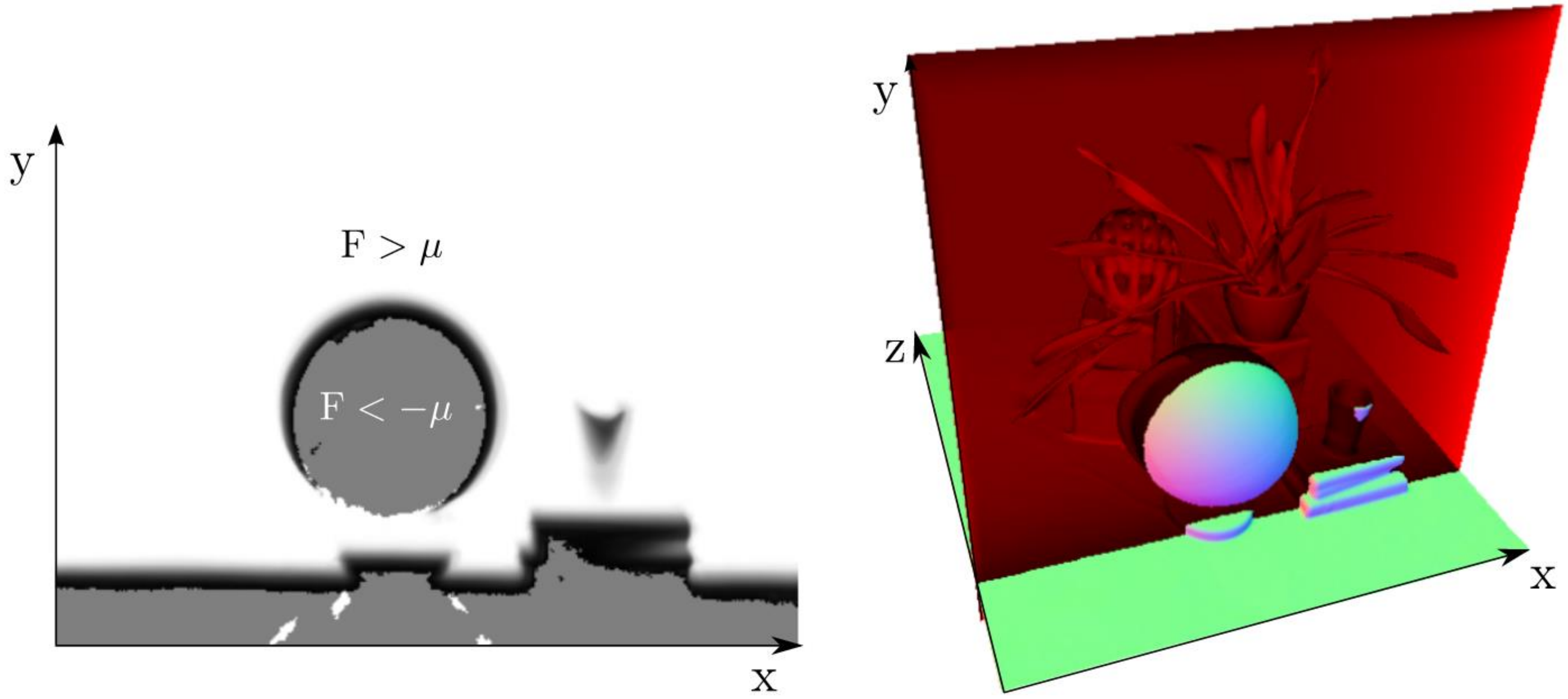
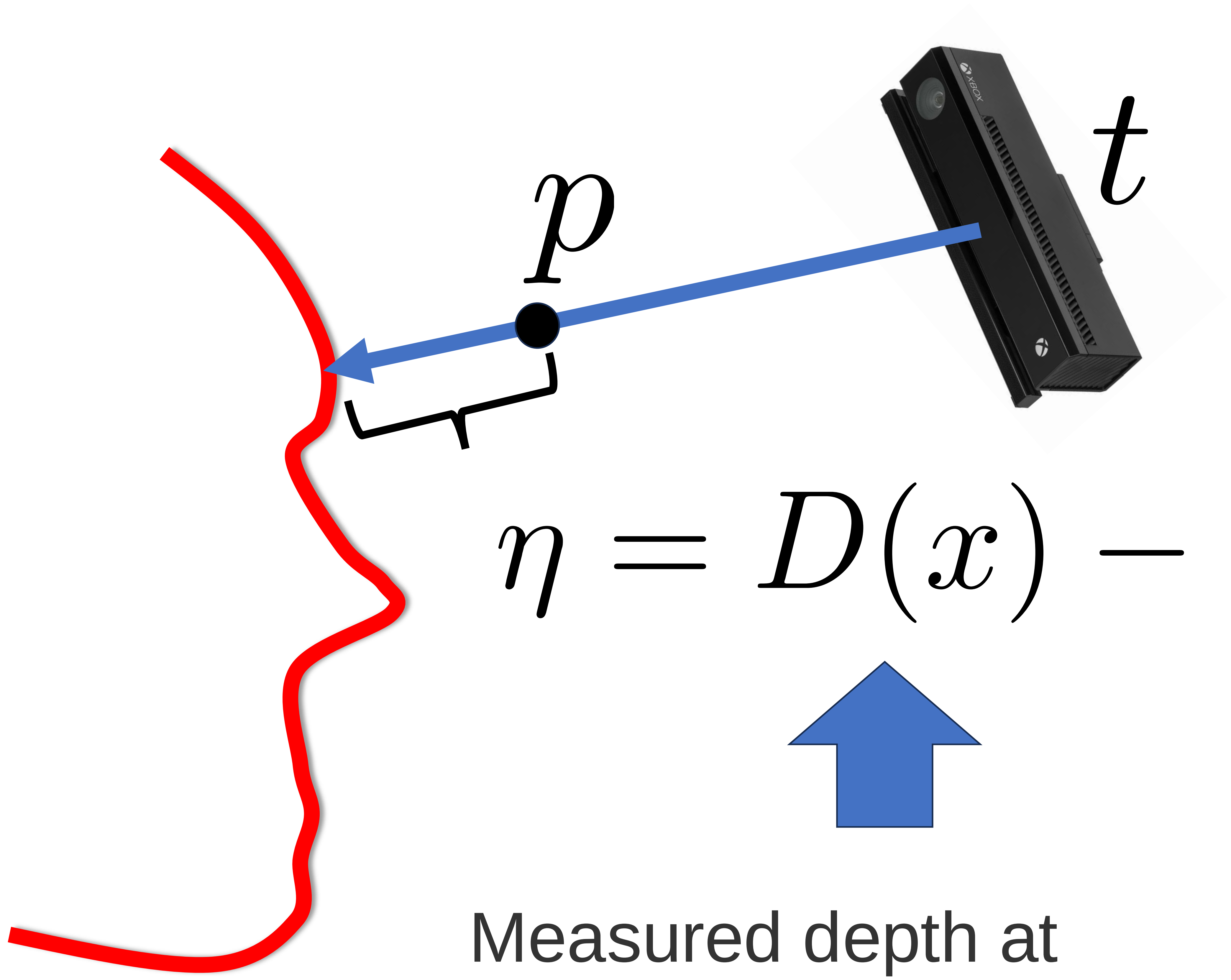



Figure 4: A slice through the truncated signed distance volume showing the truncated function $F > \mu$ (white), the smooth distance field around the surface interface $F = 0$ and voxels that have not yet had a valid measurement(grey) as detailed in eqn. 9.

# Truncated Signed Distance Function (TSDF)

- KinectFusion uses truncated SDF (TSDF) computed by projective distance




$$\eta = D(x) - \lambda^{-1} \|p - t\|_2$$

Measured depth at corresponding pixel

Depth from p to camera center t

# Truncated Signed Distance Function (TSDF)

- KinectFusion uses truncated SDF (TSDF) computed by projective distance

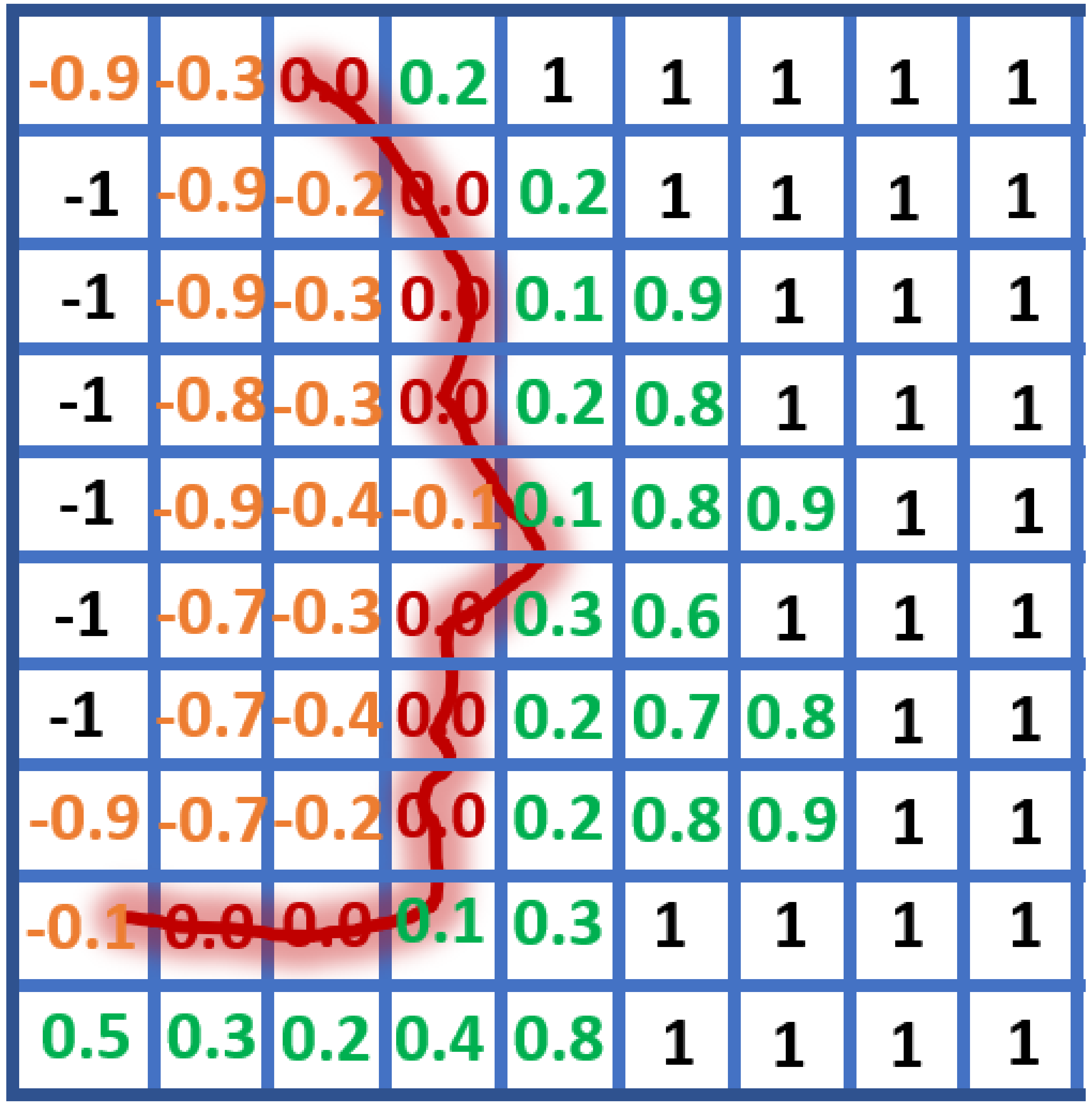



Image credit: Tim Cheng

From every raw depth image, we obtain a *normalized* and *weighted* TSDF:

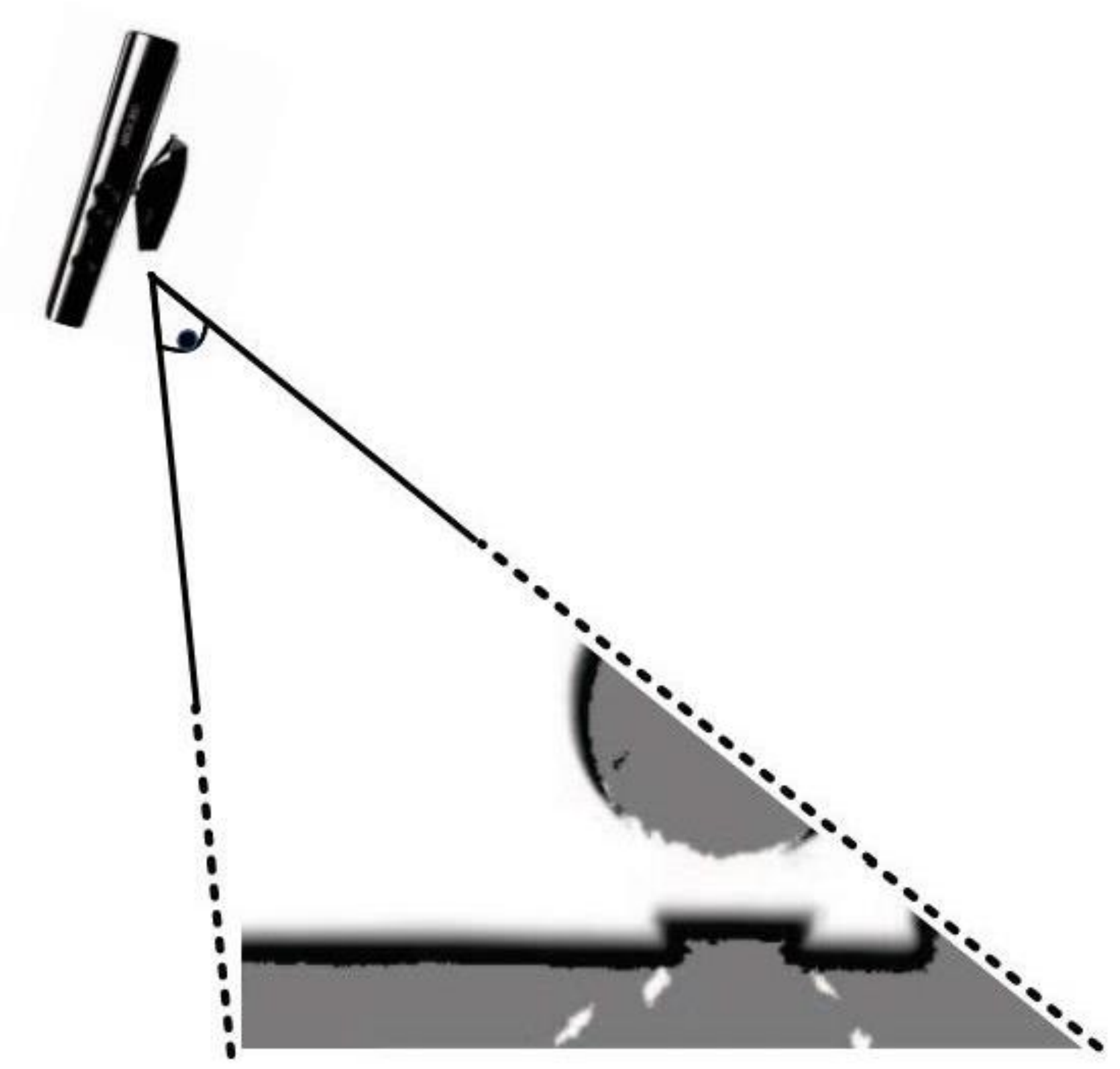
$$F_{R_k}(p) = \begin{cases} \eta/tr, & \text{if } |\eta| \leq tr, \\ \text{sign}(\eta), & \text{otherwise.} \end{cases}$$

$$W_{R_k}(p) \propto \cos(\theta)/D(x)$$

Weight is higher if closer and viewed from a perpendicular viewpoint.

# TSDF Integration

Each depth image provides a partial (noisy) observation of TSDF.



Volume integration recursively update the map given new observation.

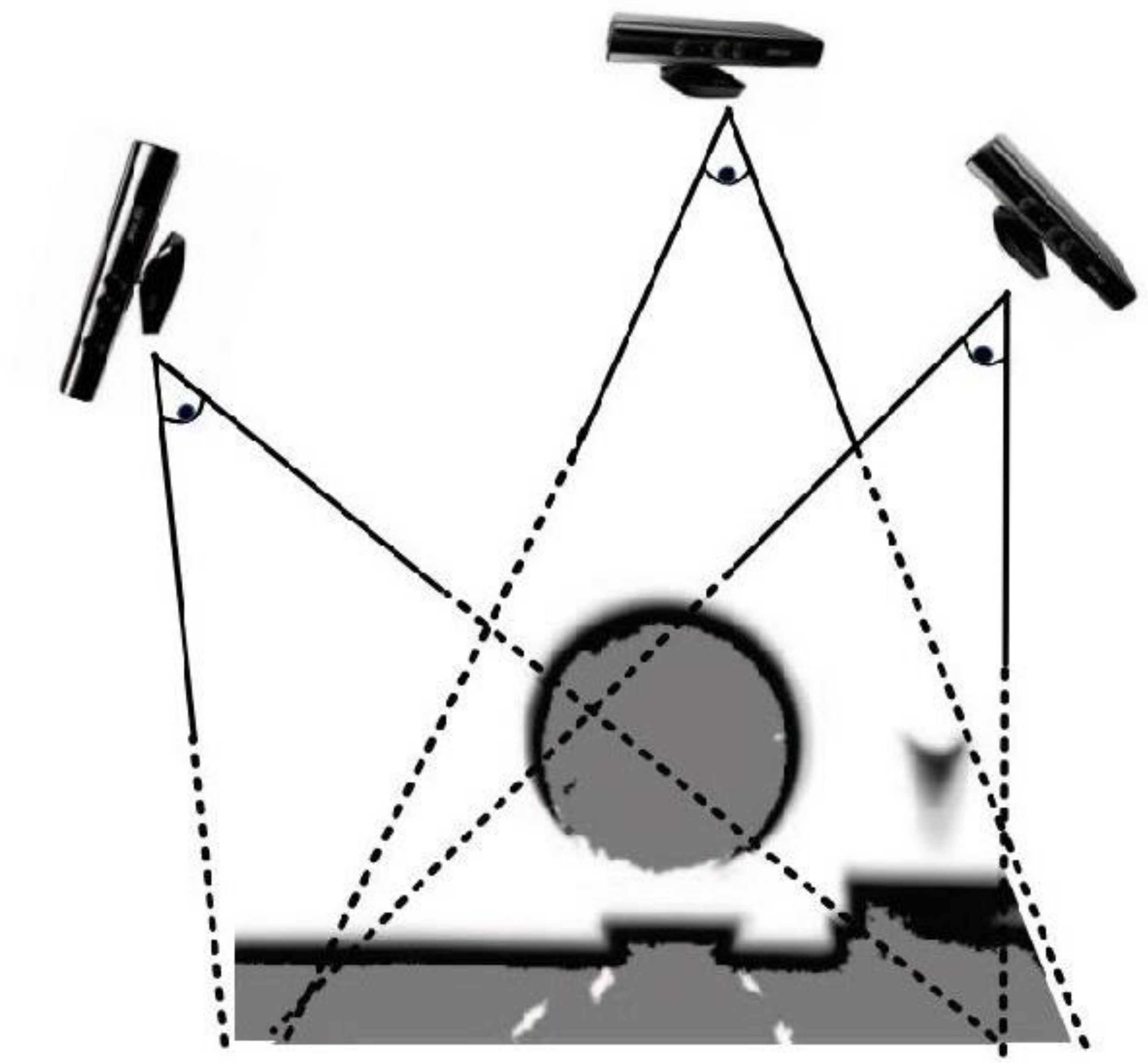


Image credit: Zhi-Hao Lin

# TSDF Integration

Updated TSDF

Old TSDF

New Observation

$$F_k(\mathbf{p}) = \frac{W_{k-1}(\mathbf{p})F_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})F_{R_k}(\mathbf{p})}{W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})}$$

$$W_k(\mathbf{p}) = W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})$$

Weight

Running weighted average implemented on GPU

(65 gigavoxels/sec, OR, ≈ 2ms per full volume update for a $512^3$ voxel reconstruction)

Image credit: Zhi-Hao Lin

# KinectFusion (2011)

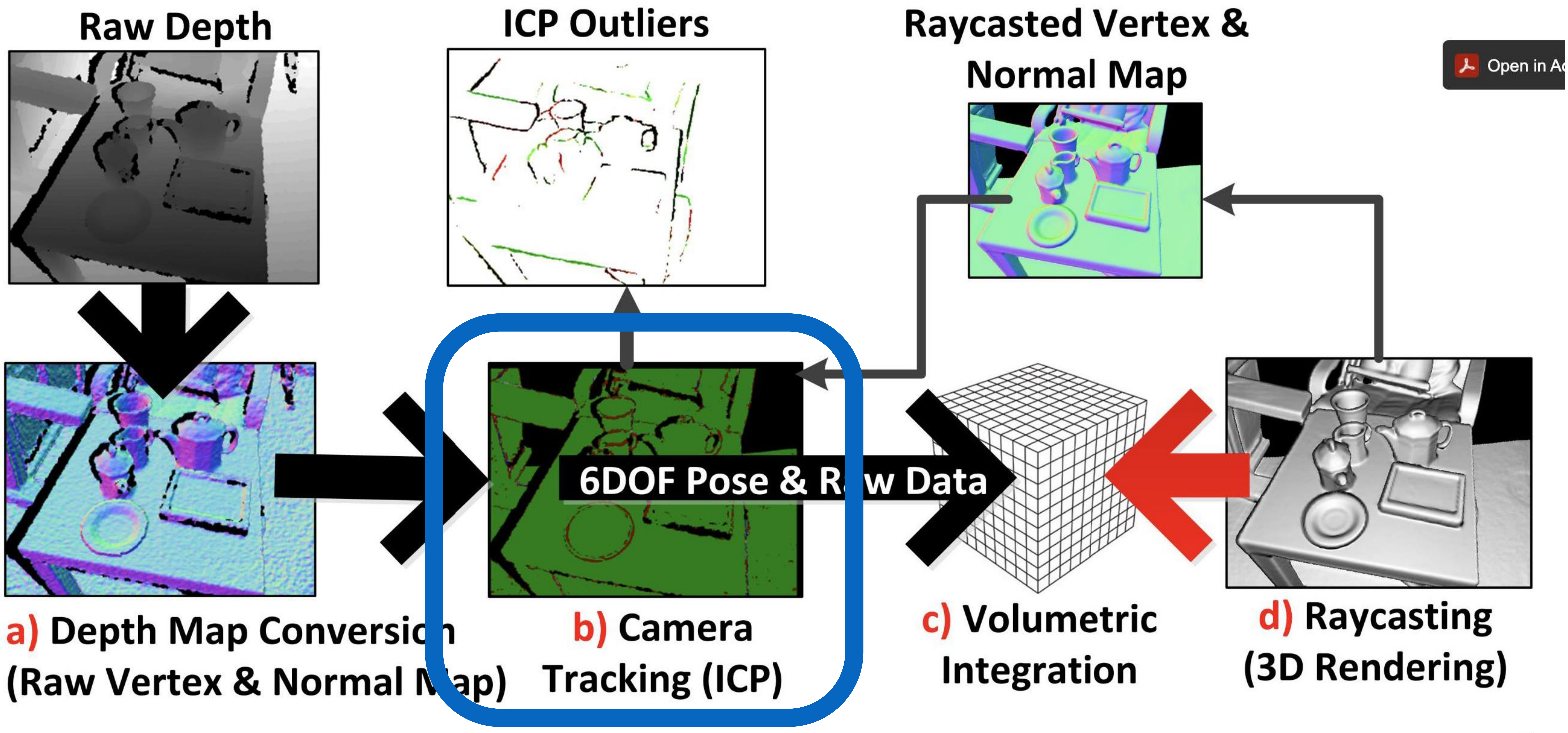



Image credit: Newcombe

# Frame-to-Model Tracking is better than Frame-to-Frame Tracking

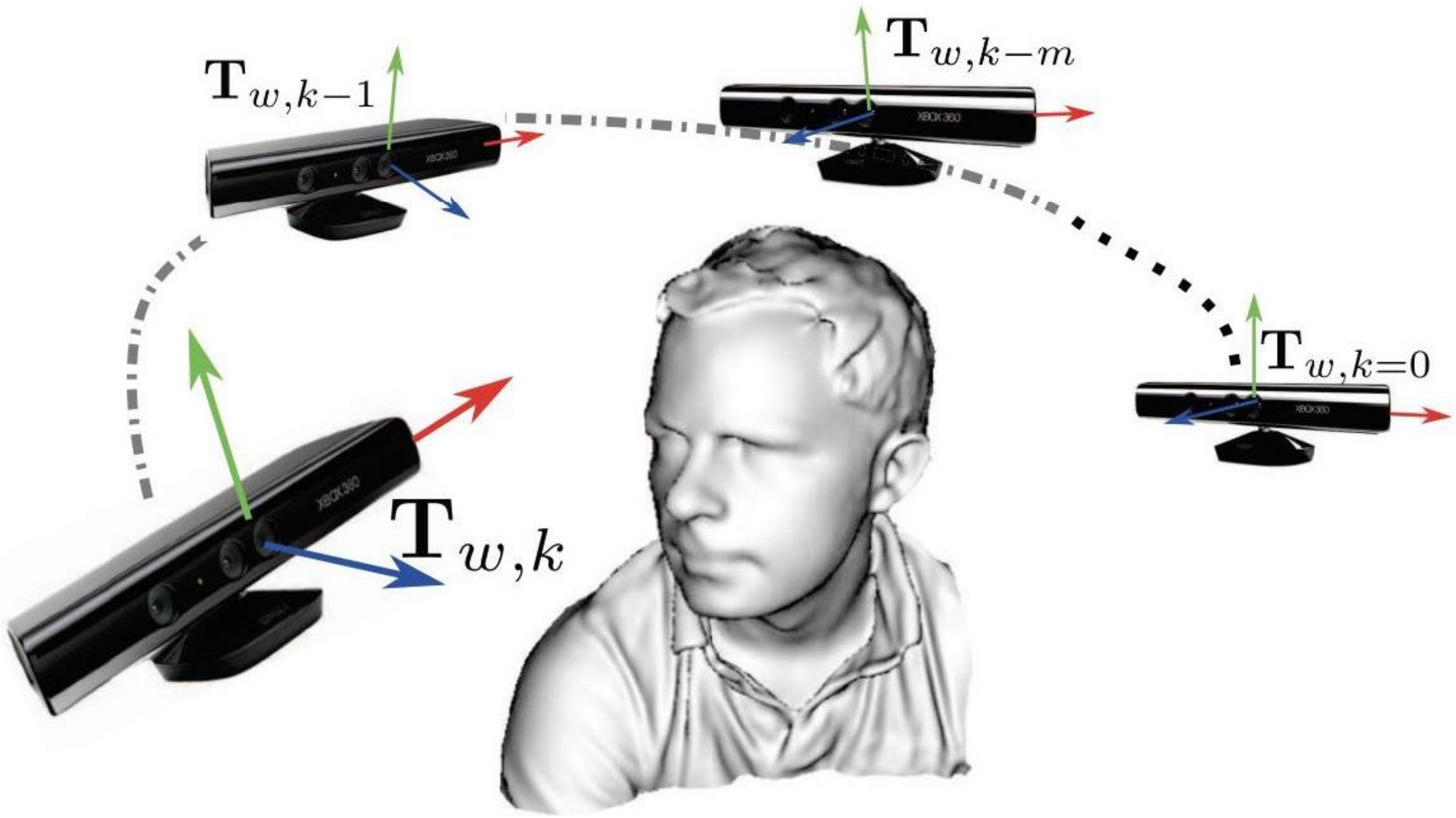
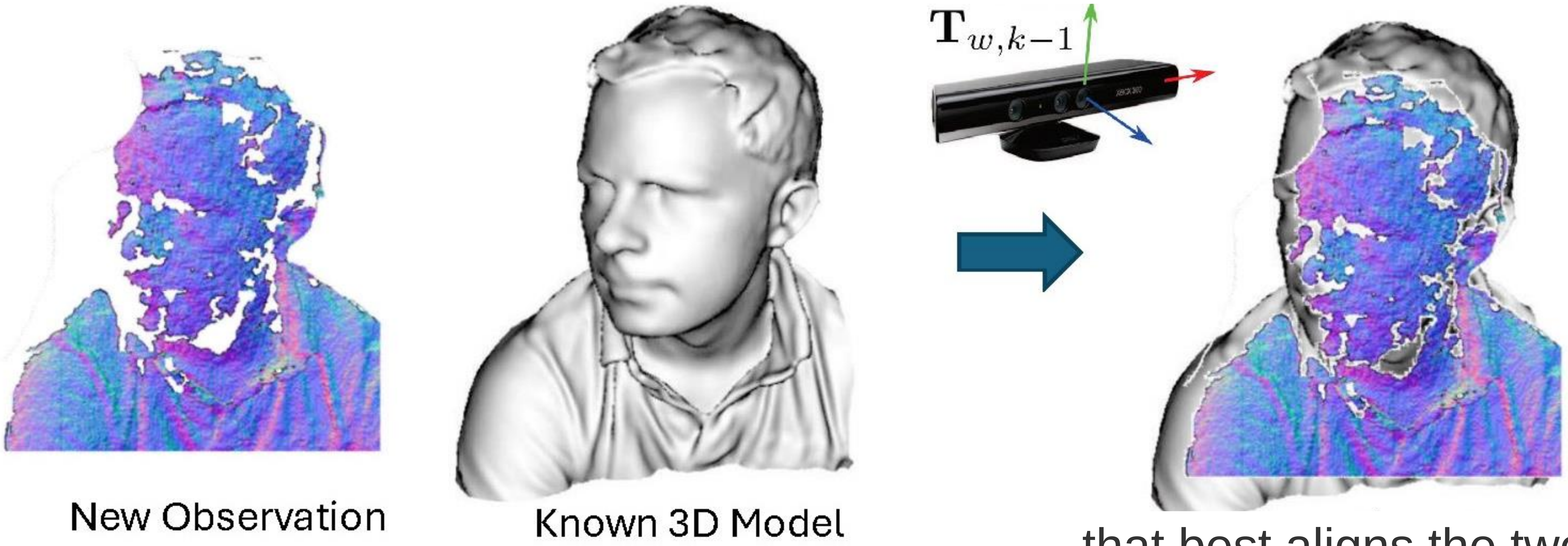



Image credit: Richard Newcombe

# Frame-to-Model Camera Tracking

Find rigid pose transformation…



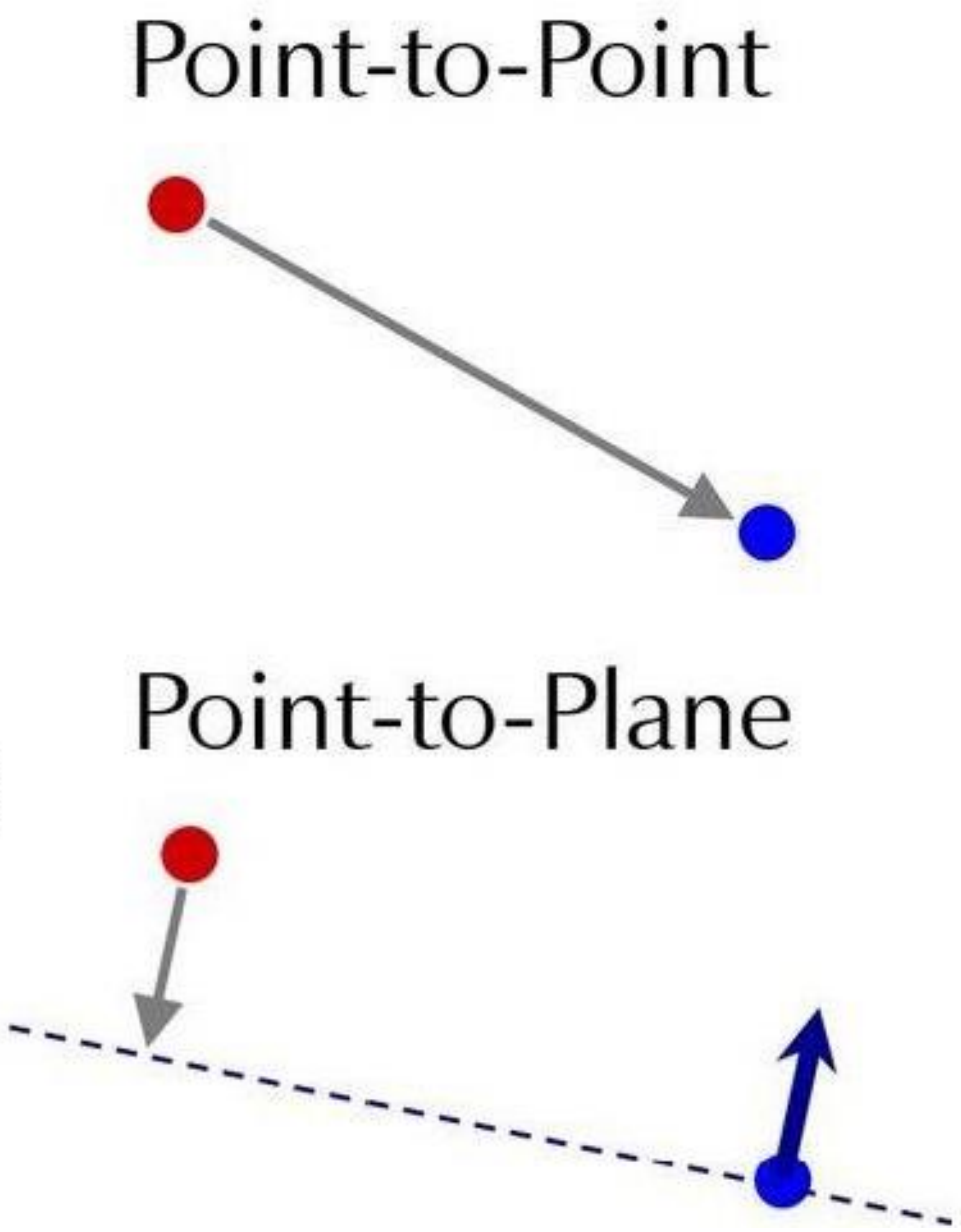

that best aligns the two.

Image credit: Richard Newcombe

# Point-to-Plane ICP

$$\min_{\mathbf{T}} \sum_{i}^{N} \|\mathbf{T}\mathbf{p}_i - \mathbf{q}_i\|_2^2$$

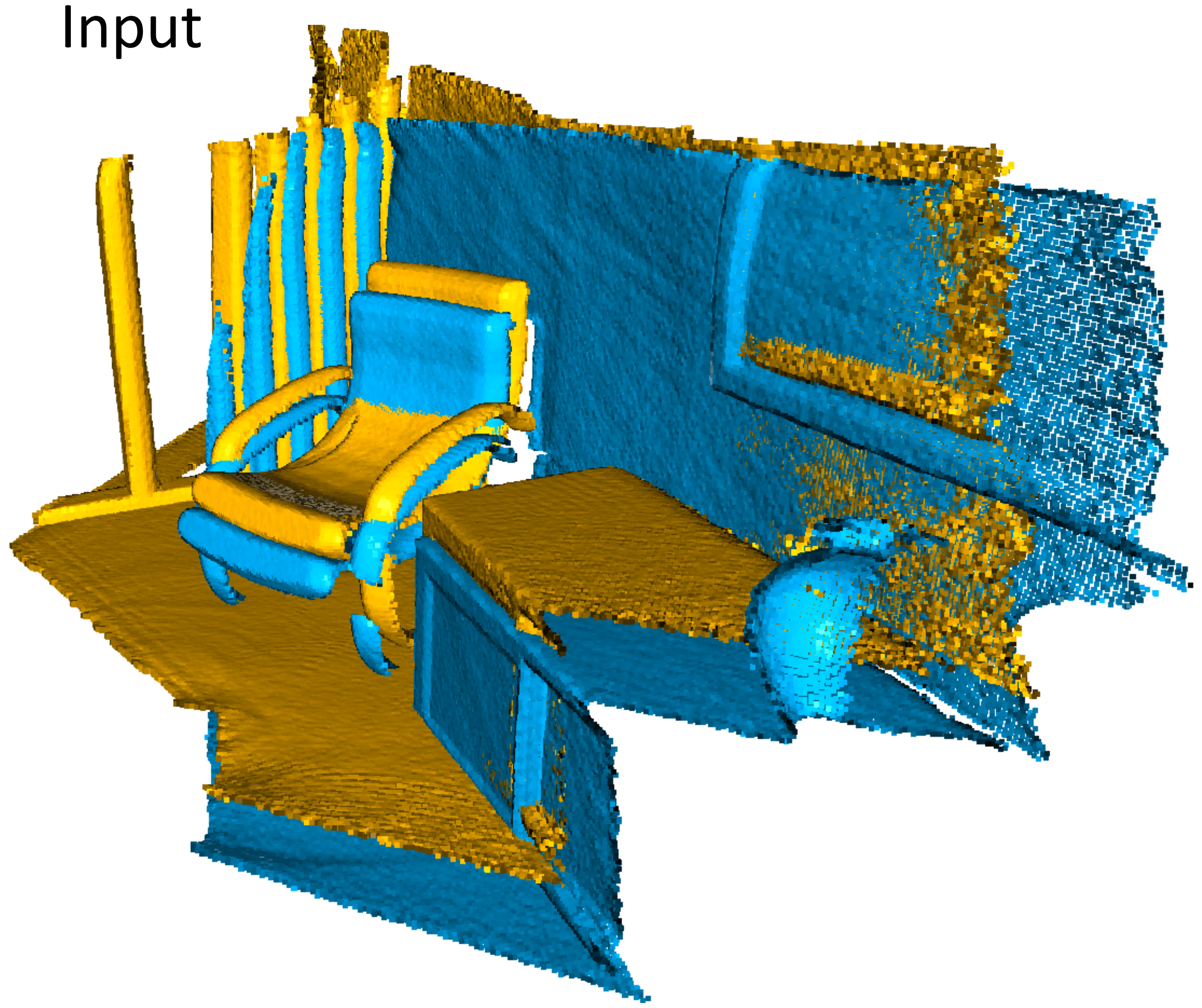
$$\min_{\mathbf{T}} \sum_{i}^{N} ((\mathbf{T}\mathbf{p}_i - \mathbf{q}_i) \cdot \mathbf{n}_{\mathbf{q}_i})^2$$




*observed* surface

*predicted* surface and normal

Input



Aligned with **Point-to-Plane ICP**

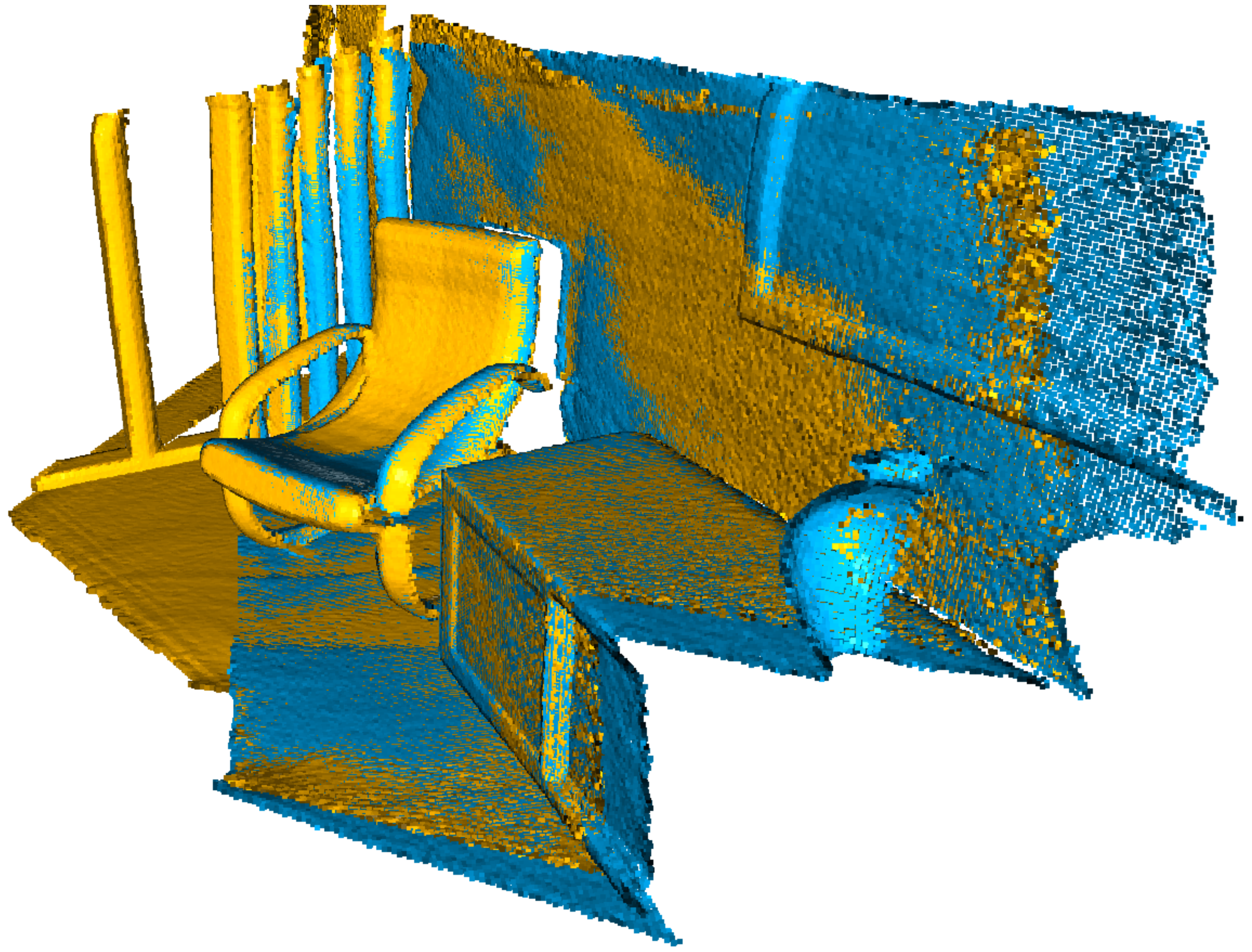


Image credit: Zhi-Hao Lin

# Point-to-Plane ICP via Gauss-Newton

$$\min_{T \in \mathrm{SE}(3)} \sum_i \left\| (Tp_i - q_i)^\top n_i \right\|_2^2$$

Using a first-order approximation for the rotation variable:

$$T = (R, t) = (\widehat{R} \exp(\varepsilon^\wedge), t) \approx (\widehat{R}(I + \varepsilon^\wedge), t)$$

Plug into $Tp_i - q_i$:

$$\begin{aligned} Tp_i - q_i &= Rp_i + t - q_i \\ &= \widehat{R}p_i + \widehat{R}\varepsilon^\wedge p_i + t - q_i \end{aligned}$$

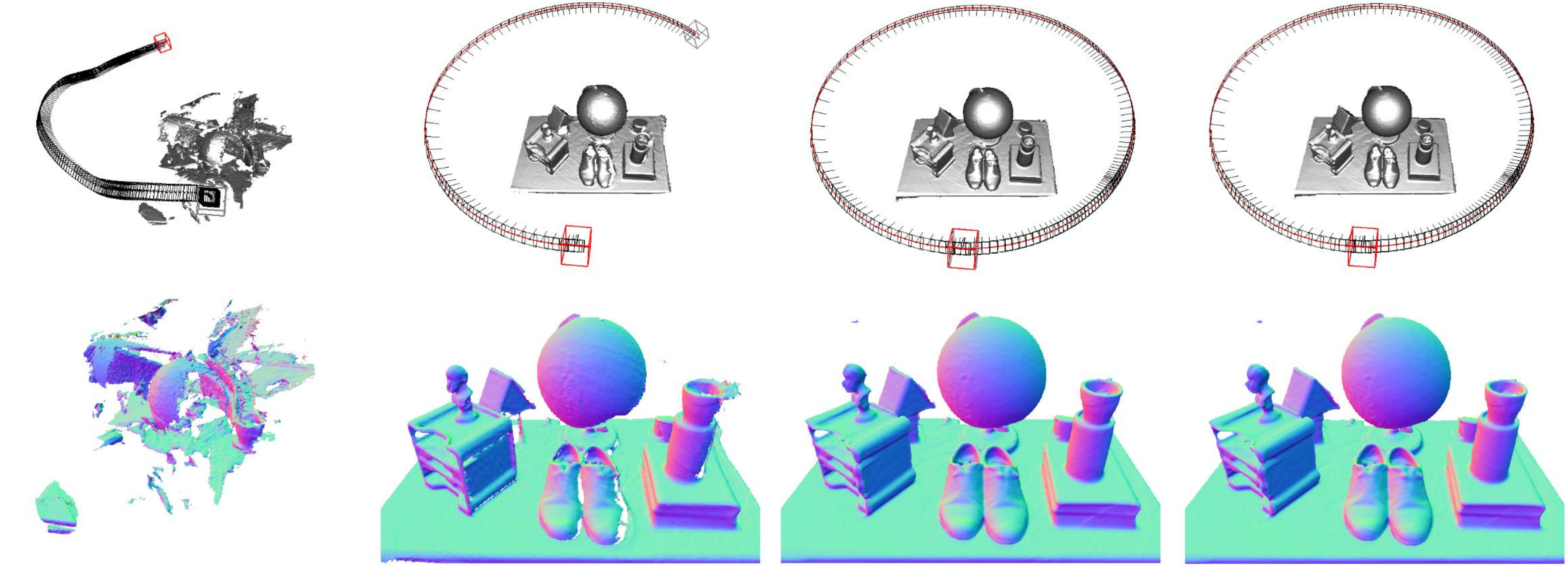
# Point-to-Plane ICP via Gauss-Newton

Plug into the overall cost function:

$$\begin{aligned}
&\sum_i \left\| (Tp_i - q_i)^\top n_i \right\|_2^2 \\
\approx &\sum_i \left\| (\widehat{R}p_i - q_i - \widehat{R}p_i{}^\wedge \varepsilon + t)^\top n_i \right\|_2^2 \\
= &\sum_i \left\| a_i^\top x + b_i \right\|_2^2,
\end{aligned}$$

where $x = (\varepsilon, t)$ is the decision variable, and $a_i, b_i$ are constant.
➔ This is a least squares optimization ➔ implemented on GPU and **run in frame rate (30 Hz)**

# Frame-to-Model Tracking is better than Frame-to-Frame Tracking

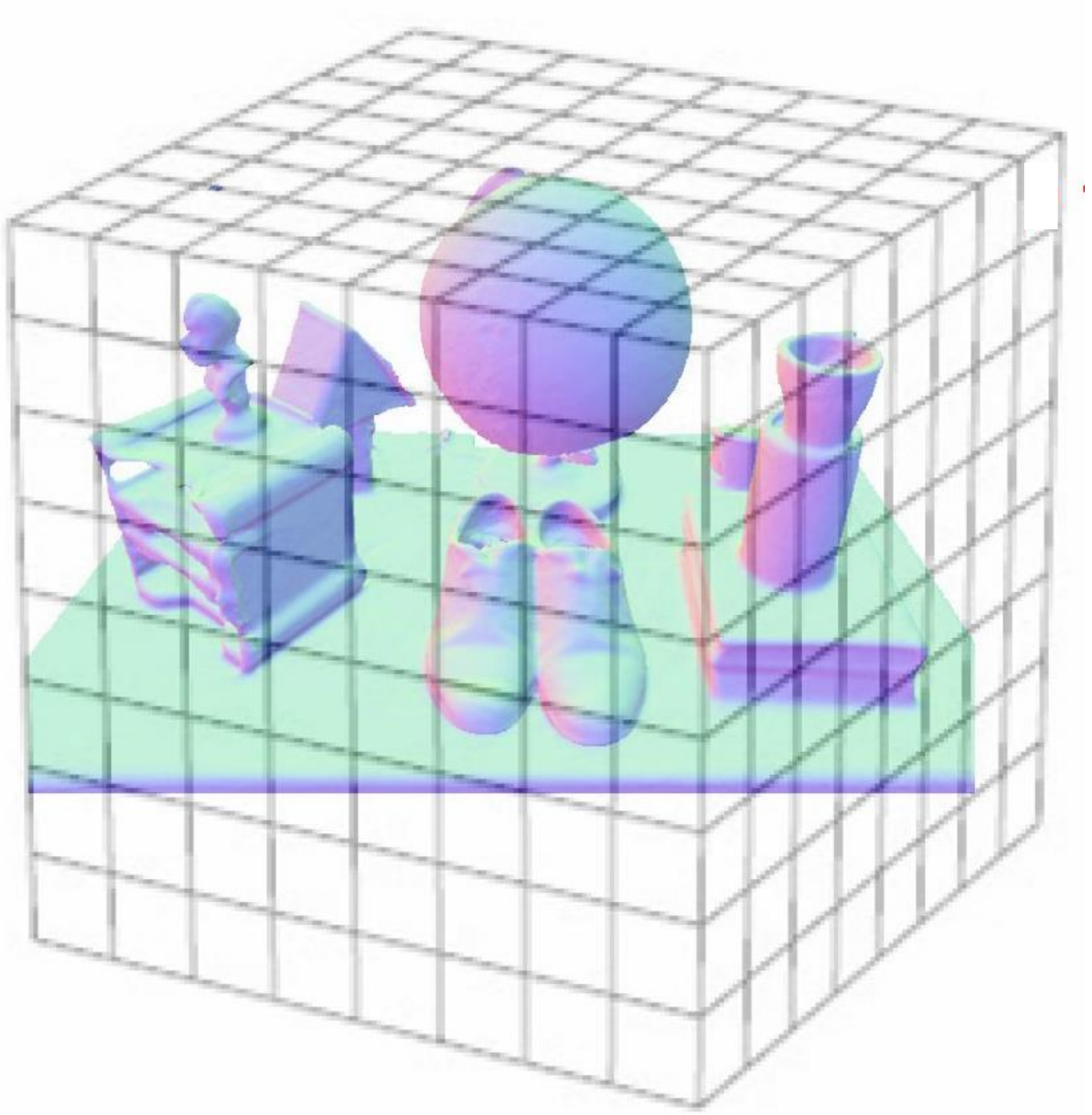


(a) Frame to frame tracking (b) Partial loop (c) Full loop (d) M times duplicated loop

# Full System

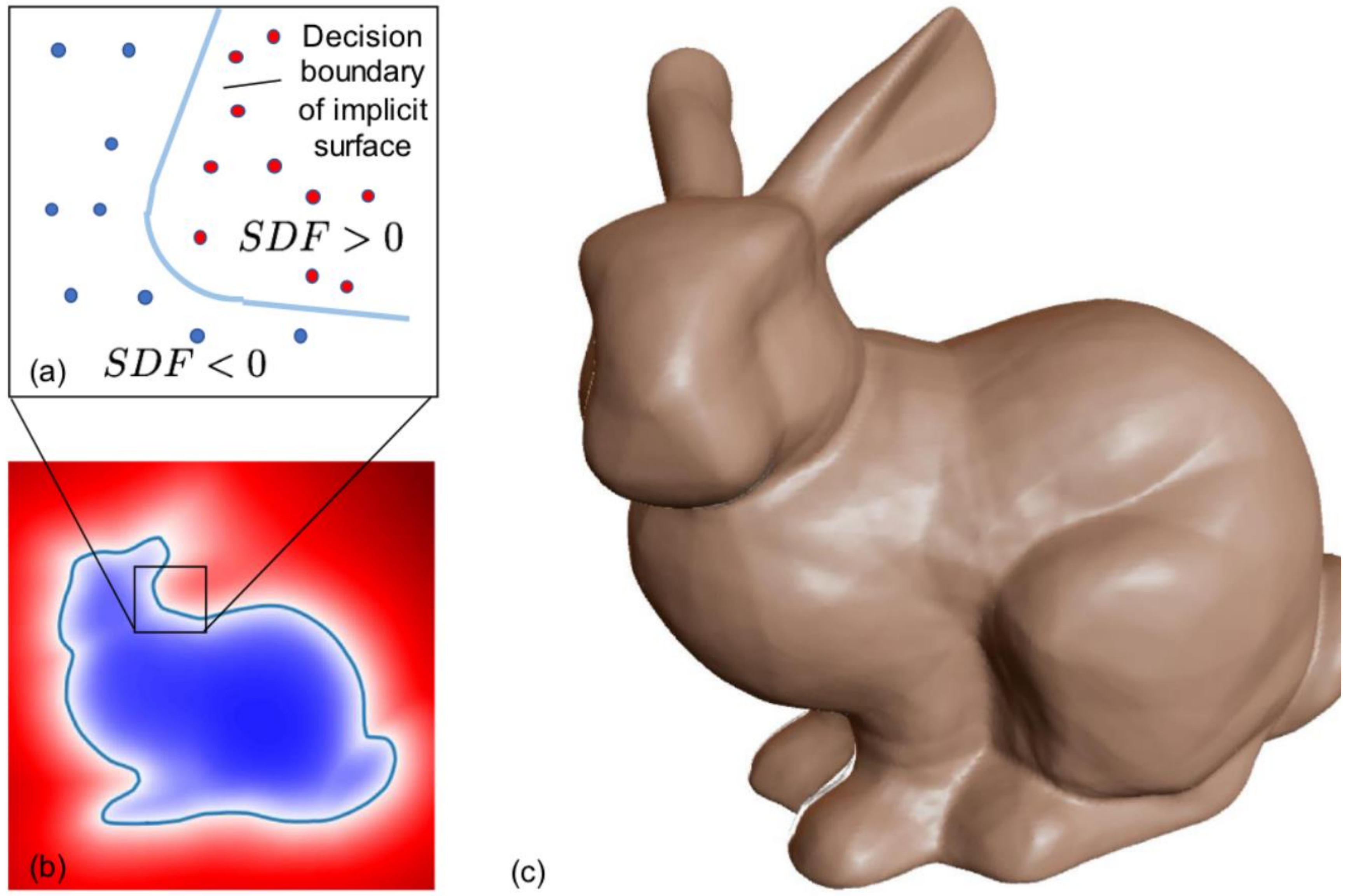
# KinectFusion: Summary

- First real-time dense RGB-D SLAM on GPU
- Mapping: fast TSDF integration
- Localization: frame-to-model tracking via point-to-plane ICP
- 🤔 How to resolve the scaling issue due to regular grid (for storing TSDF)?

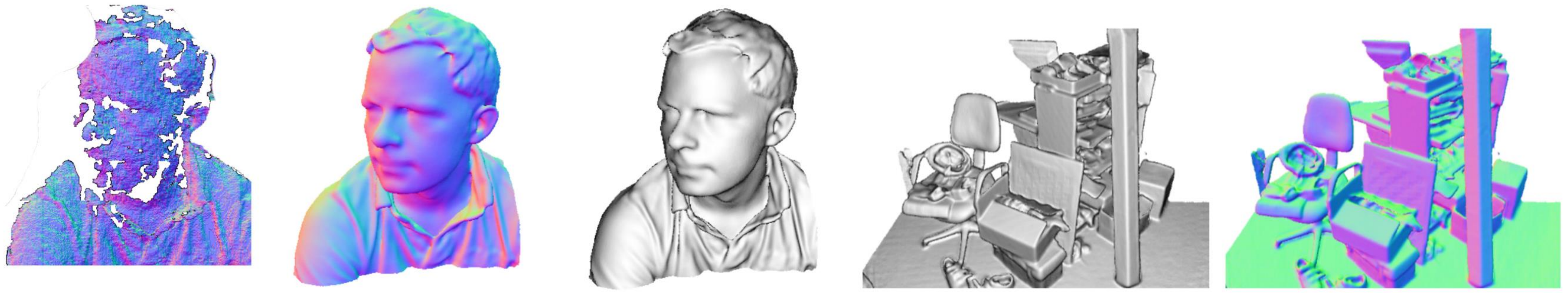


# Today’s Lecture

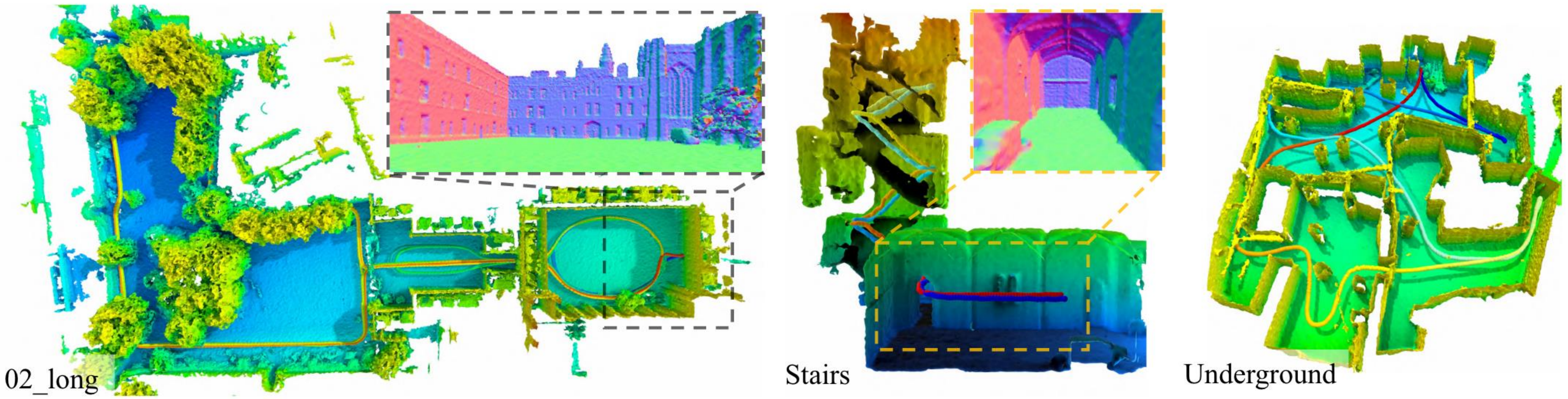
- Dense signed distance function (SDF) representation and properties
- Basics of 3D dense SLAM using SDF
- Recent advancements to improve SDF-based SLAM




Park et al. 2019



Newcombe et al. 2011




Pan et al. 2024

# Beyond Regular Grid: **Hierarchical Sparse Grids**

- Octree:
  - Recursively divide voxel into 8 child voxels
  - Increase spatial resolution as needed
  - Retrieval has $O(\log n)$ complexity
  - Can be used to store occupancy [1], TSDF values [2], and distribution over semantic categories [3]

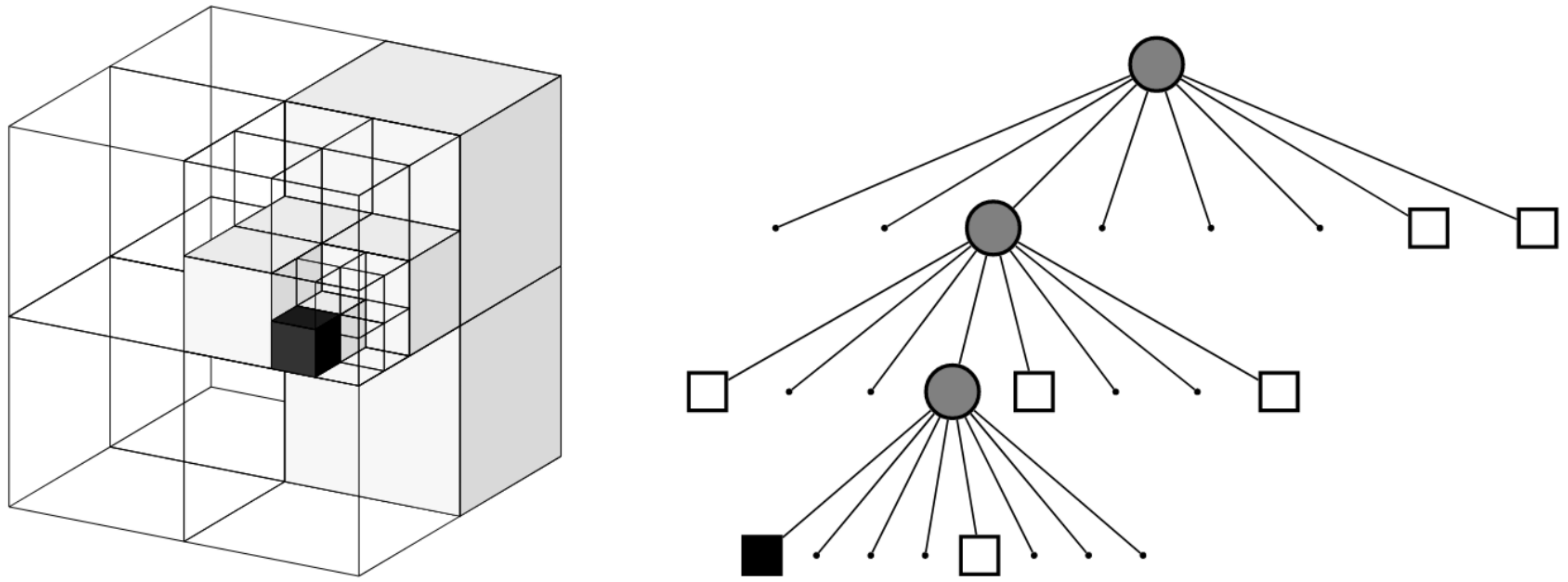


**Fig. 2** Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left and the corresponding tree representation on the right.

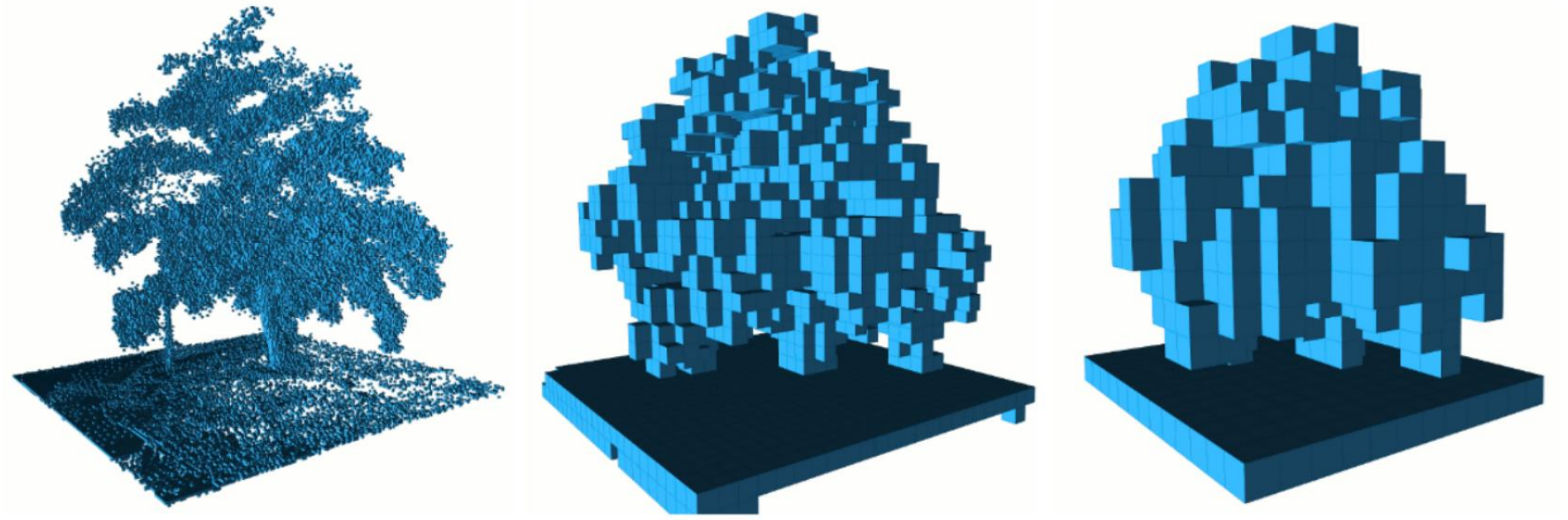


**Fig. 3** By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

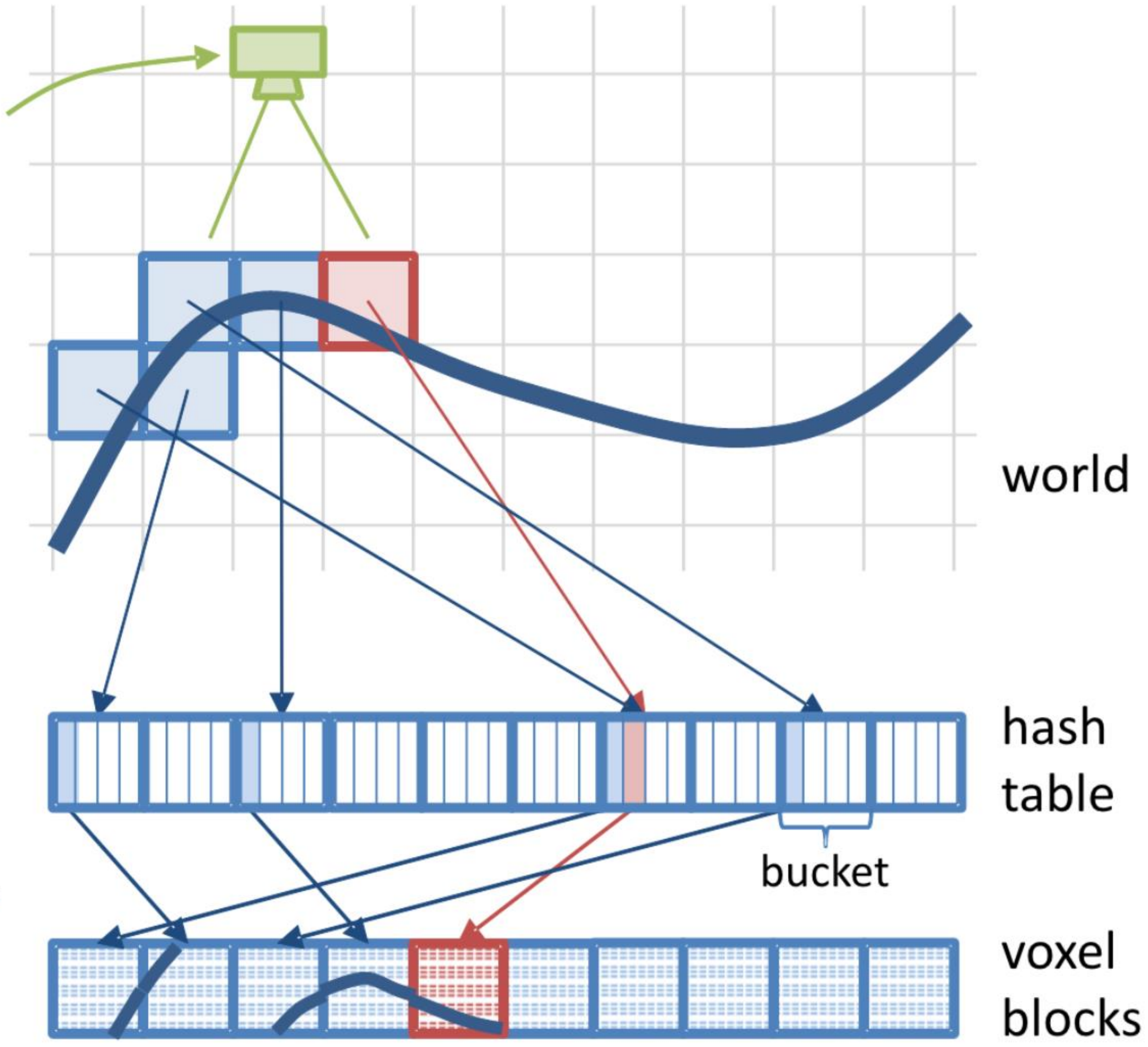
[1] Hornung et al. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." Autonomous robots 2013.

[2] Vespa et al. "Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping." IEEE RA-L 2018.

[3] Asgharivaskasi and Atanasov. "Semantic octree mapping and Shannon mutual information computation for robot exploration." IEEE T-RO, 2023.

# Beyond Regular Grid: **Voxel Hashing**

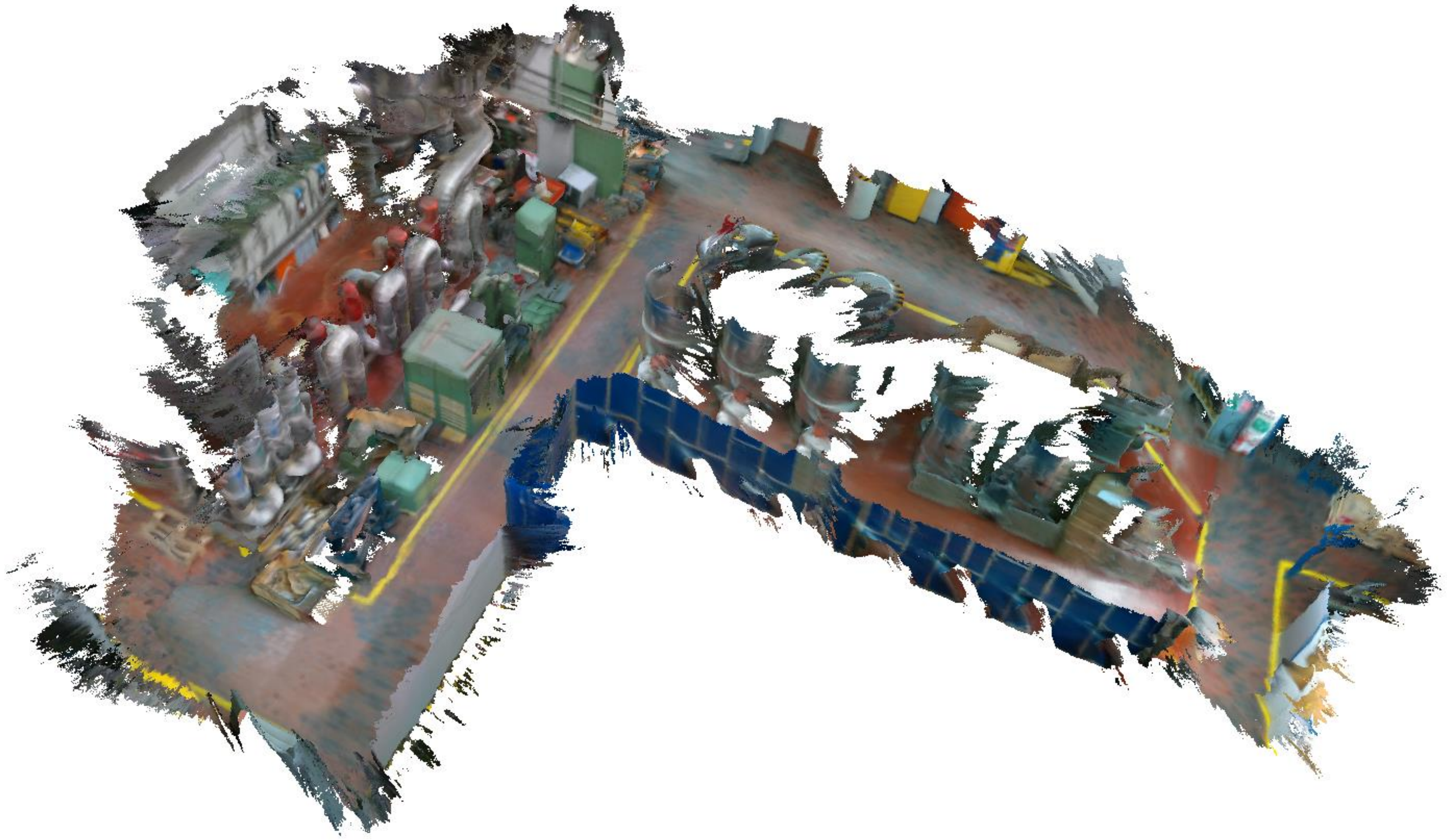
- Map (integer) 3D coordinates to hash values
- Retrieve voxel block from hash table
- Pro: constant time operation
- Con: need to handle hash collisions




Nießner et al. “Real-time 3D Reconstruction at Scale using Voxel Hashing.” ACM Transactions on Graphics 2013.

# Beyond Regular Grid: Voxel Hashing Implementation

voxblox

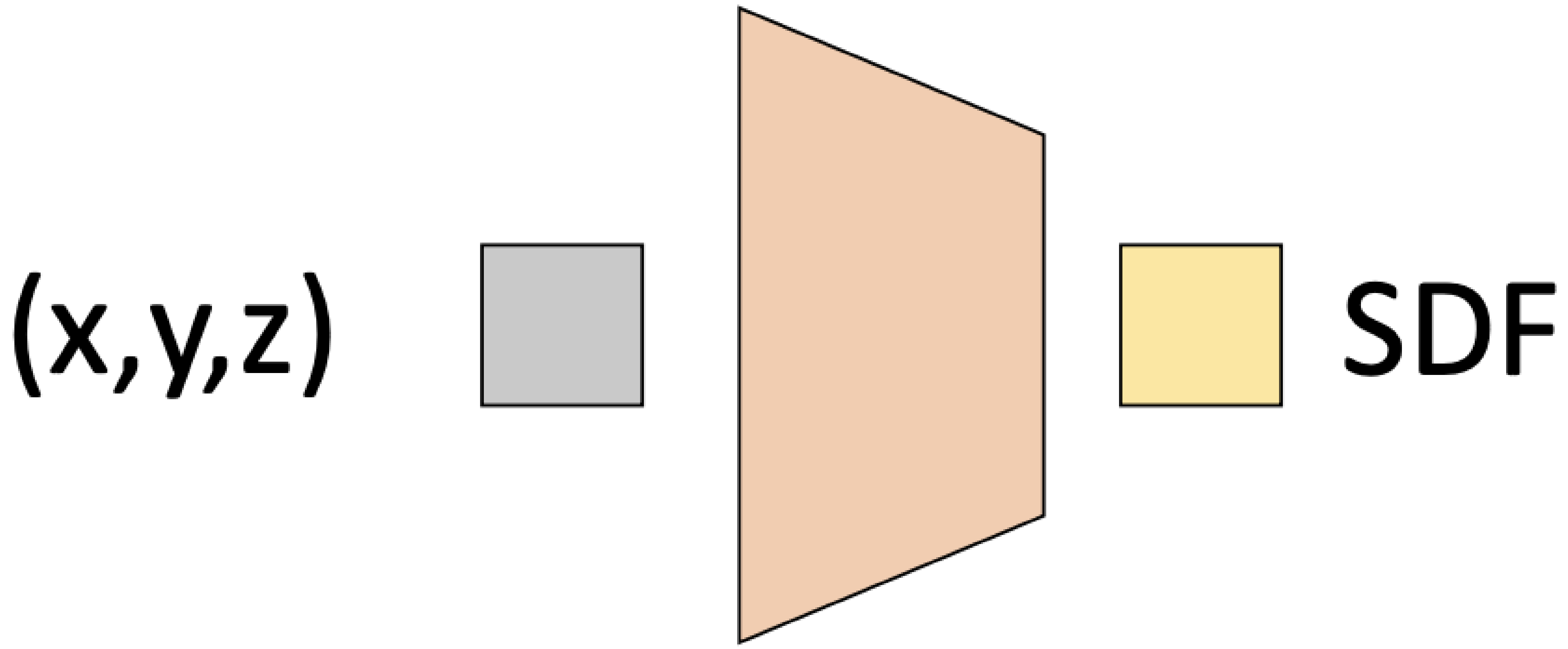


Oleynikova et al. “Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning.” IROS 2017.

Millane et al. “nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping.” ICRA 2024.

# DeepSDF: SDF via Neural Networks

- Use a multi-layer perceptron (MLP) to represent a **single shape**
- Input: 3D coordinate
- Output: SDF value
- **Differentiable by design!**



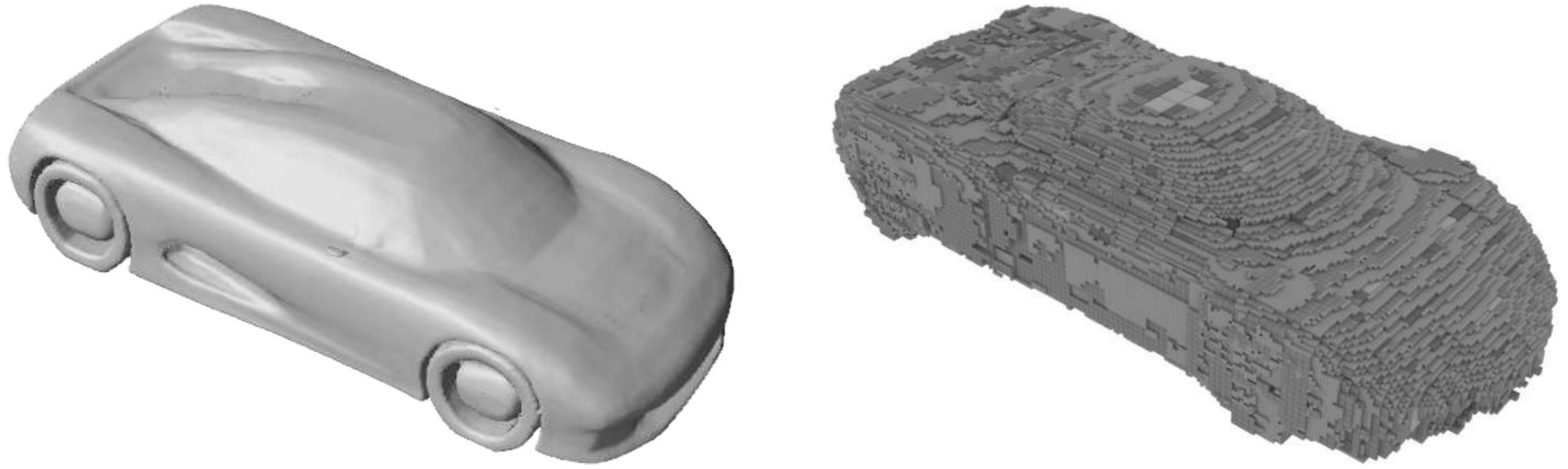
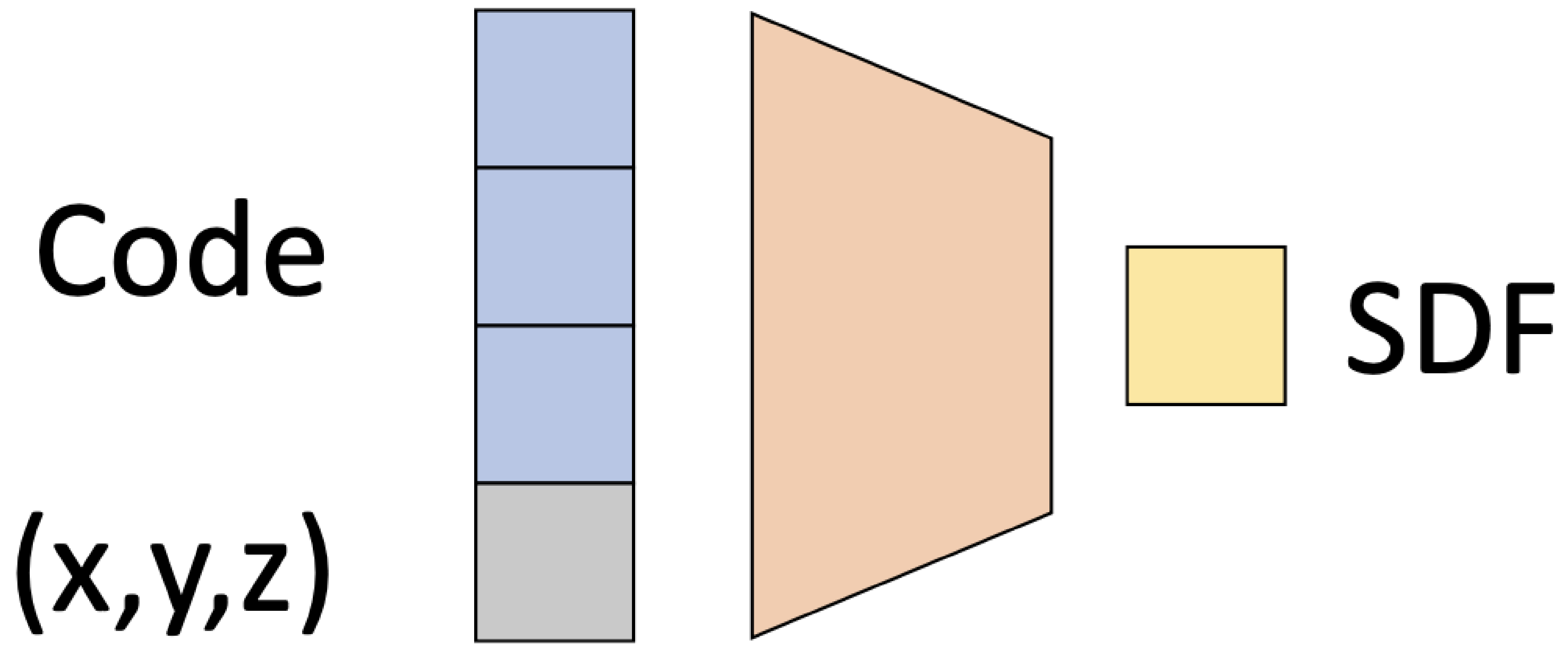



**Figure 5:** Compared to car shapes memorized using OGN [49] (right), our models (left) preserve details and render visually pleasing results as DeepSDF provides oriented surace normals.

Park et al. “Deepsdf: Learning continuous signed distance functions for shape representation.” CVPR 2019.

# DeepSDF: SDF via Neural Networks

- Use a multi-layer perceptron (MLP) to represent multiple shapes
- Input: 3D coordinate + *learnable latent code*
- Output: SDF value
- Support shape interpolation



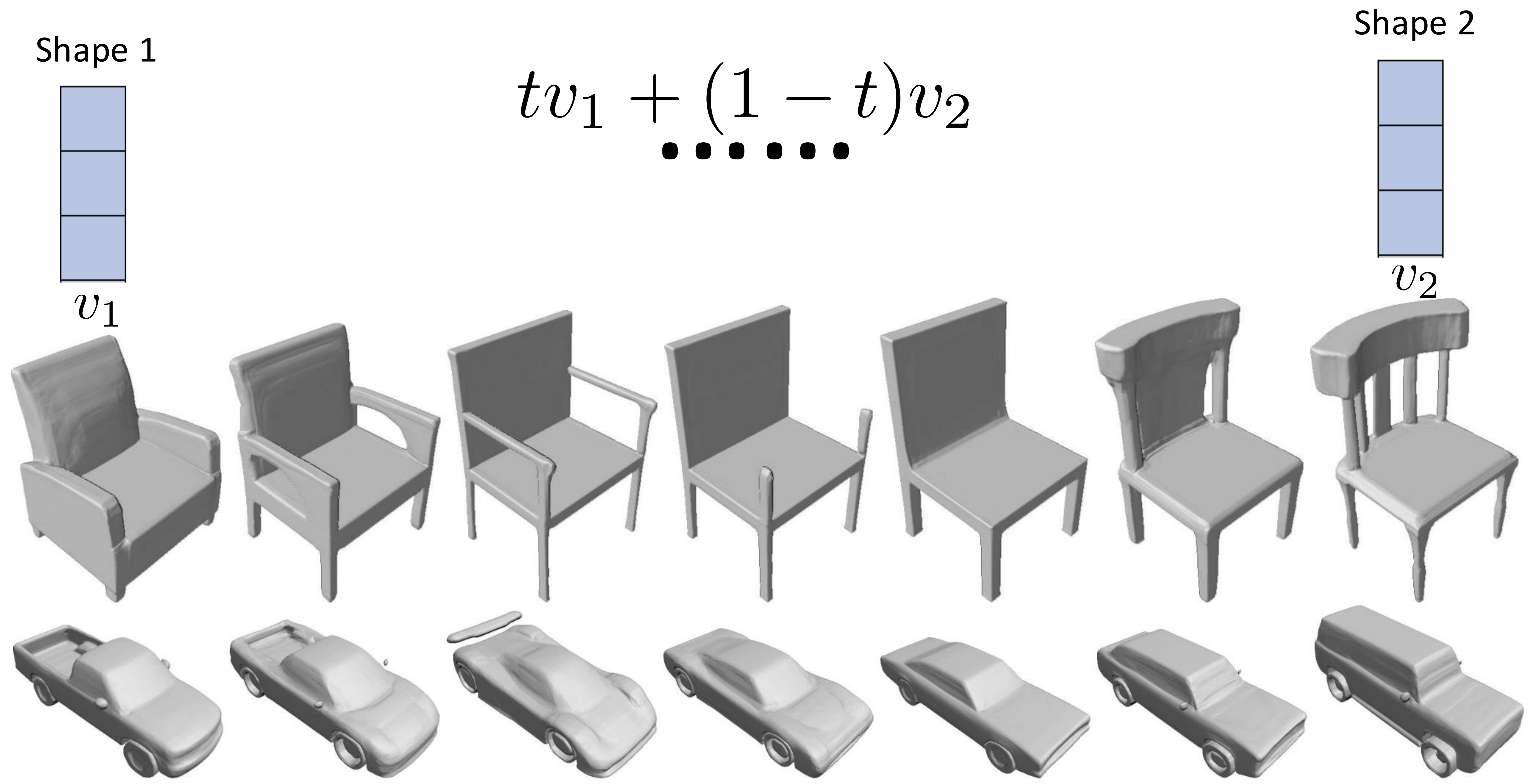
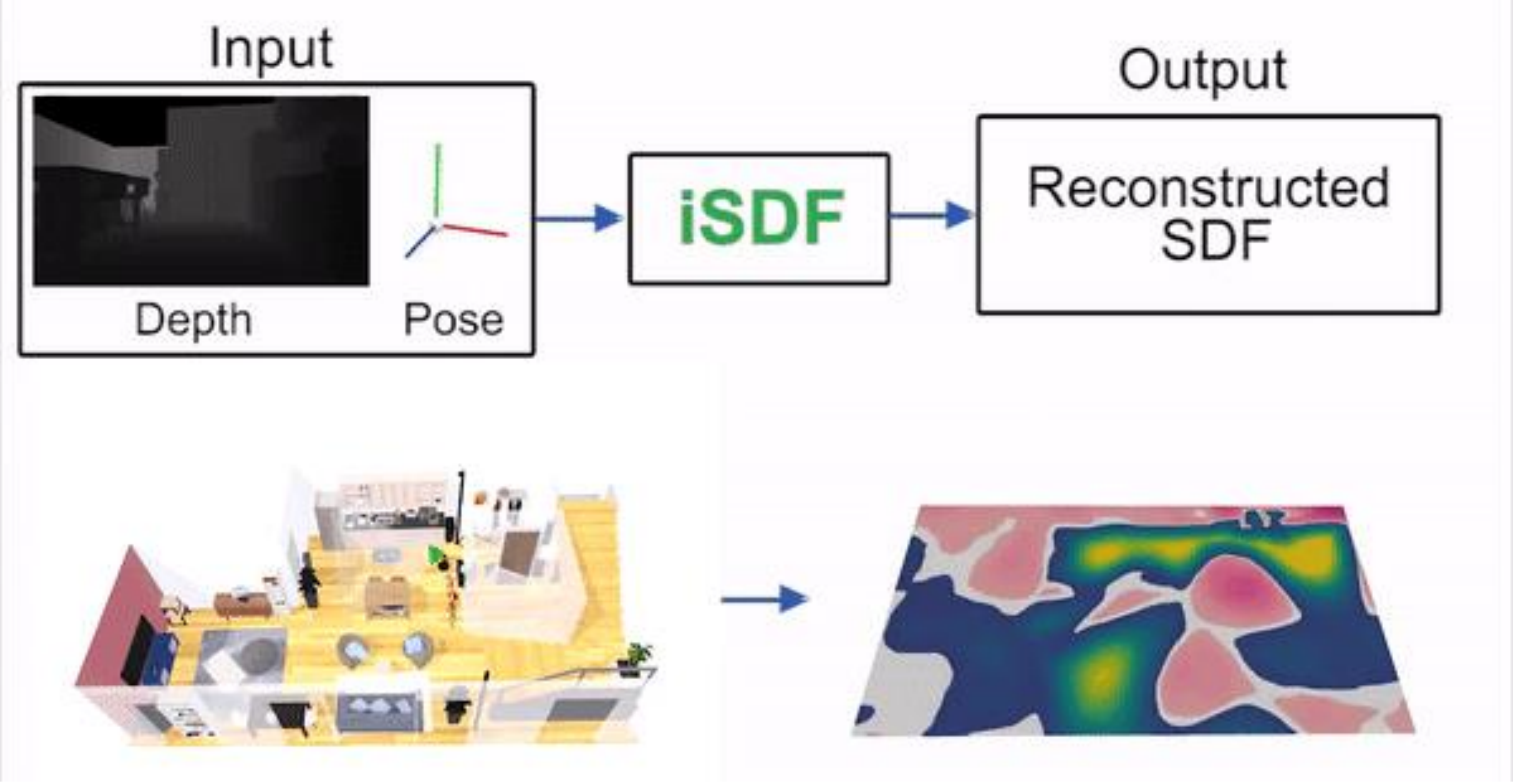



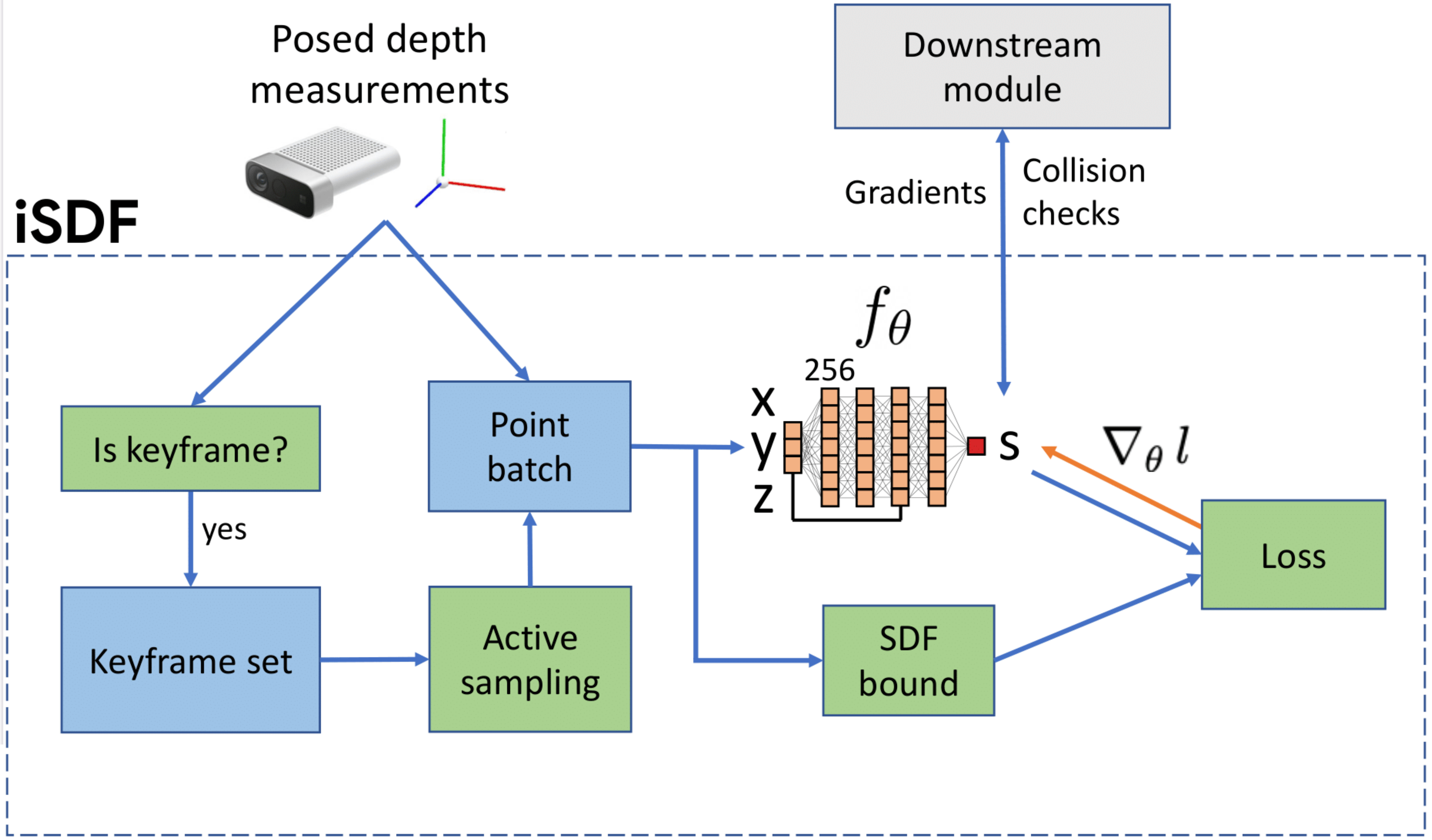

**Figure 1:** DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation." CVPR 2019.

# iSDF: neural SDF for real-time mapping

- Assume external localization (e.g., provided by external odometry module)
- Mapping using neural SDF representation

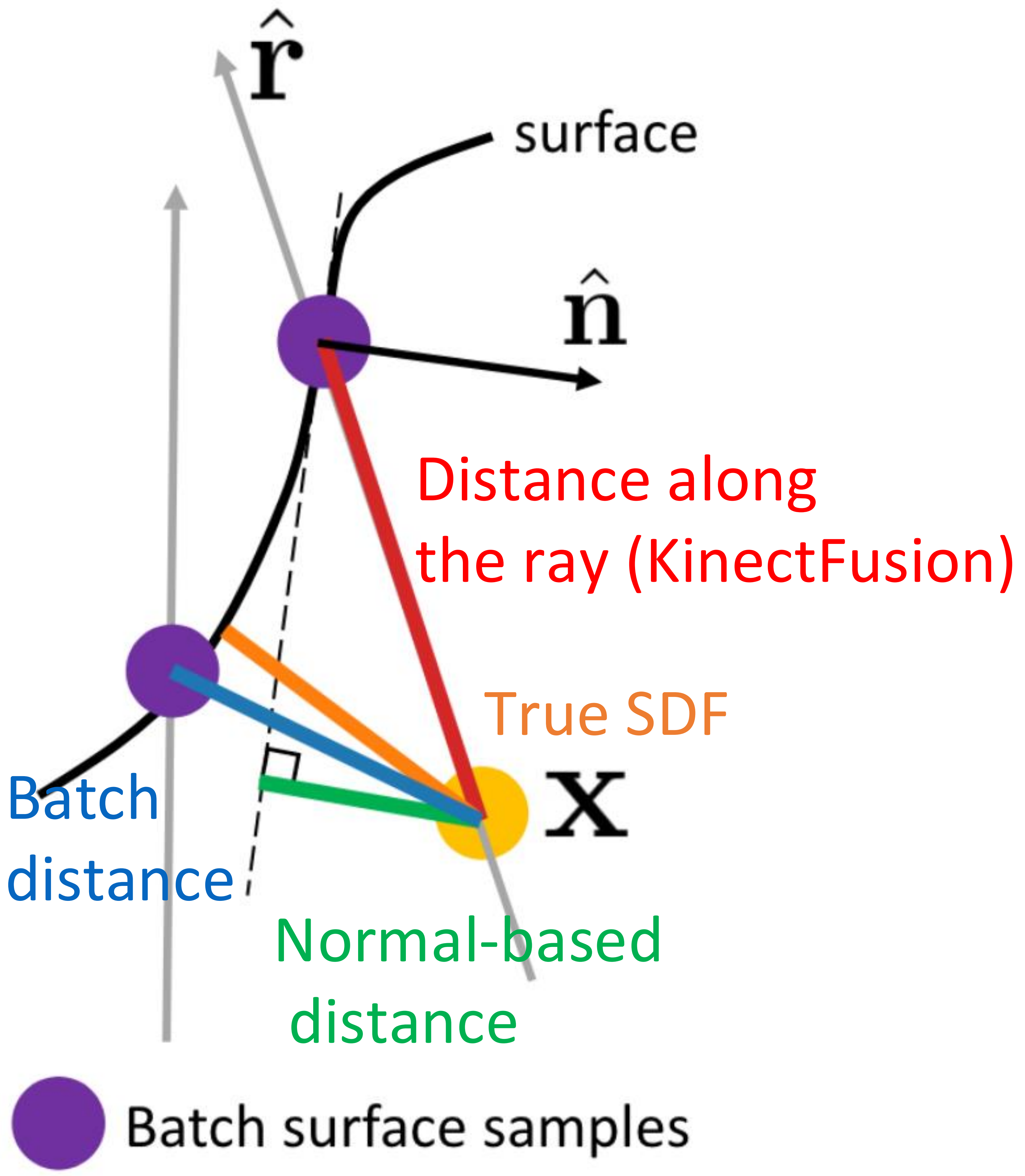



Ortiz et al. "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception." RSS 2022.

# iSDF: neural SDF for real-time mapping

- Improved loss function for training the SDF network




$$l(\theta) = \boxed{\mathcal{L}_{\text{sdf}}} + \lambda_{\text{grad}}\,\mathcal{L}_{\text{grad}} + \lambda_{\text{eik}}\,\mathcal{L}_{\text{eik}}\ .$$

$$\mathcal{L}_{\text{sdf}}(f(\mathbf{x};\theta), b) = \begin{cases} \lambda_{\text{surf}}\mathcal{L}_{\text{near\_surf}} & \text{if } |D[u,v] - d| \leq t \\ \mathcal{L}_{\text{free\_space}} & \text{otherwise.} \end{cases}$$

Near surface: use measurement as direct supervision

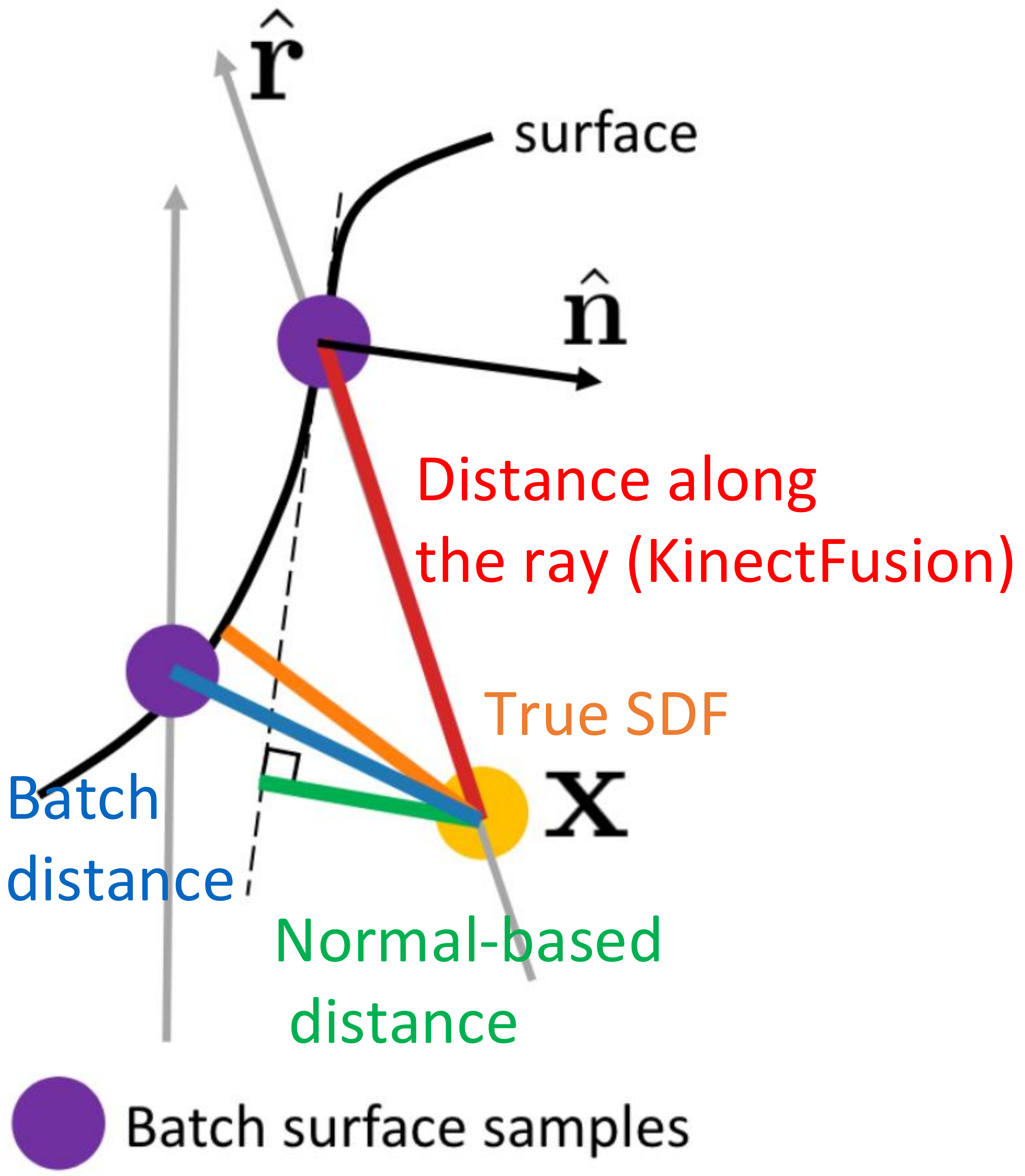
$$\mathcal{L}_{\text{near\_surf}}(f(\mathbf{x};\theta), b) = |f(\mathbf{x}_i;\theta) - b|\ .$$

Far away from surface: use measurement as upper bound

$$\mathcal{L}_{\text{free\_space}}(f(\mathbf{x};\theta), b) = \max\left(0,\ e^{-\beta f(\mathbf{x}_i;\theta)} - 1,\ f(\mathbf{x}_i;\theta) - b\right)\ .$$

# iSDF: neural SDF for real-time mapping

- Improved loss function for training the SDF network




$$l(\theta) = \mathcal{L}_{\text{sdf}} + \lambda_{\text{grad}} \boxed{\mathcal{L}_{\text{grad}}} + \lambda_{\text{eik}} \mathcal{L}_{\text{eik}} \ .$$

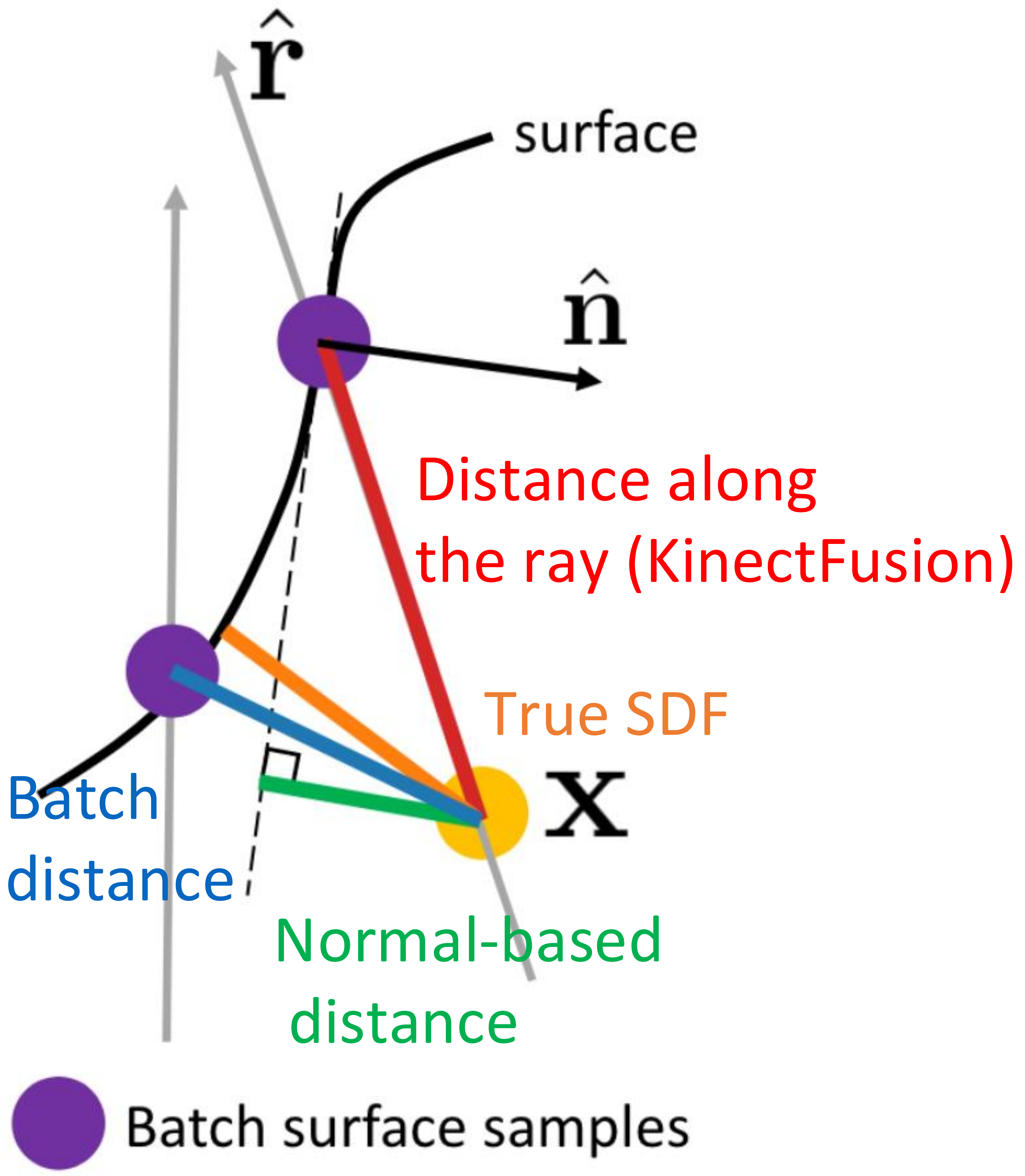
Maximize cosine similarity between observed and predicted normals

$$\mathcal{L}_{\text{grad}}(\nabla_{\mathbf{x}} f(\mathbf{x};\theta), \mathbf{g}) = 1 - \frac{\nabla_{\mathbf{x}} f(\mathbf{x};\theta) \cdot \mathbf{g}}{\|\nabla_{\mathbf{x}} f(\mathbf{x};\theta)\| \|\mathbf{g}\|} \ .$$

Ortiz et al. "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception." RSS 2022.

# iSDF: neural SDF for real-time mapping

- Improved loss function for training the SDF network



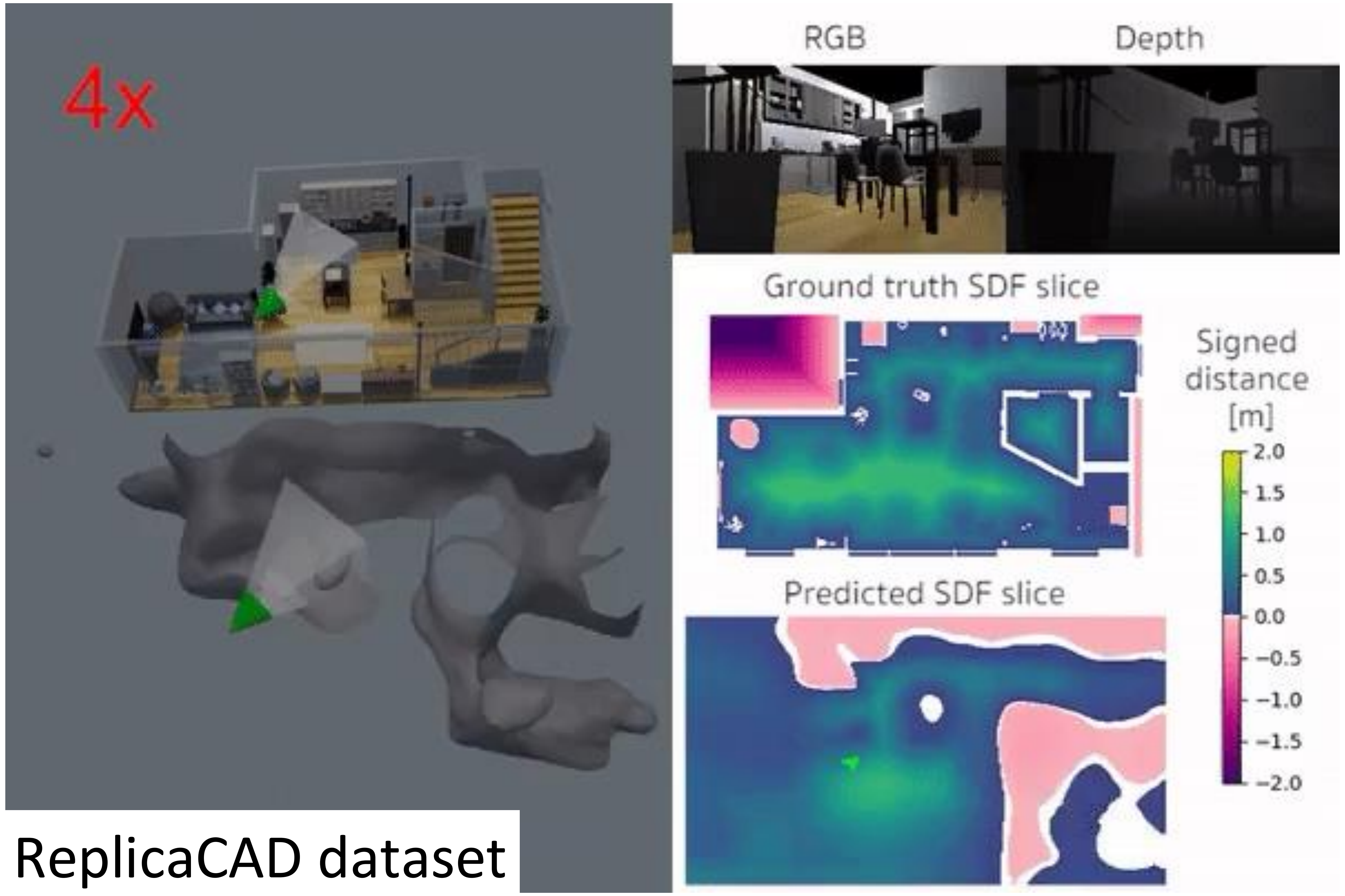

$$l(\theta) = \mathcal{L}_{\text{sdf}} + \lambda_{\text{grad}}\,\mathcal{L}_{\text{grad}} + \lambda_{\text{eik}}\boxed{\mathcal{L}_{\text{eik}}}.$$

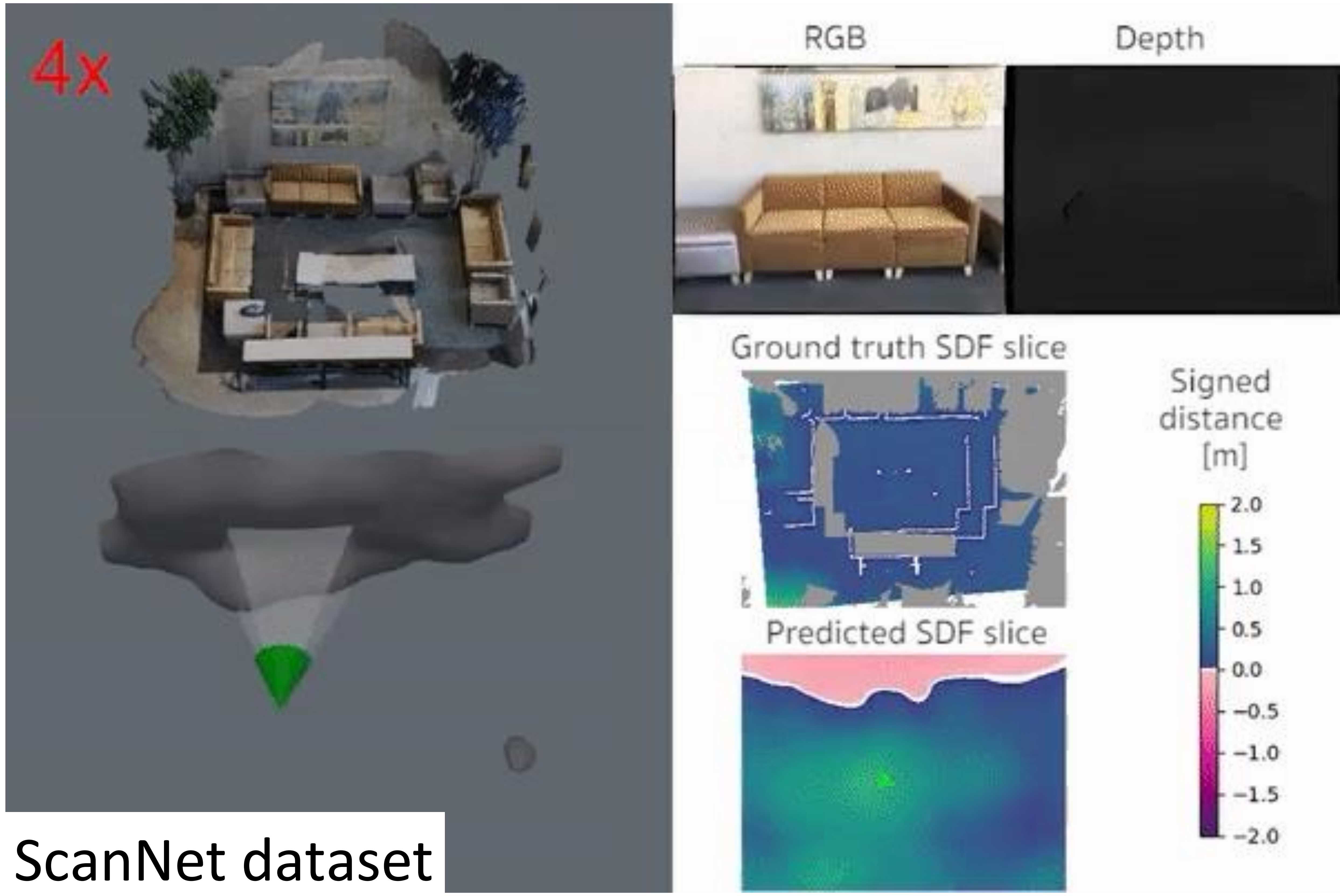
Enforce Eikonal property of the learned SDF

$$\mathcal{L}_{\text{eik}}(f(\mathbf{x};\theta)) = \begin{cases} \left|\, \|\nabla_{\mathbf{x}} f(\mathbf{x};\theta)\| - 1 \right| & \text{if } |D[u,v] - d| \geq a \\ 0 & \text{otherwise.} \end{cases}$$

Ortiz et al. “iSDF: Real-Time Neural Signed Distance Fields for Robot Perception.” RSS 2022.

# iSDF: neural SDF for real-time mapping



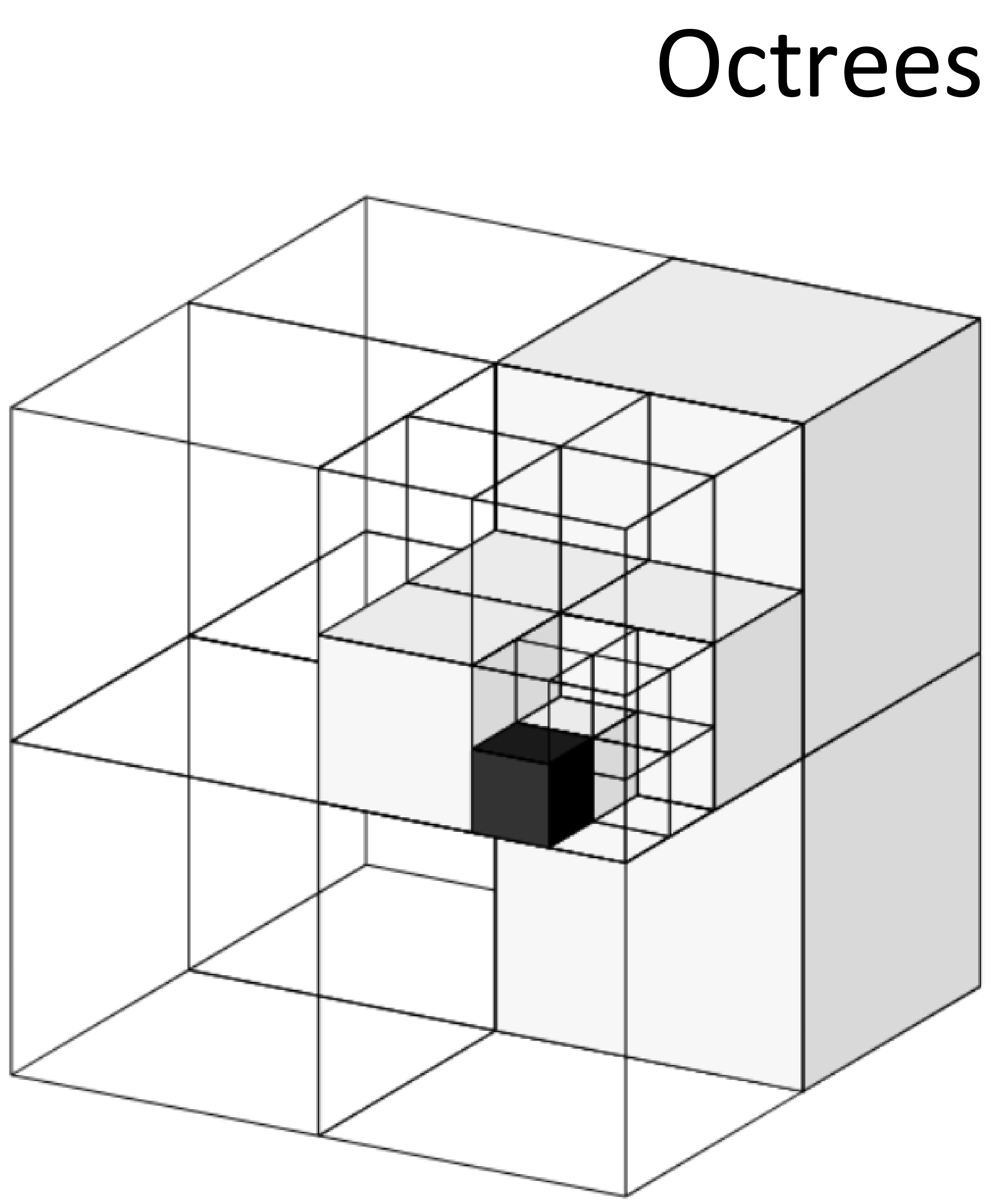



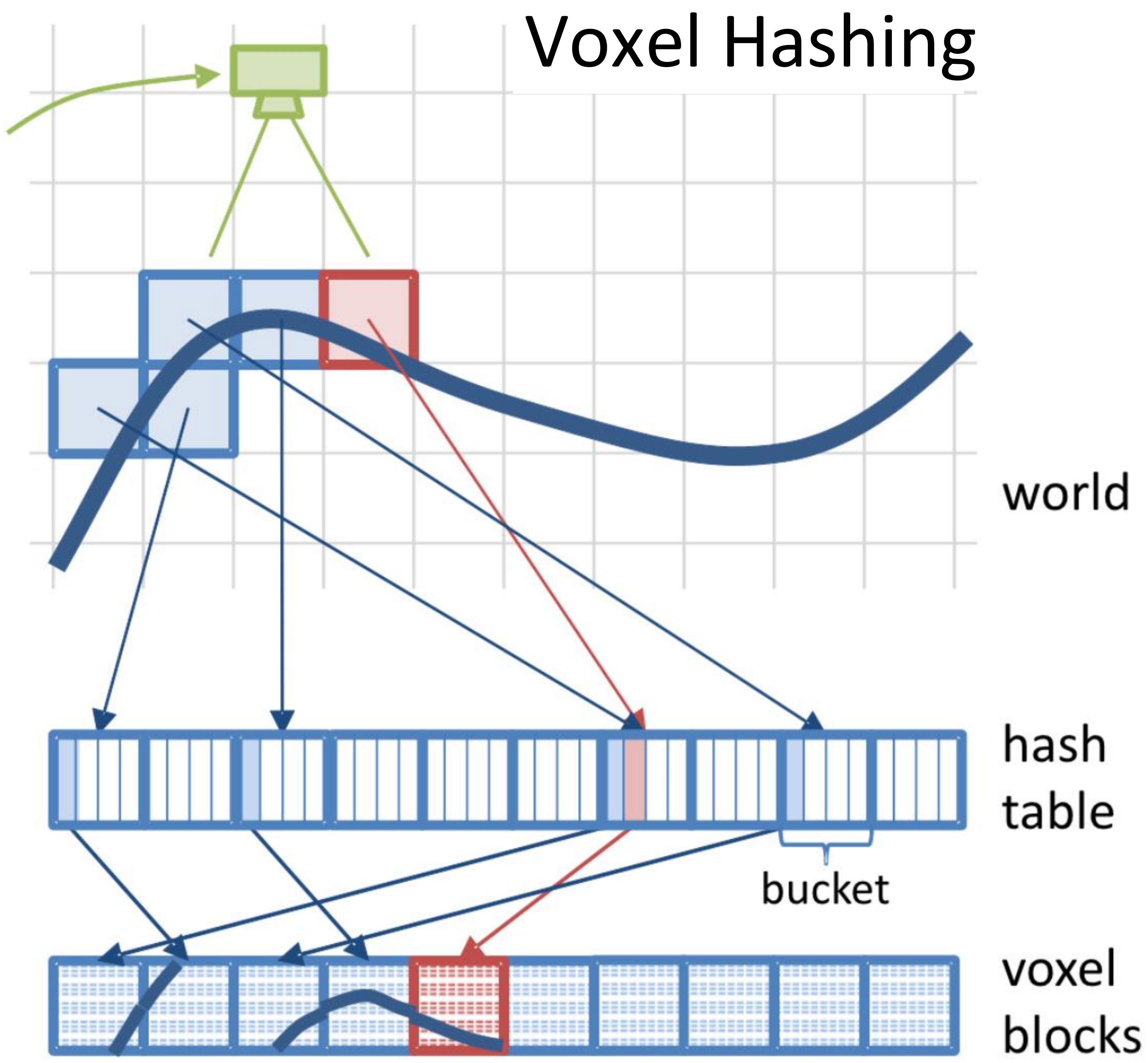

- MLP has limited memory capacity
  - Struggle to fit geometry with fine details
  - The “catastrophic forgetting” problem

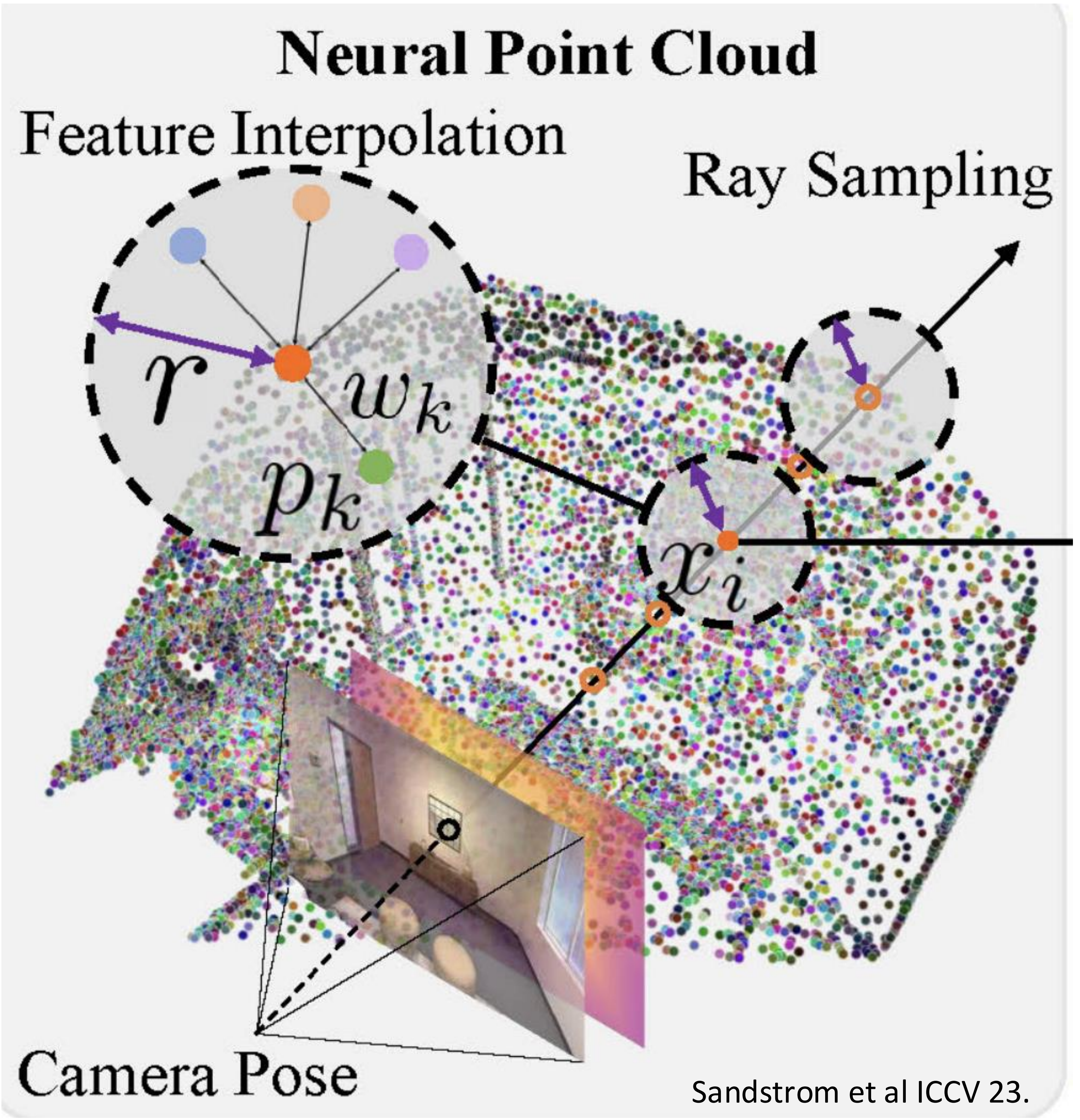
Ortiz et al. “iSDF: Real-Time Neural Signed Distance Fields for Robot Perception.” RSS 2022.

# Hybrid Representations

- Extend neural SDF with additional data structures!

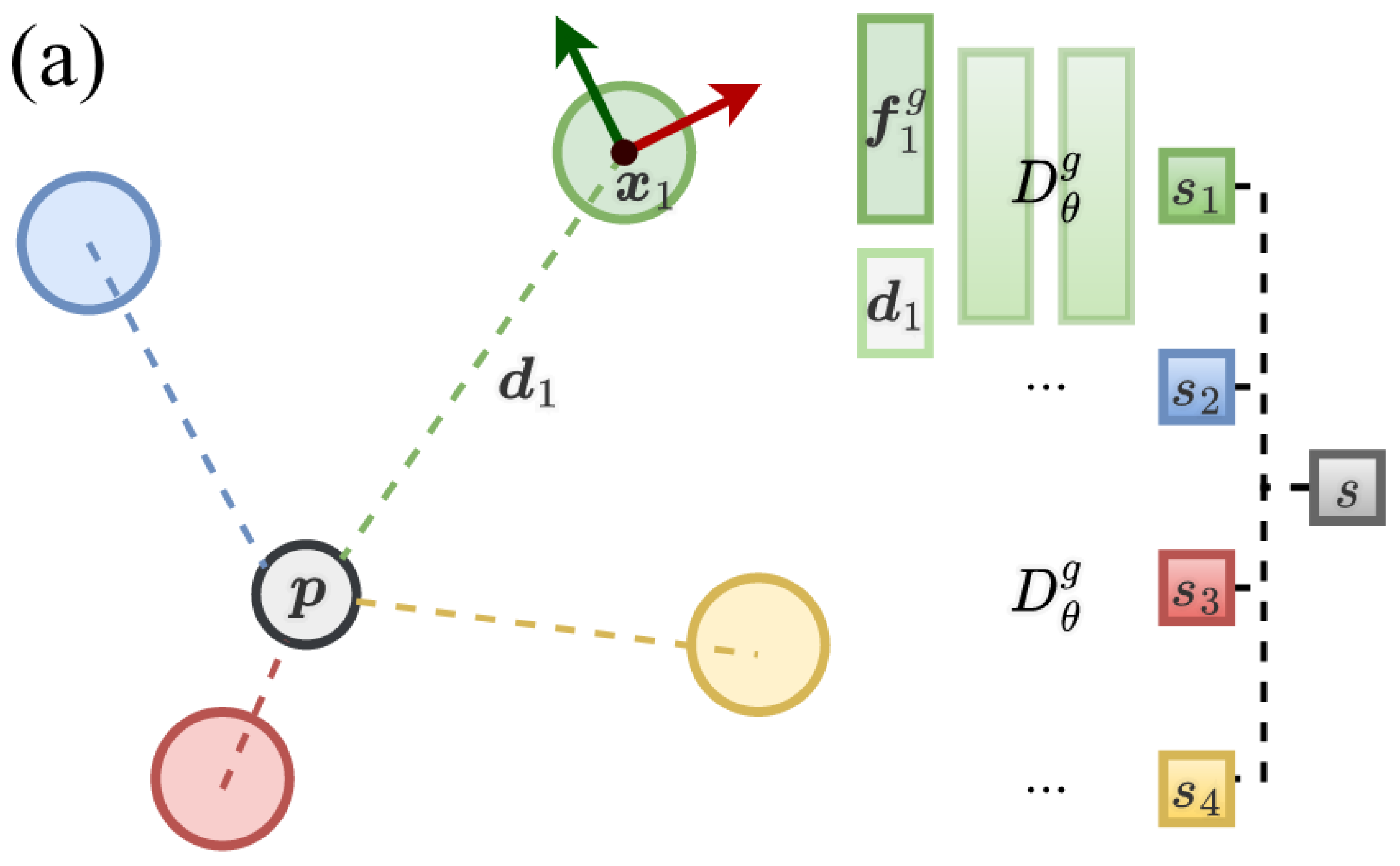


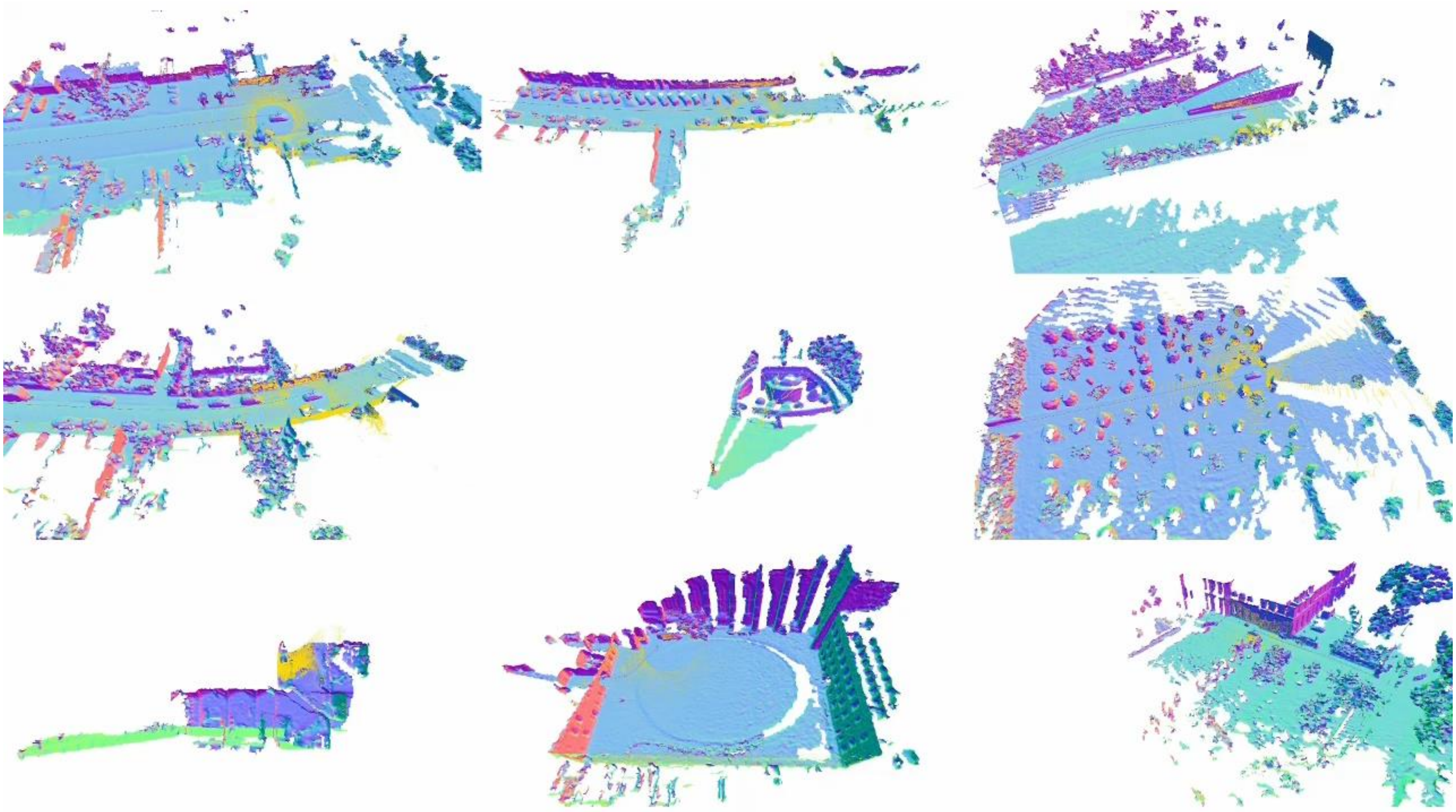
# PIN-SLAM: Neural SDF with Voxel Hashing

- The environment is covered by neural points, each with a learnable feature vector
- Given observed point $p$, find nearby neural points via voxel hashing
- Each neural point $x_i$ predicts a SDF value by passing its latent feature $f_i$ to a MLP decoder network
- Final SDF prediction obtained by weighted averaging



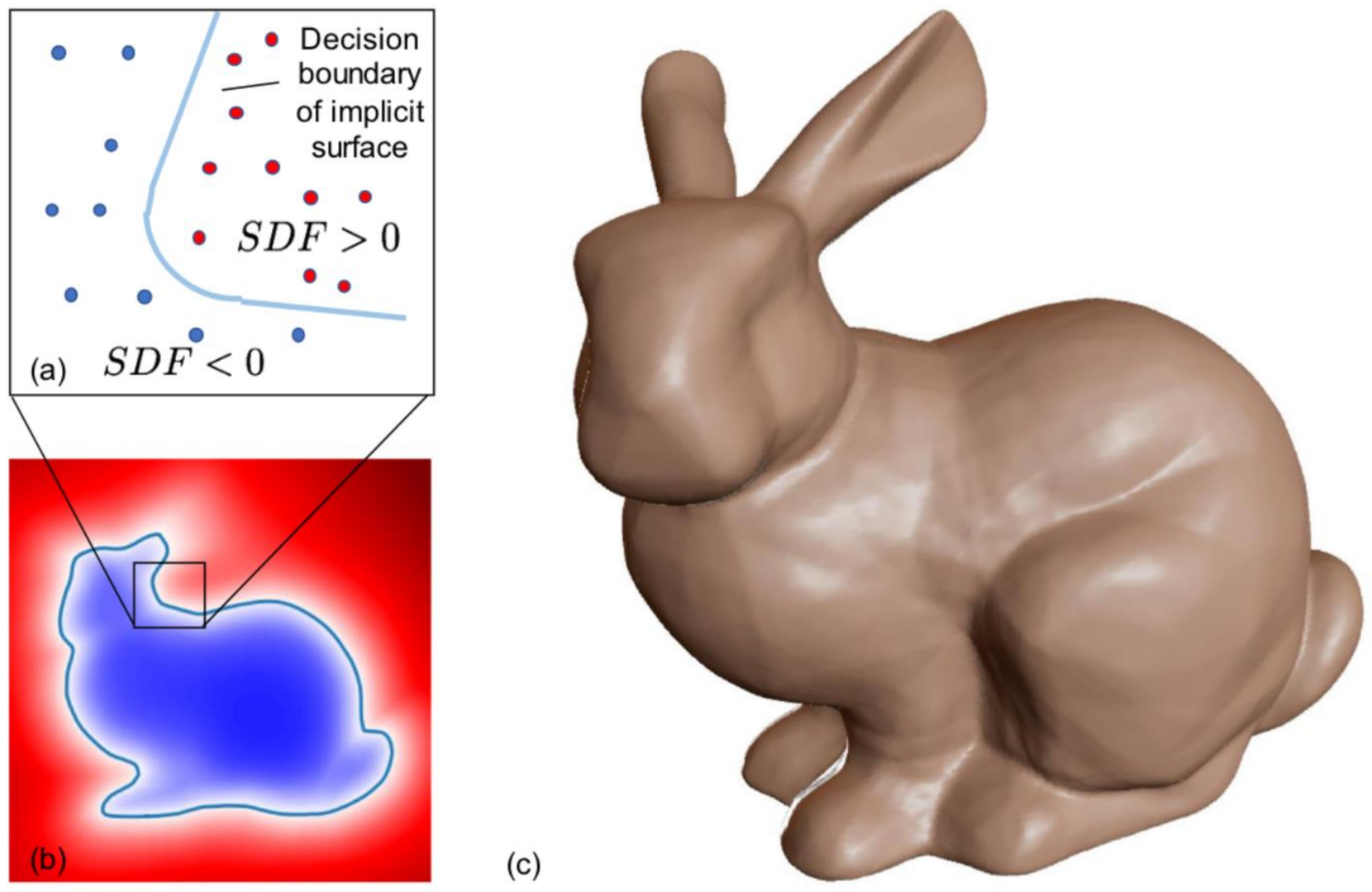

Pan et al., “PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency,” IEEE T-RO, 2024.

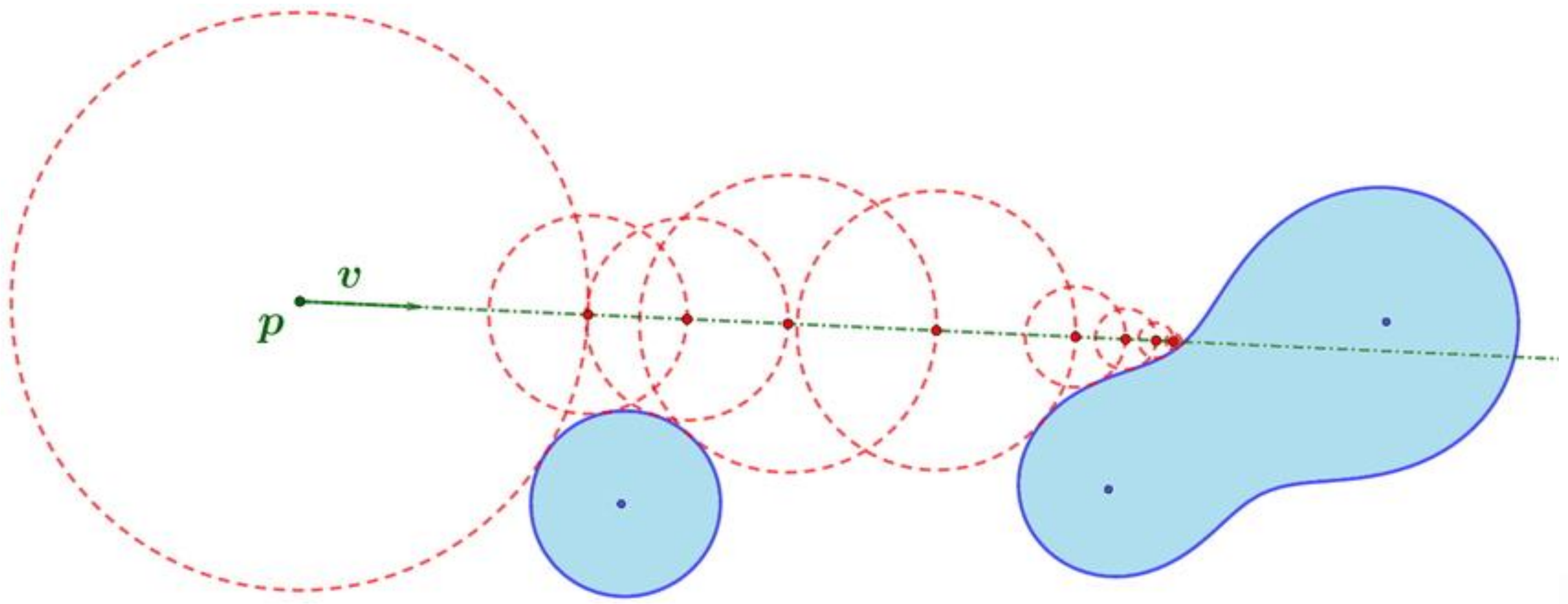
# PIN-SLAM: Large-Scale TSDF Mapping using Lidar



# Summary

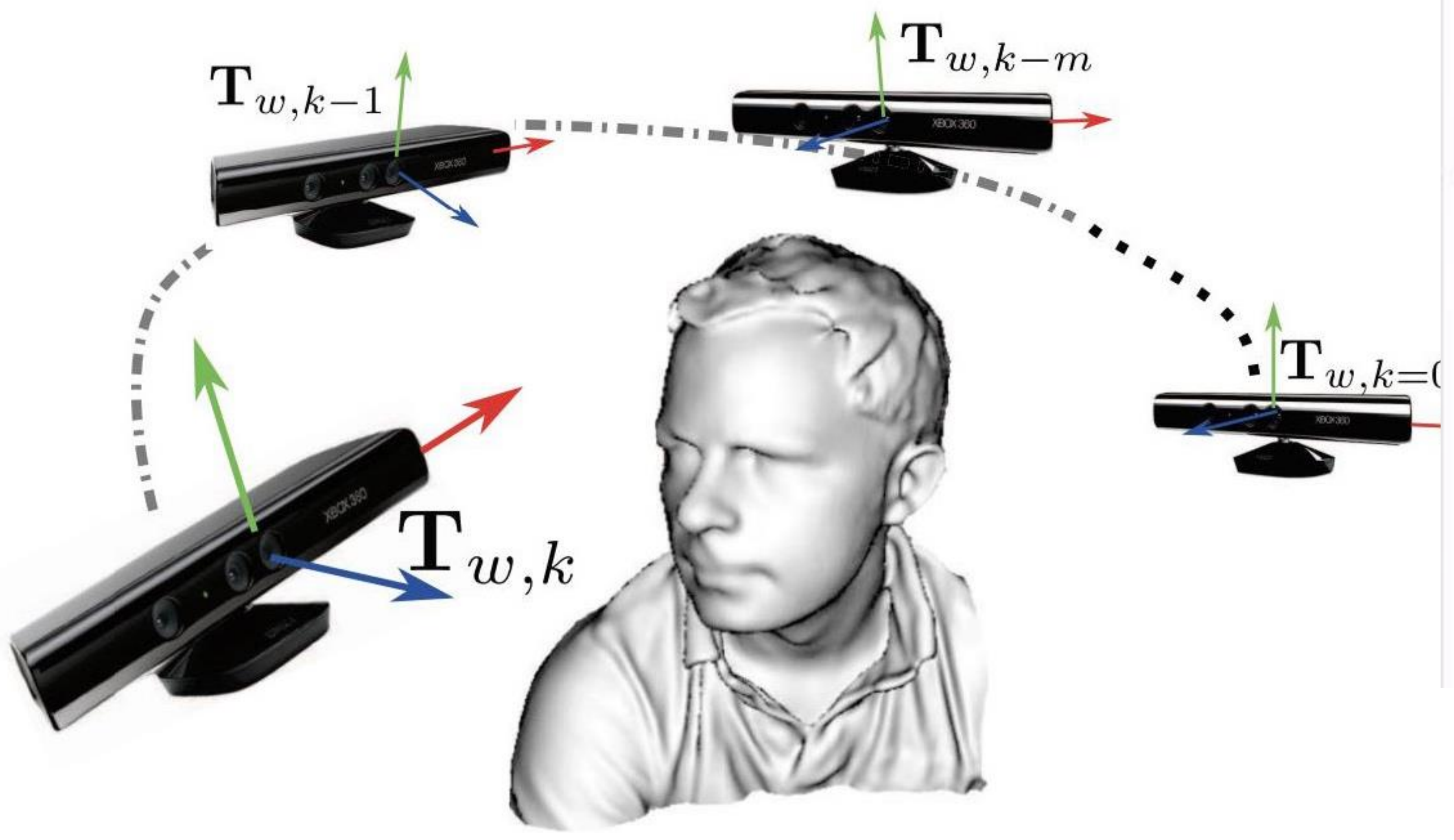
## SDF: definition and applications



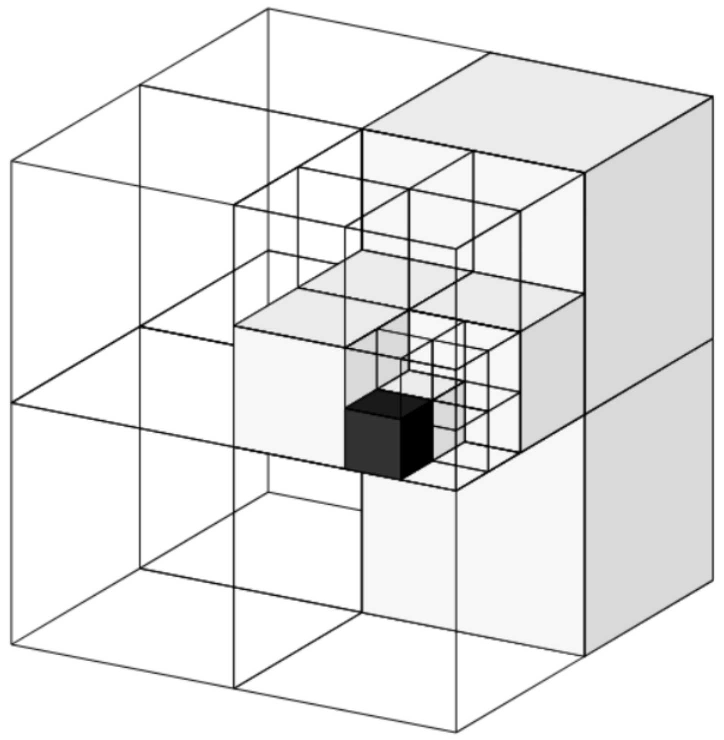



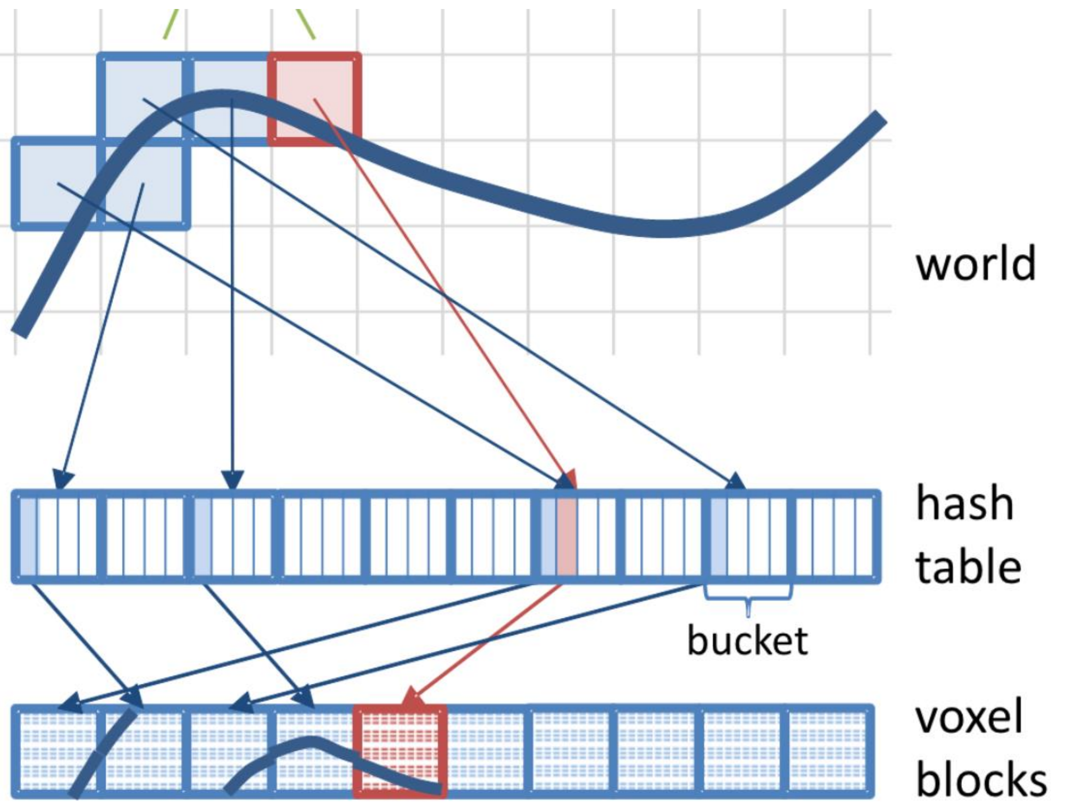

## Dense SLAM with TSDF (KinectFusion)

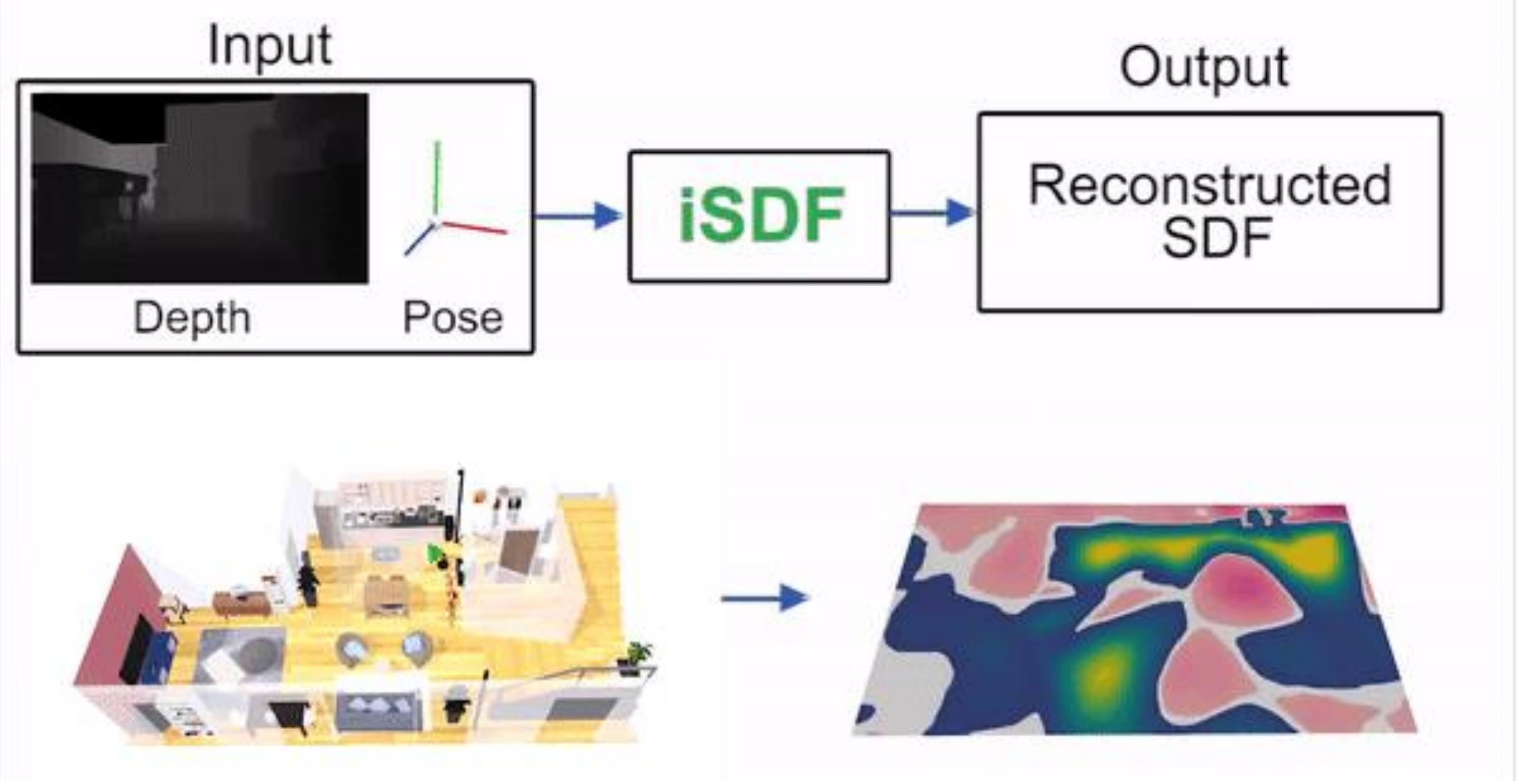
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -0.9 | -0.3 | 0.0 | 0.2 | 1 | 1 | 1 | 1 | 1 |
| -1 | -0.9 | -0.2 | 0.0 | 0.2 | 1 | 1 | 1 | 1 |
| -1 | -0.9 | -0.3 | 0.0 | 0.1 | 0.9 | 1 | 1 | 1 |
| -1 | -0.8 | -0.3 | 0.0 | 0.2 | 0.8 | 1 | 1 | 1 |
| -1 | -0.9 | -0.4 | -0.1 | 0.1 | 0.8 | 0.9 | 1 | 1 |
| -1 | -0.7 | -0.3 | 0.0 | 0.3 | 0.6 | 1 | 1 | 1 |
| -1 | -0.7 | -0.4 | 0.0 | 0.2 | 0.7 | 0.8 | 1 | 1 |
| -0.9 | -0.7 | -0.2 | 0.0 | 0.2 | 0.8 | 0.9 | 1 | 1 |
| -0.1 | 0.0 | 0.0 | 0.1 | 0.3 | 1 | 1 | 1 | 1 |
| 0.5 | 0.3 | 0.2 | 0.4 | 0.8 | 1 | 1 | 1 | 1 |




## Recent advancements: data structures and neural networks





# Questions?