3D World Representations

Maren Bennewitz, Rohit Menon

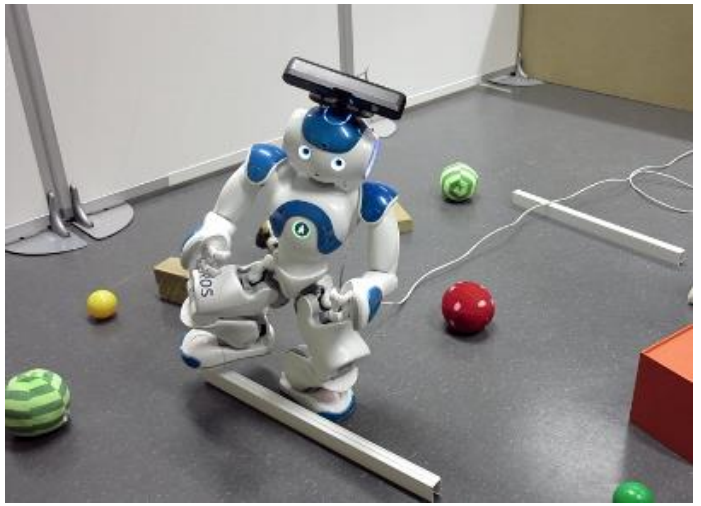
Humanoid Robots Lab, University of Bonn

Goal of This Chapter

- Overview of different 2.5/3D world representations with their strengths and weaknesses
- Understand which representation is useful for which application
- Know how to calculate transformations between point clouds

Robots in 3D Environments

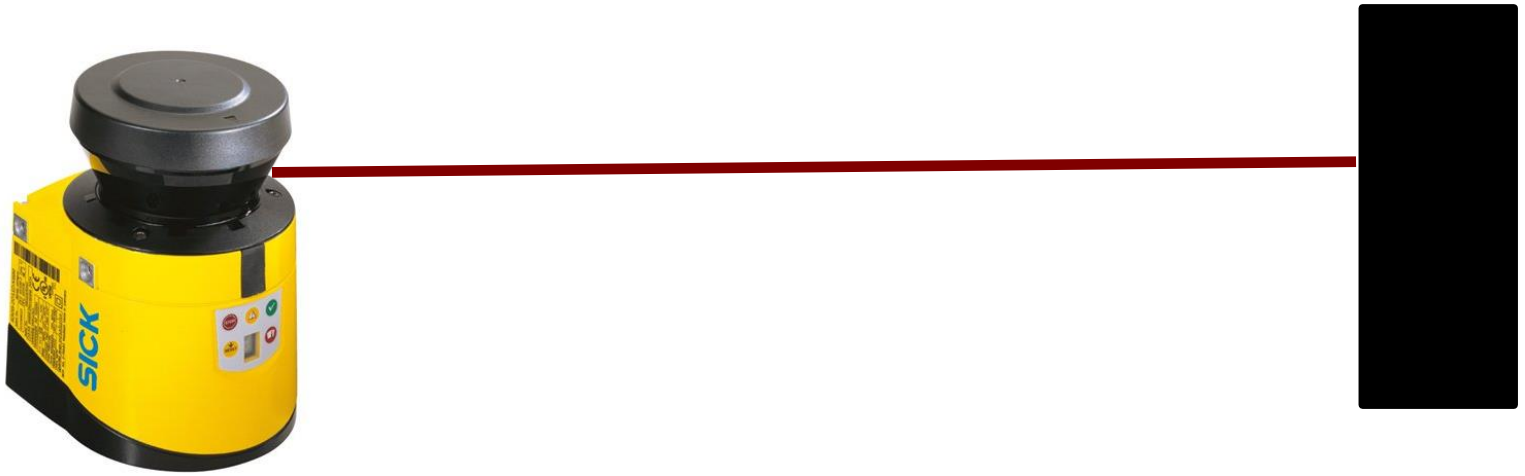
source: Honda



Motivation

- Robots live in the 3D world
- Collision avoidance, motion planning, and localization require accurate 3D world models
- Given: 3D point cloud data from known robot and sensor poses
- Question: How to represent the 3D structure of the environment?

Laser Scanning Principle



Range scanners measure the distance
to the closest obstacle

Image courtesy: Sick

Example: Data Acquisition

Popular Representations of the 3D World

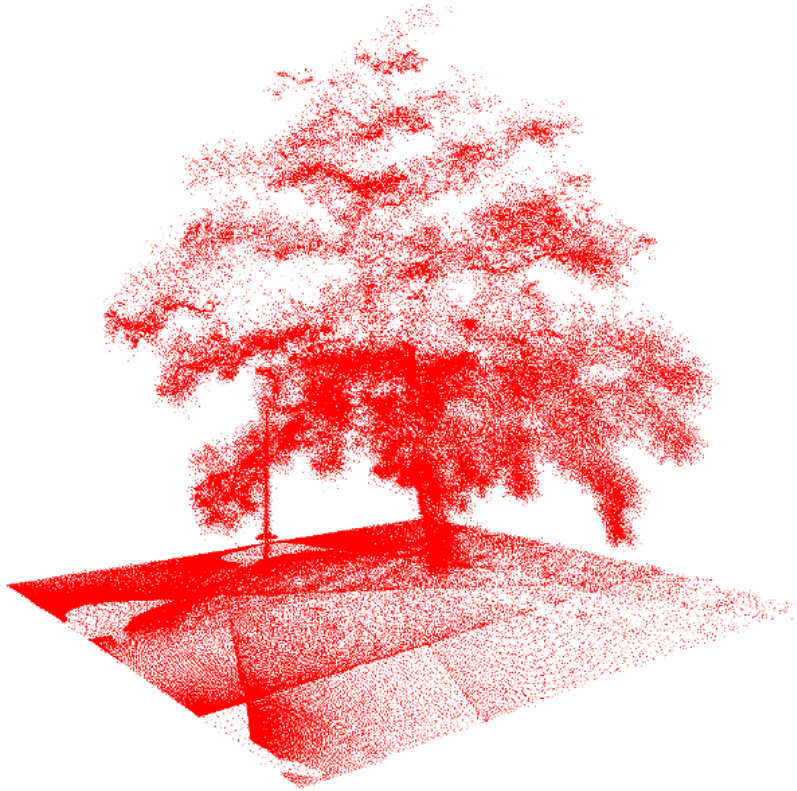
- Point clouds
- Voxel grids
- Height maps
- Surface maps
- Meshes
- Distance fields
- …

All models are wrong, but some are useful

- George Box

Point Clouds

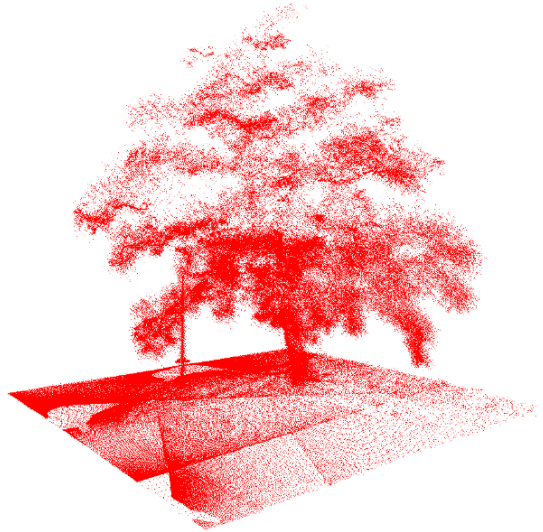
- Set of 3D data points in world frame
- Obtained, e.g., by a laser scanner or depth camera



Point Clouds

Pros

- No discretization of data
- Mapped area not limited



Cons

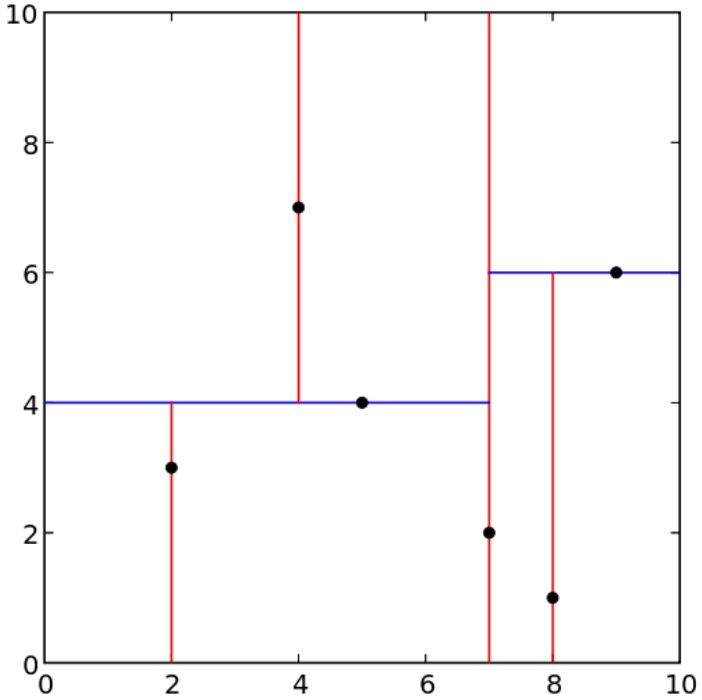
- Unbounded memory usage
- No constant time access for location queries
- No distinction between free or unknown space

Point Clouds and Efficient Location Queries

- Naïve implementation (list, array) has a linear complexity for location queries
- More effective solutions through kd-trees
- **kd-trees** operate in k-dimensions
- Space-partitioning data structure for organizing k-dimensional points
- Search/insert/delete in **logarithmic** time on average

Example: kd-Tree (2-dim.)

Binary space partitioning



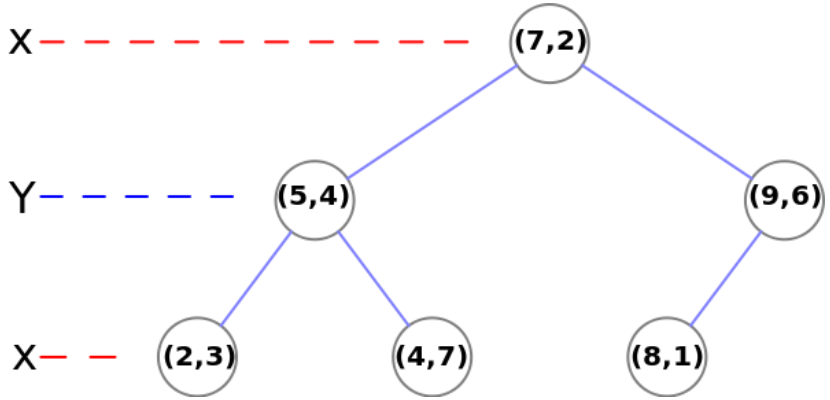
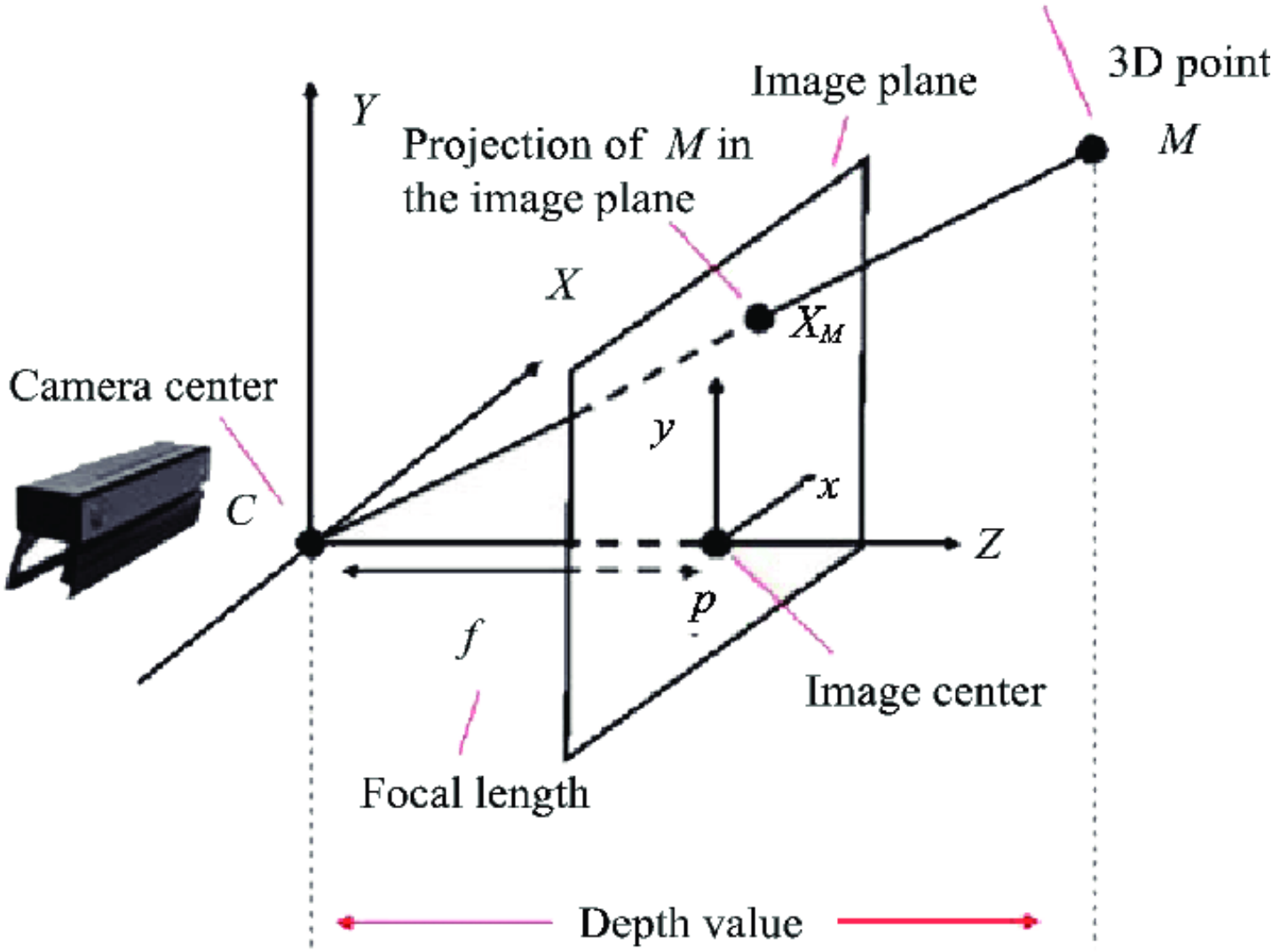


Image courtesy: Wikipedia

From Depth Maps to Point Clouds



[Xu, Hantong & Xu, Jiamin & Xu, Weiwei. (2019). Survey of 3D modeling using depth cameras.]

From Depth Maps to Point Clouds

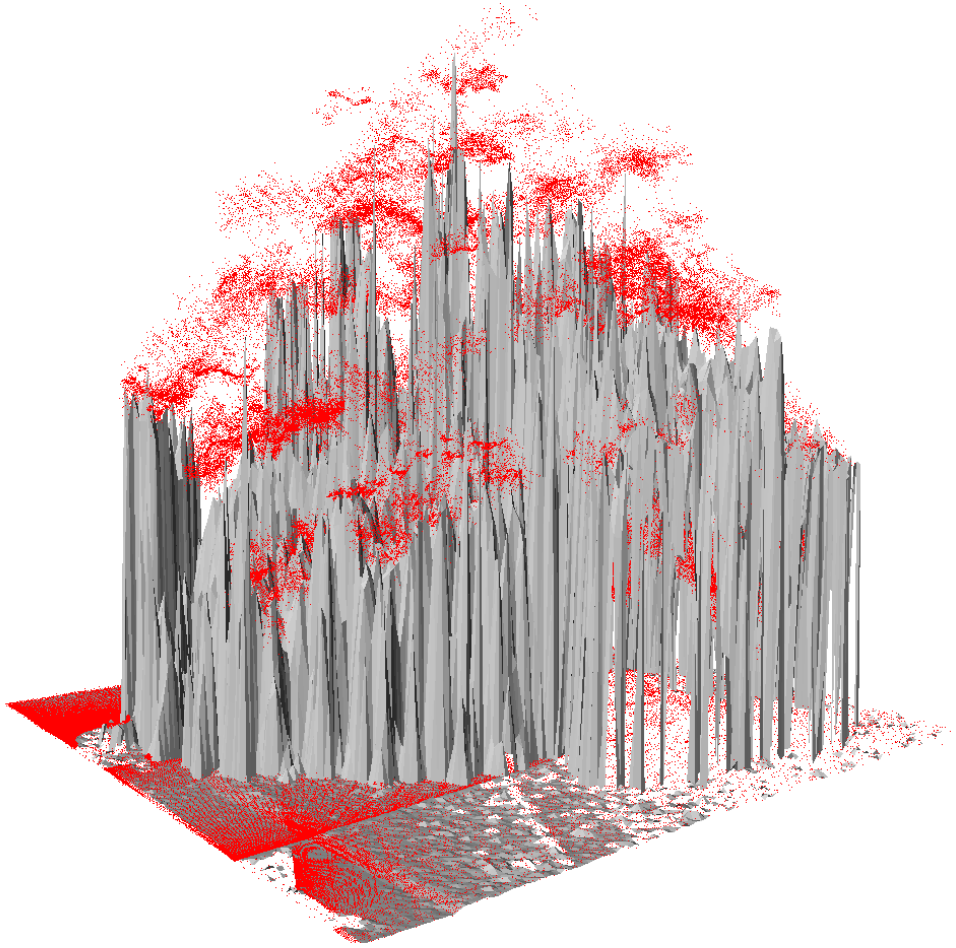
- Each depth map point x_m with coordinates (u, v) represents the distance z from the focal point to the world point M along the principal axis
- $z = D(u, v), \quad x = z \cdot \frac{u - c_x}{f_x}, y = z \cdot \frac{v - c_y}{f_y}, \; M = [x, y, z]$
- $C \rightarrow$ Camera Center
- $P \rightarrow$ Image Center or Principal Point i.e. Coordinates of P in Image Plane = (c_x, c_y)
- $f_x, f_y \rightarrow$ Focal Length along x and y axes. Typically similar if pixels are scaled identically along both axes

How to Align Depth and RGB Images?

- Example scenario: RGB 1920×1080, Depth 640×480, unsynced sensors.
- Undistort both images with respective parameters
- Convert depth (640×480) to 3D points
- Apply extrinsic R,T to 3D points
- Project into 1920×1080 RGB image plane
- Interpolate depth to RGB image using nearest values
- Different point types: PointXYZ, PointXYZI, PointXYZRGB, etc.

2.5D Maps: Height Maps

Average over all points that fall into a 2D cell and consider this as the height value



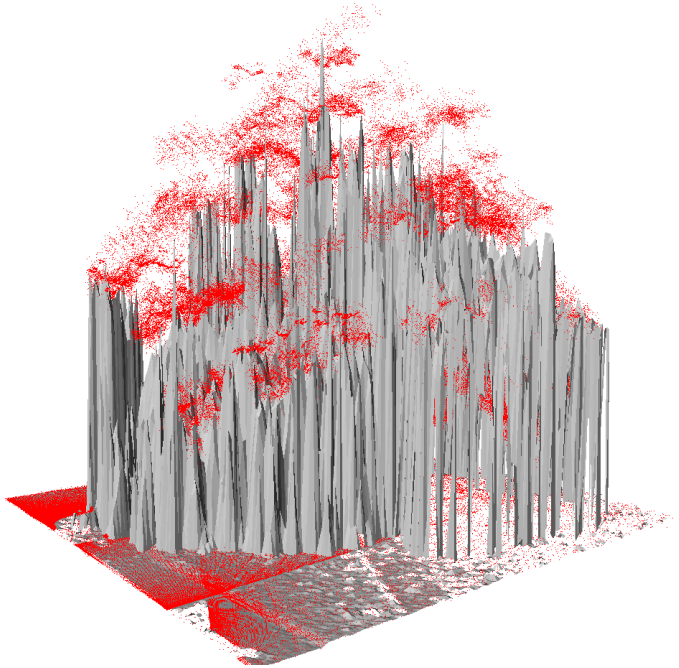
2.5D Maps: Height Maps

Pros

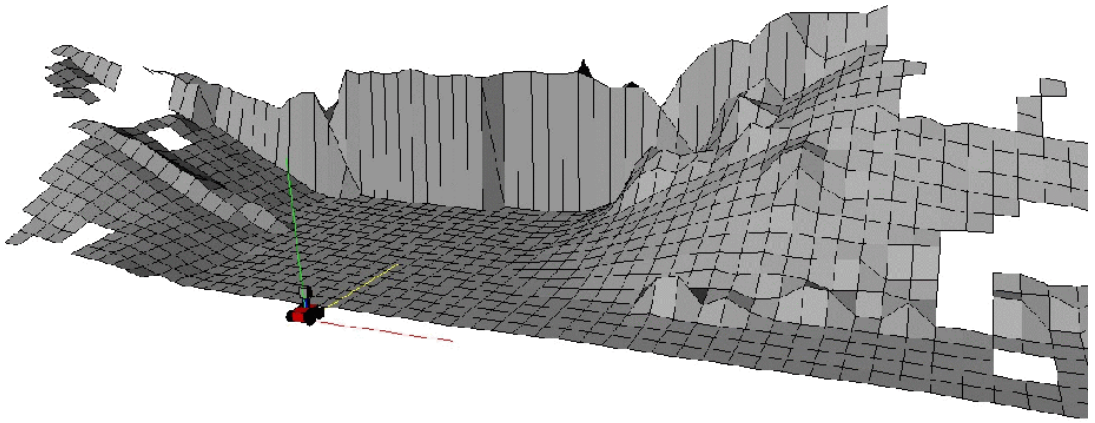
- Memory efficient (2D)
- Constant time access

Cons

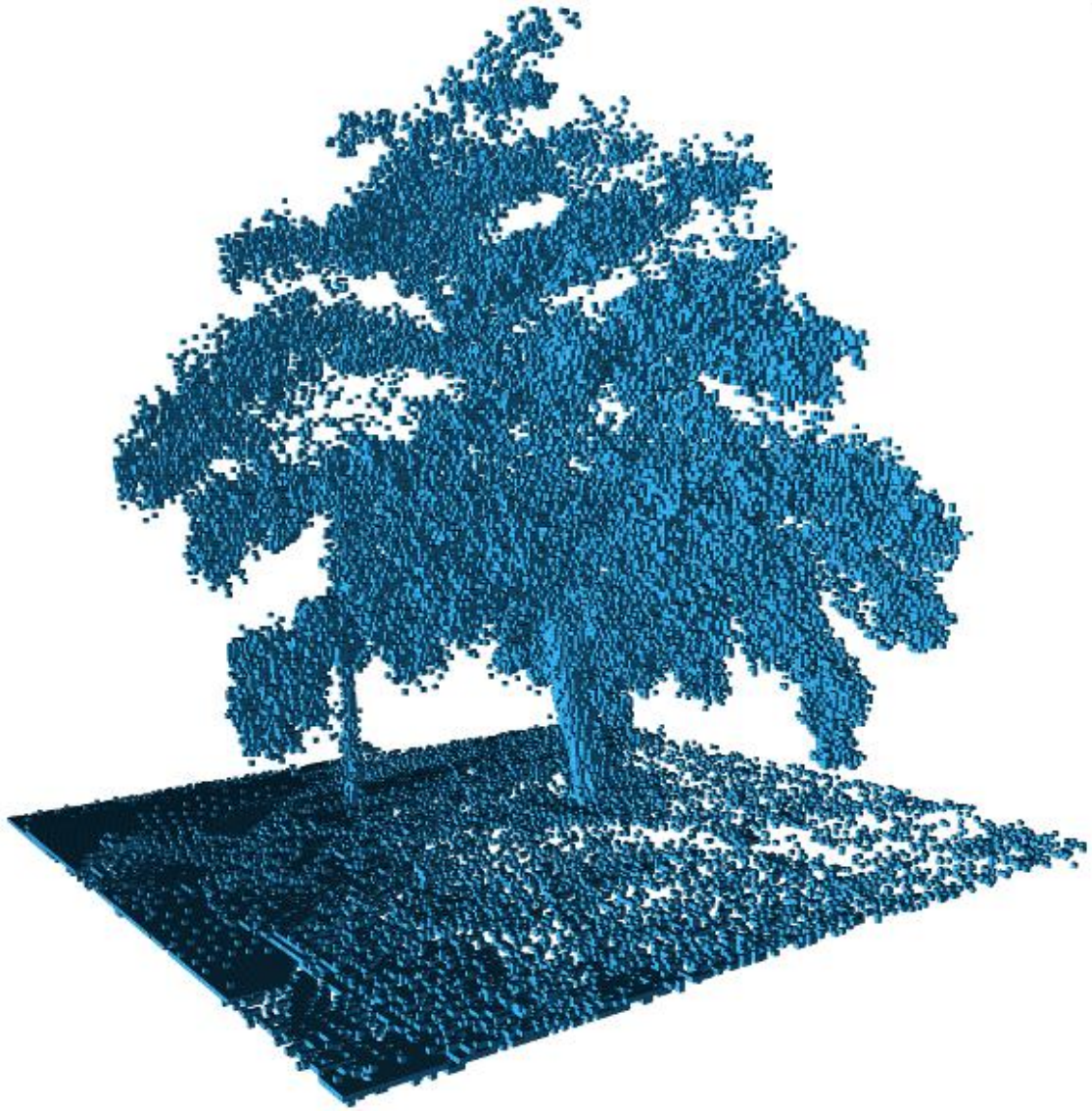
- No vertical objects
- Only one level is represented



Example: Problem of Height Maps



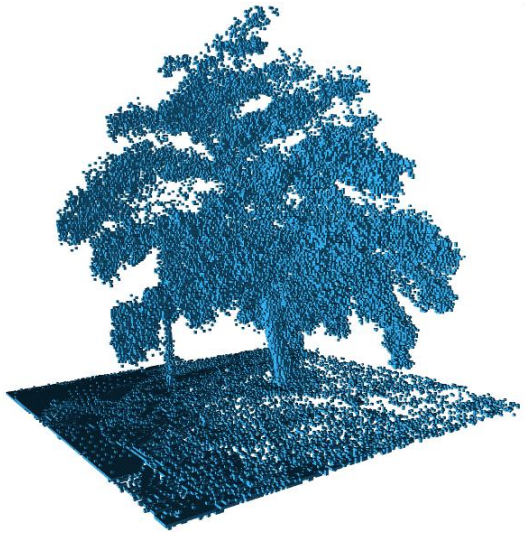
3D Voxel Grids



3D Voxel Grids

Pros

- Volumetric representation
- Constant access time
- Probabilistic update possible

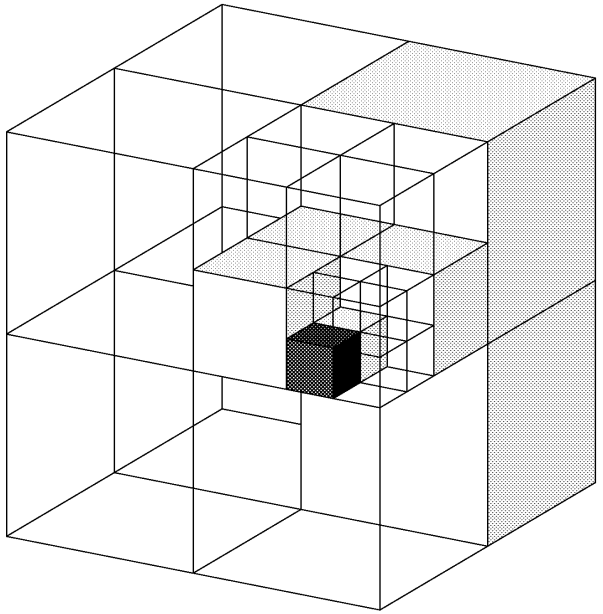


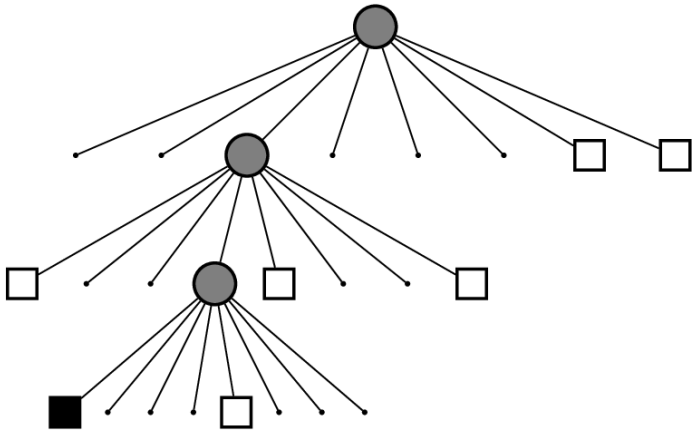
Cons

- Memory requirement: Complete grid is allocated in memory
- Extent of the map has to be known/guessed
- Discretization errors

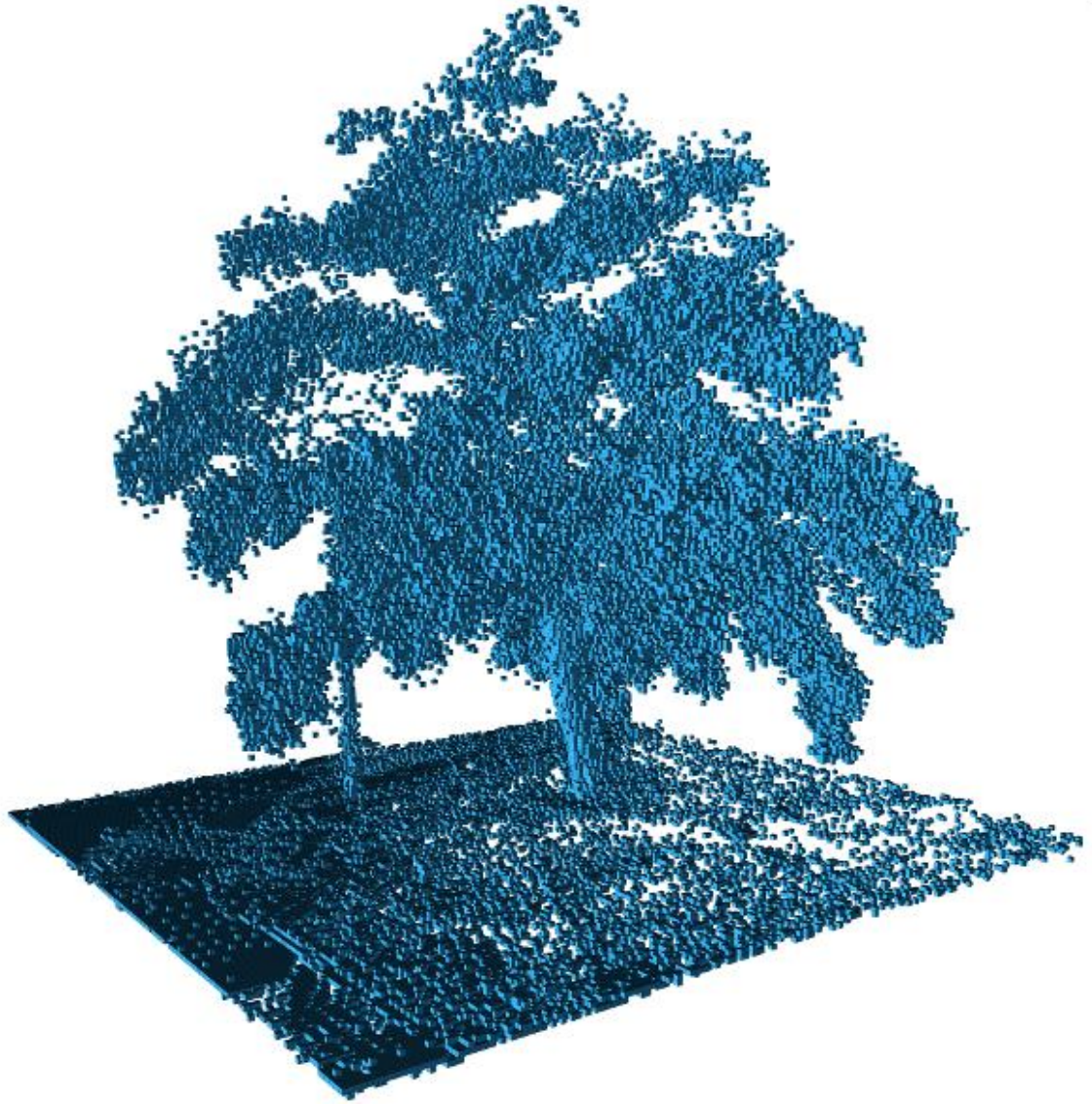
Octree-Based Representation

- Tree-based data structure
- Recursive subdivision of the space into octants
- Volumes allocated as needed
- “Smart” 3D grid

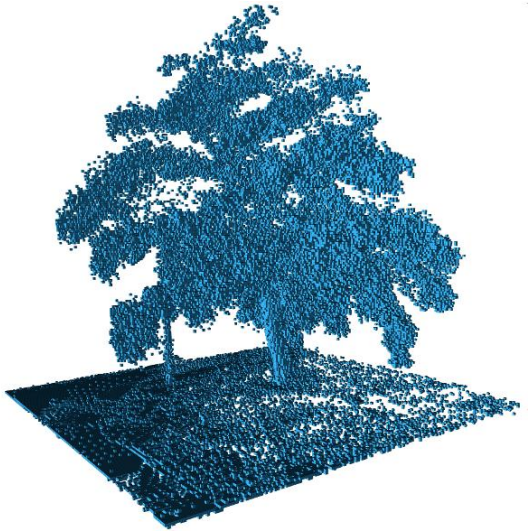




Octrees



Octrees



Pros

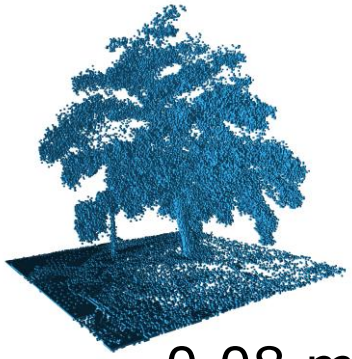
- Full 3D model
- Inherently multi-resolution
- Memory-efficient, volumes only allocated as needed
- Probabilistic update possible

Cons

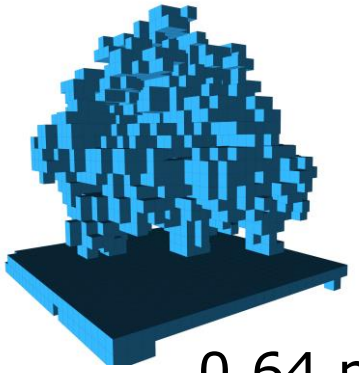
- Efficient implementation can be tricky (memory allocation, update, map files, …)

Multi-Resolution Queries

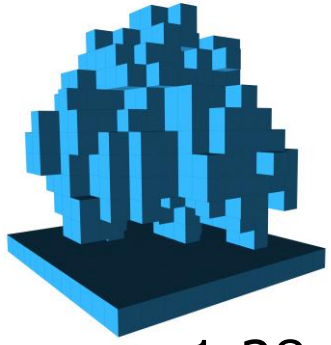
$$P(m_i) = \max_{j=1\ldots8} P(m_{i_j}) \text{ with } m_{i_j} \in \text{children}(m_i)$$



0.08 m



0.64 m



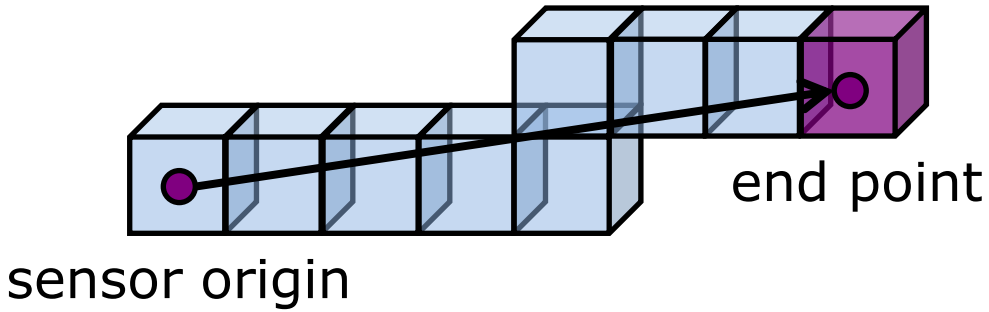
1.28 m

OctoMap Framework

- Based on octrees
- Probabilistic, volumetric representation of occupancy including unknown
- Supports multi-resolution map queries
- Memory efficient
- Generates compact map files (maximum likelihood map as bit stream)
- Open source implementation as C++ library available at http://octomap.github.io/

Ray Casting for Map Updates

- Ray casting from sensor origin to end point in the map along the beam
- Mark last voxel as occupied, all other voxels on ray as free
- Measurements are integrated probabilistically given the robot's pose (recursive binary Bayes' filter)



[Lecture on Cognitive Robotics]

Probabilistic Map Update

- Occupancy probability modeled as recursive binary Bayes' filter

$$p(m_i \mid z_{1:t}, x_{1:t})$$

$$= \left[1 + \frac{1 - p(m_i \mid z_t, x_t)}{p(m_i \mid z_t, x_t)} \frac{1 - p(m_i \mid z_{1:t-1}, x_{1:t-1})}{p(m_i \mid z_{1:t-1}, x_{1:t-1})} \frac{p(m_i)}{1 - p(m_i)}\right]^{-1}$$

- Efficient update using log-odds notation

$$l(m_i \mid z_{1:t}, x_{1:t})$$

$$= \underbrace{l(m_i \mid z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i \mid z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} - \underbrace{l(m_i)}_{\text{prior}}$$

[Lecture on Cognitive Robotics]

Video: Large Outdoor Area

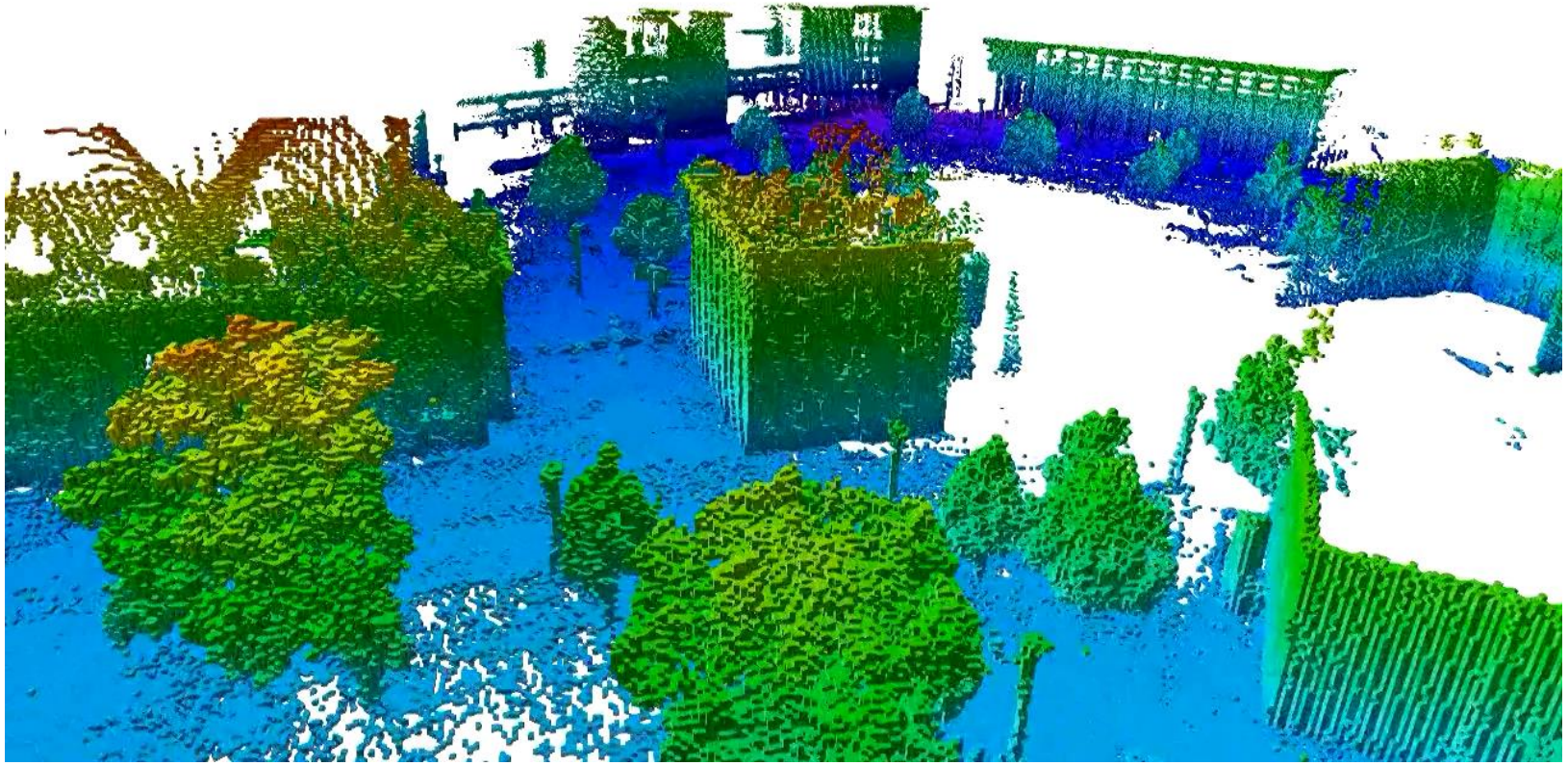
Freiburg computer science campus

(292 x 167 x 28 m³, 20 cm resolution)

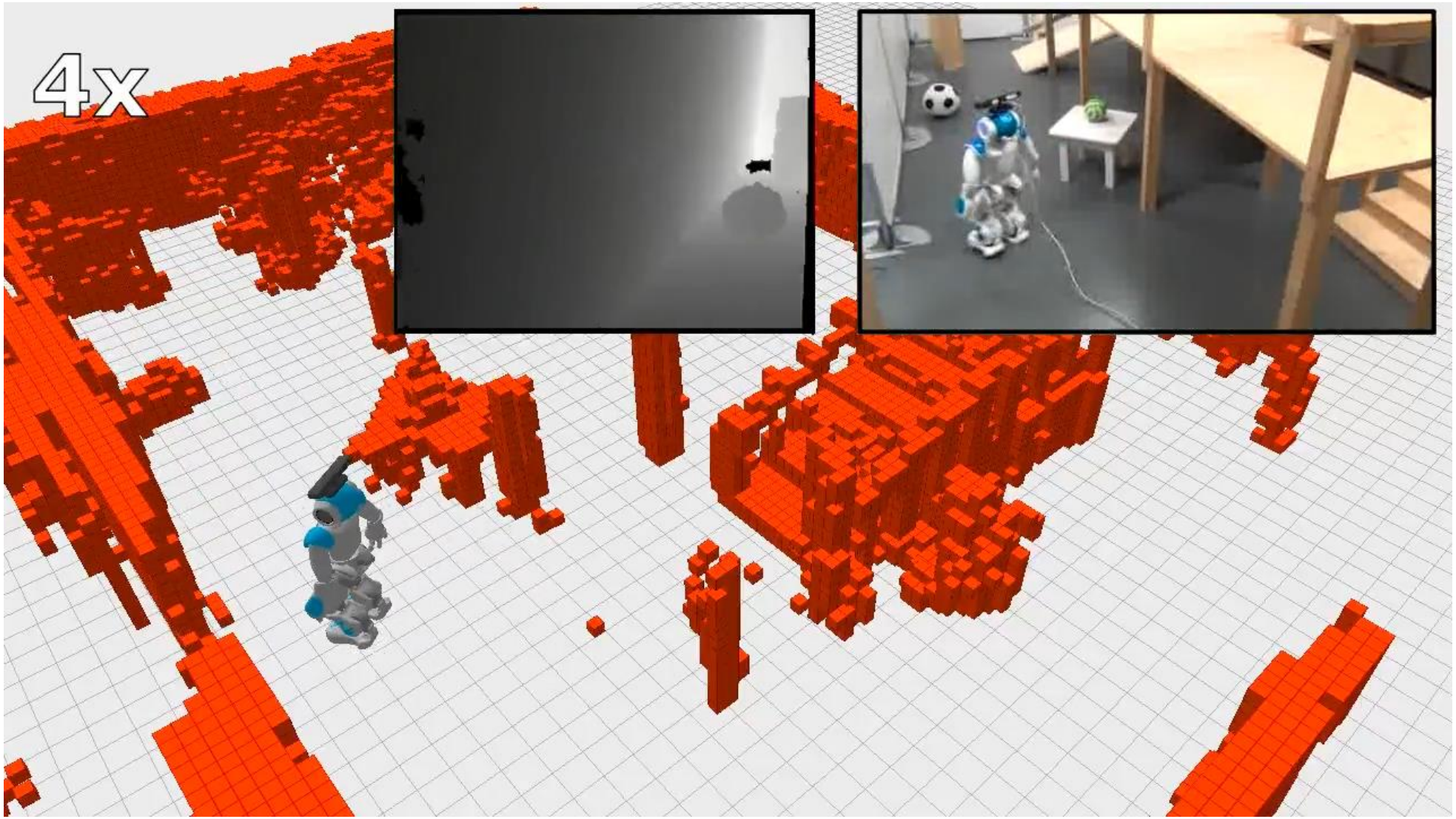
Octree in memory:
130 MB

3D Grid:
649 MB

Octree file: 2 MB
(bit stream)



Online Mapping With Octomap



[D. Maier, A. Hornung and M. Bennewitz, Humanoids 2012]

Signed Distance Function

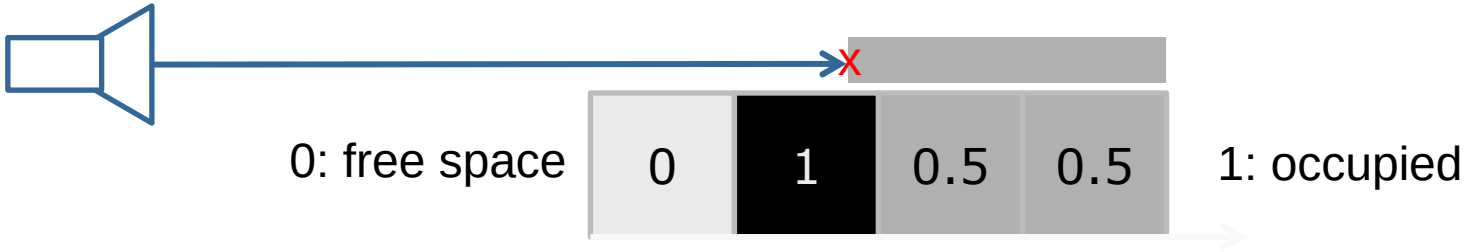
Signed Distance Function (SDF)

Key idea:

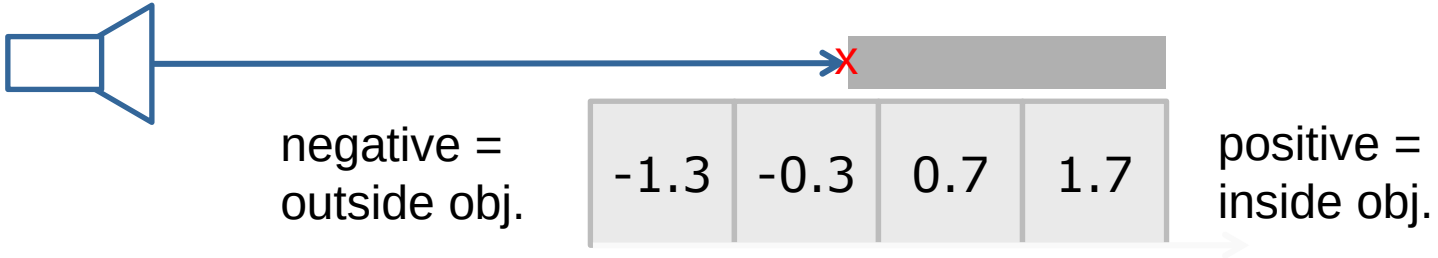
- Instead of representing occupancy values, represent the **distance of each cell** to the **nearest measured surface**
- The surface can be extracted afterwards at **sub-voxel** accuracy

Signed Distance Function (SDF)

- Grid maps: explicit representation

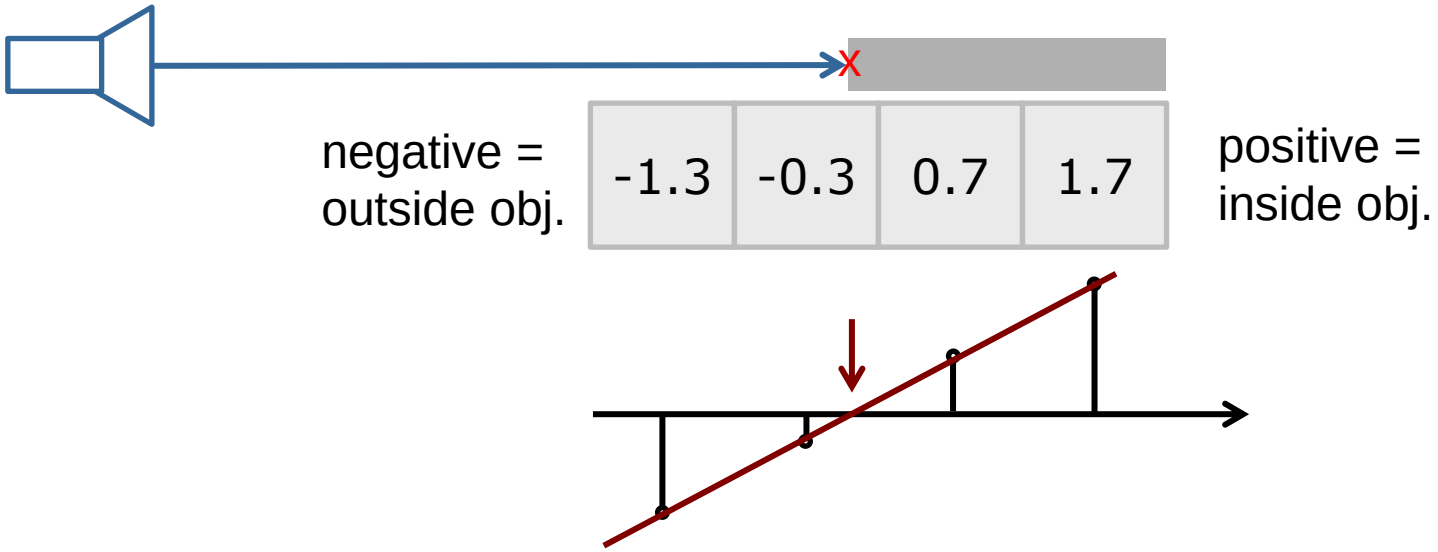


- SDF: implicit representation



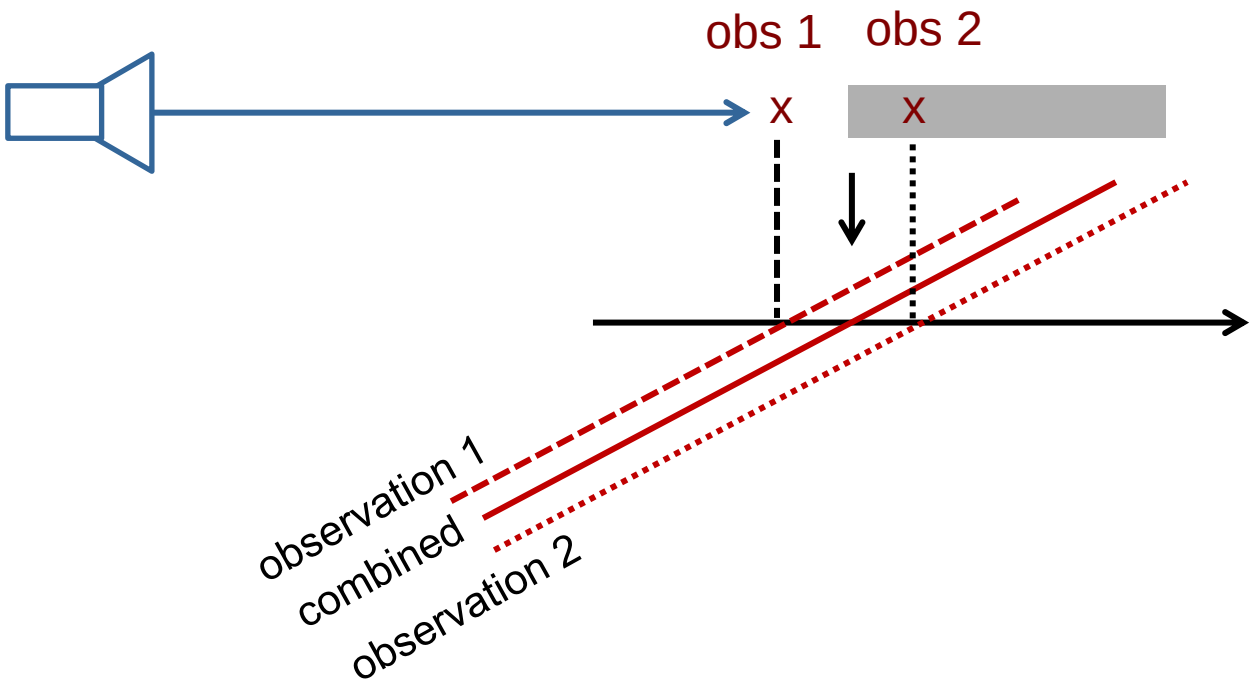
SDF Approach

1. Compute the signed distance values
2. Extract the surface using interpolation
3. The surface is located at the zero-crossing



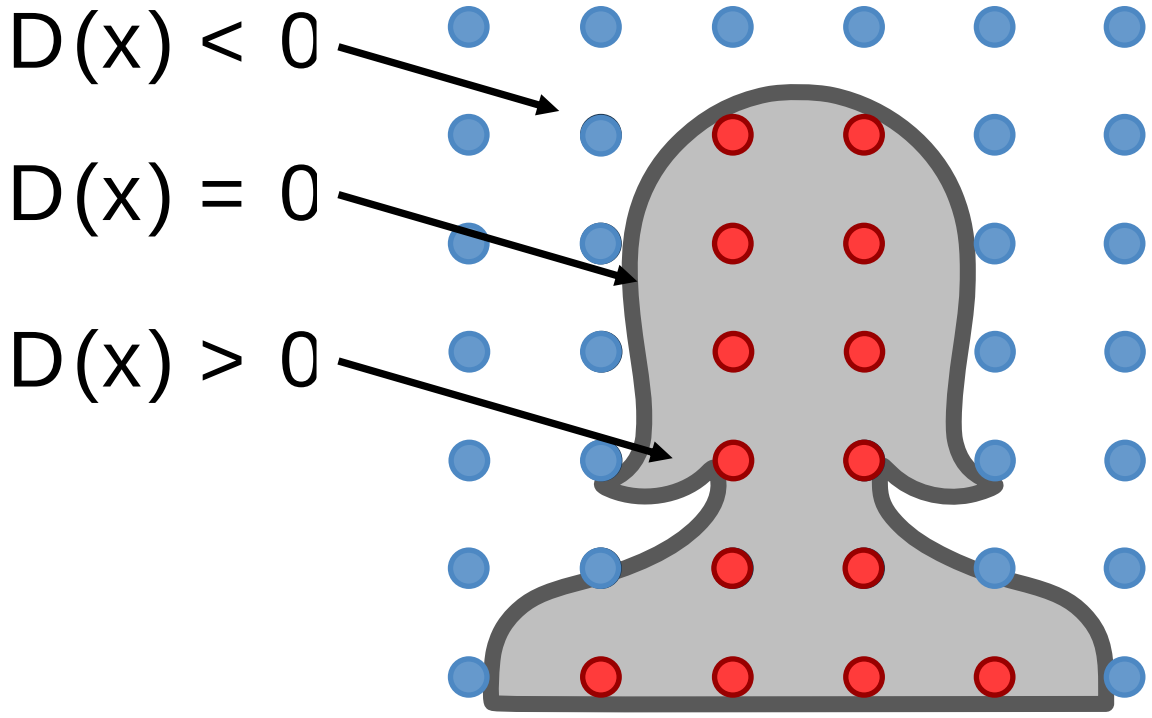
Properties

- Noise cancels out over multiple measurements



- Zero-crossing can be extracted at sub-voxel accuracy

Voxel Grid to Store SDF in 3D



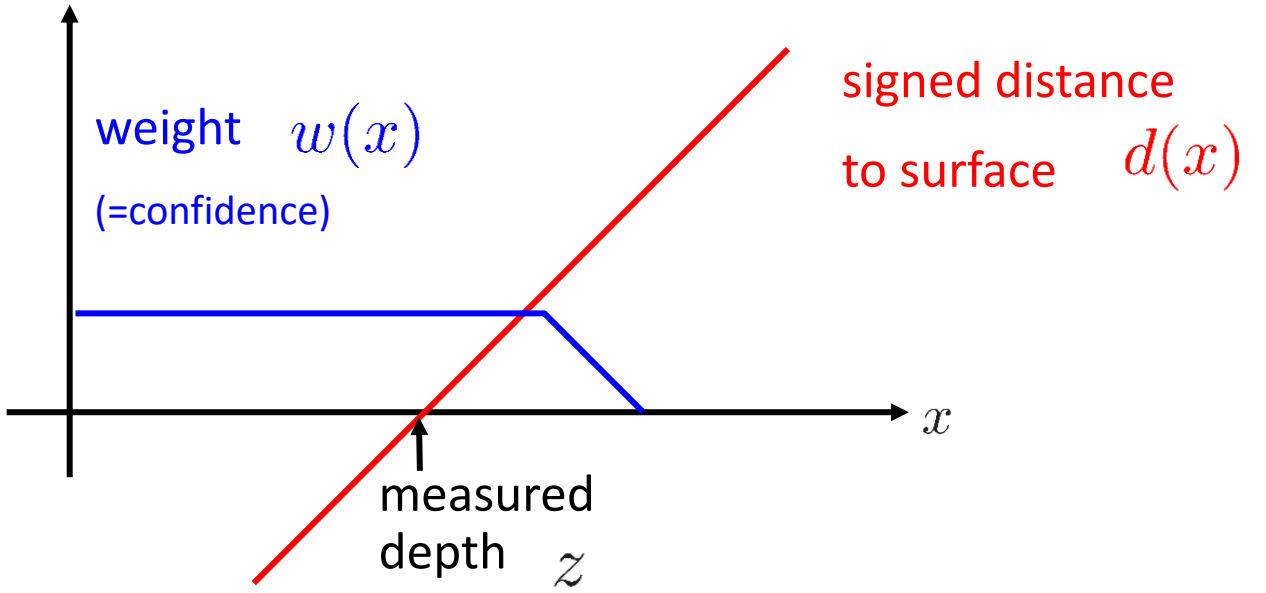
— Negative signed distance (=outside)

— Positive signed distance (=inside)

in general, there are several measurements for the voxels

Weighting Function for Multiple Measurements

- For each **voxel along the beam**, weigh the observation according to its **confidence**

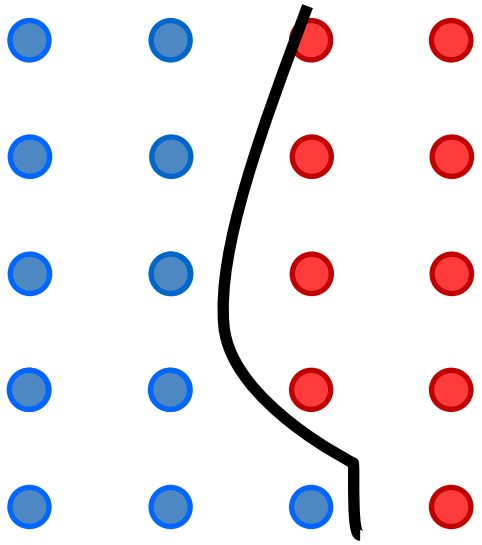


- Small weights ensure that values can be updated when new observations are available

SDF

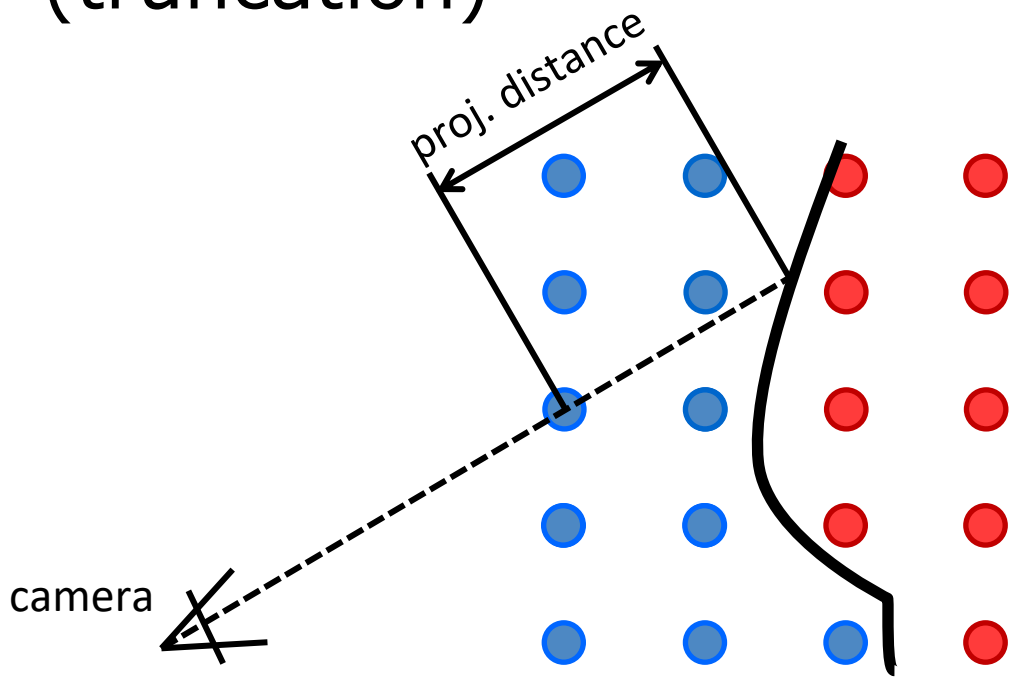
For each voxel along the beam, store

- Distance to the next surface D
- Weight W



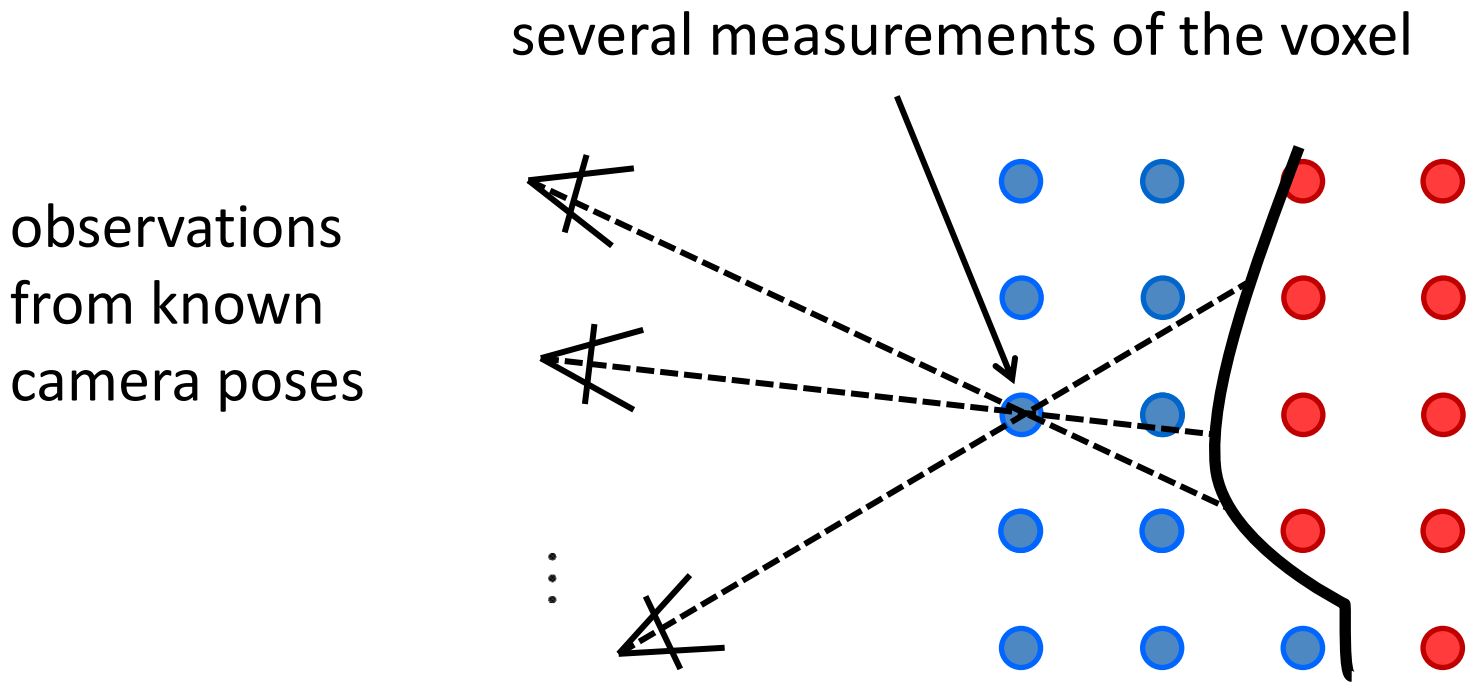
Truncated SDF

- Compute the SDF from a depth image
- Compute the **distance** of the voxels to the observed surface **along the beam**
- Update only a **small region around the endpoint** for efficiency (truncation)



Weighted Update

For each voxel, calculate the weighted average over all its measurements



Weighted Average

- For each voxel, store two values
  - Weighted sum of signed distances D
  - Sum of all weights W
- When new data arrive, update the values of each voxel according to

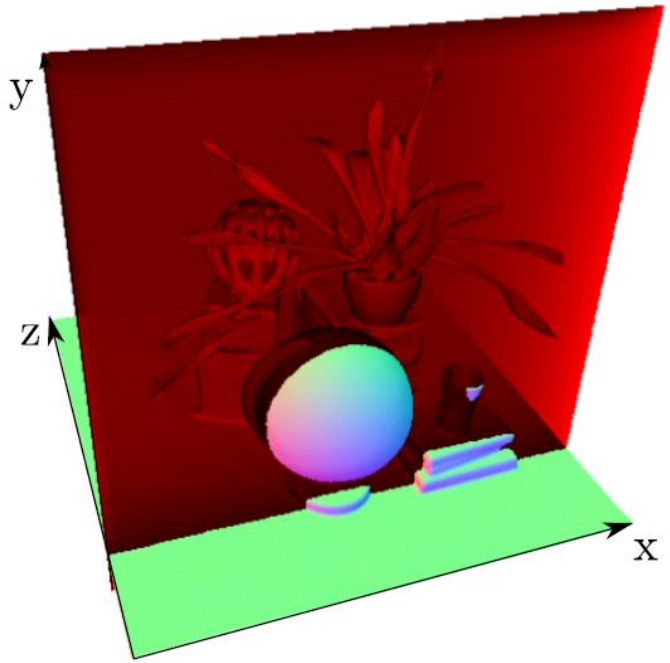
$$D \leftarrow \frac{W D + w_t d_t}{W + w_t}$$

$$W \leftarrow W + w_t$$

incremental computation of the weighted mean

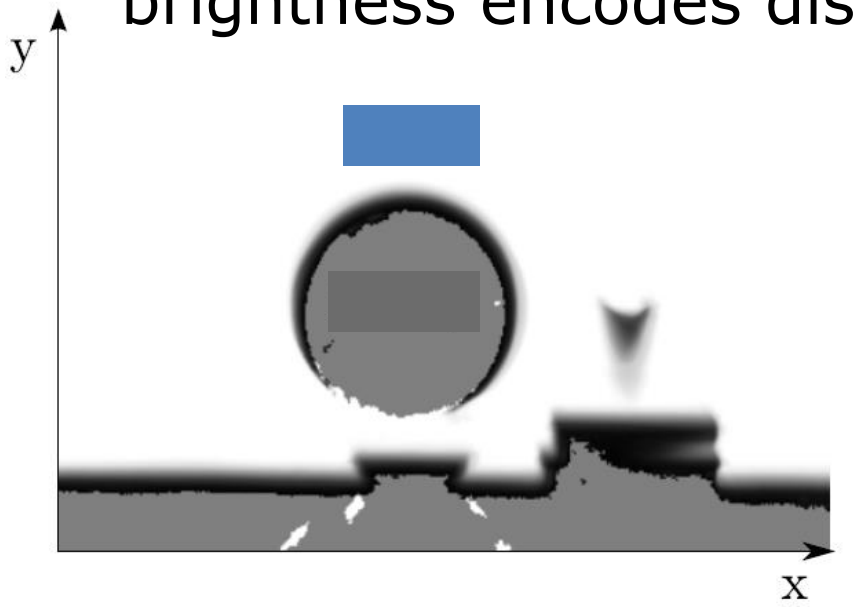
SDF Example

A cross section through a 3D signed distance function of a real scene



Surface with cross-section

brightness encodes distance



SDF

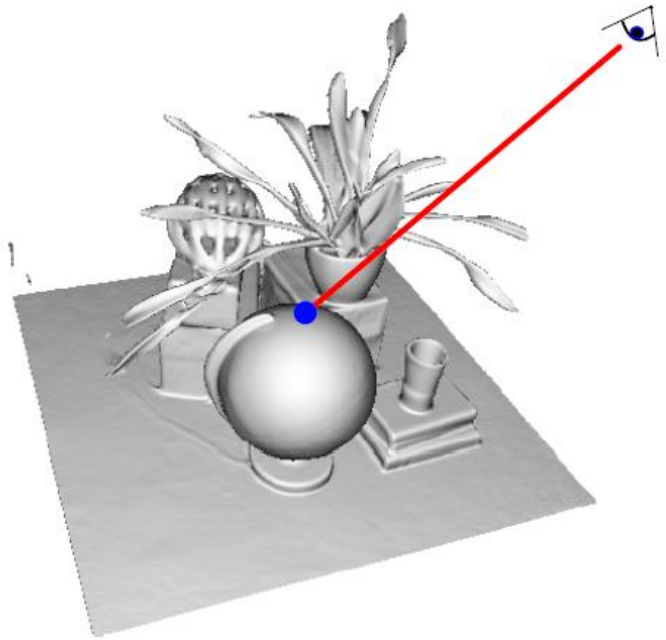
Surface Rendering

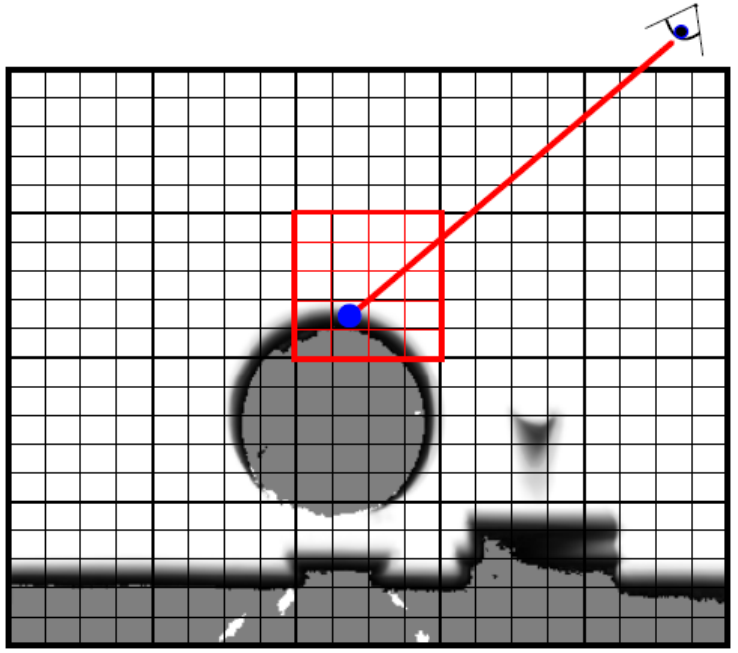
1. Ray casting (GPU, fast)
   For each camera pixel, shoot a ray and search for the zero crossing

2. Polygonization (CPU, slow)
   Use the marching cubes algorithm to generate a triangle mesh

Ray Casting

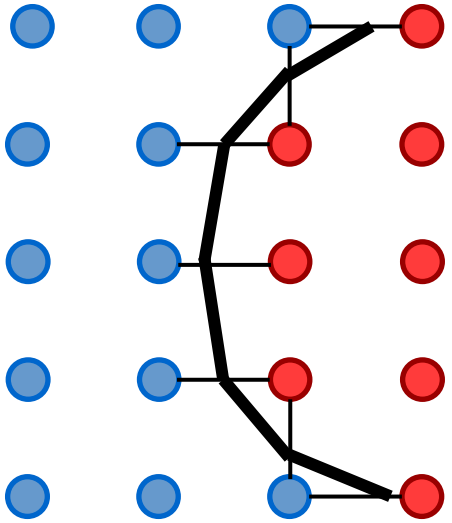
- For each pixel, shoot a ray and search for the first zero crossing in the SDF
- Value in the SDF can be used to skip along the ray when far from surface



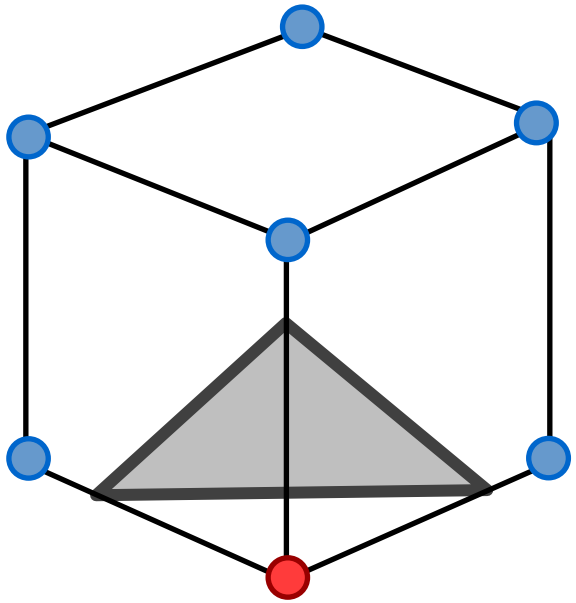


Mesh Extraction Using Marching Cubes

- Process the whole grid
- Find zero-crossings in the signed distance function by interpolation



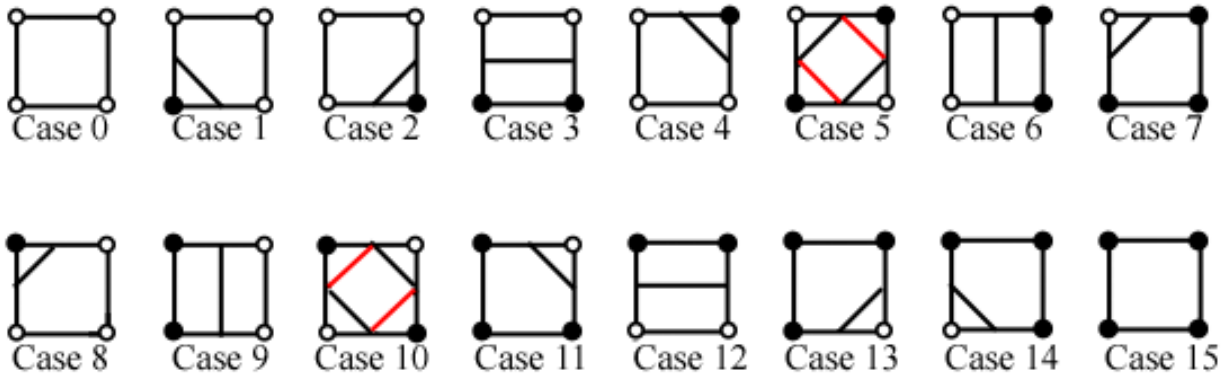
2D

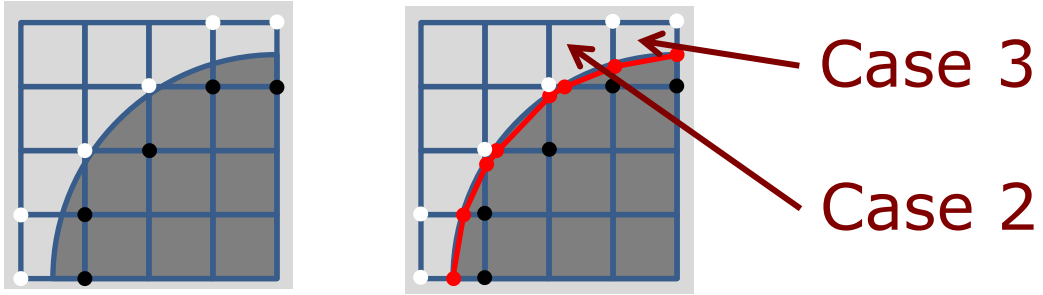


3D

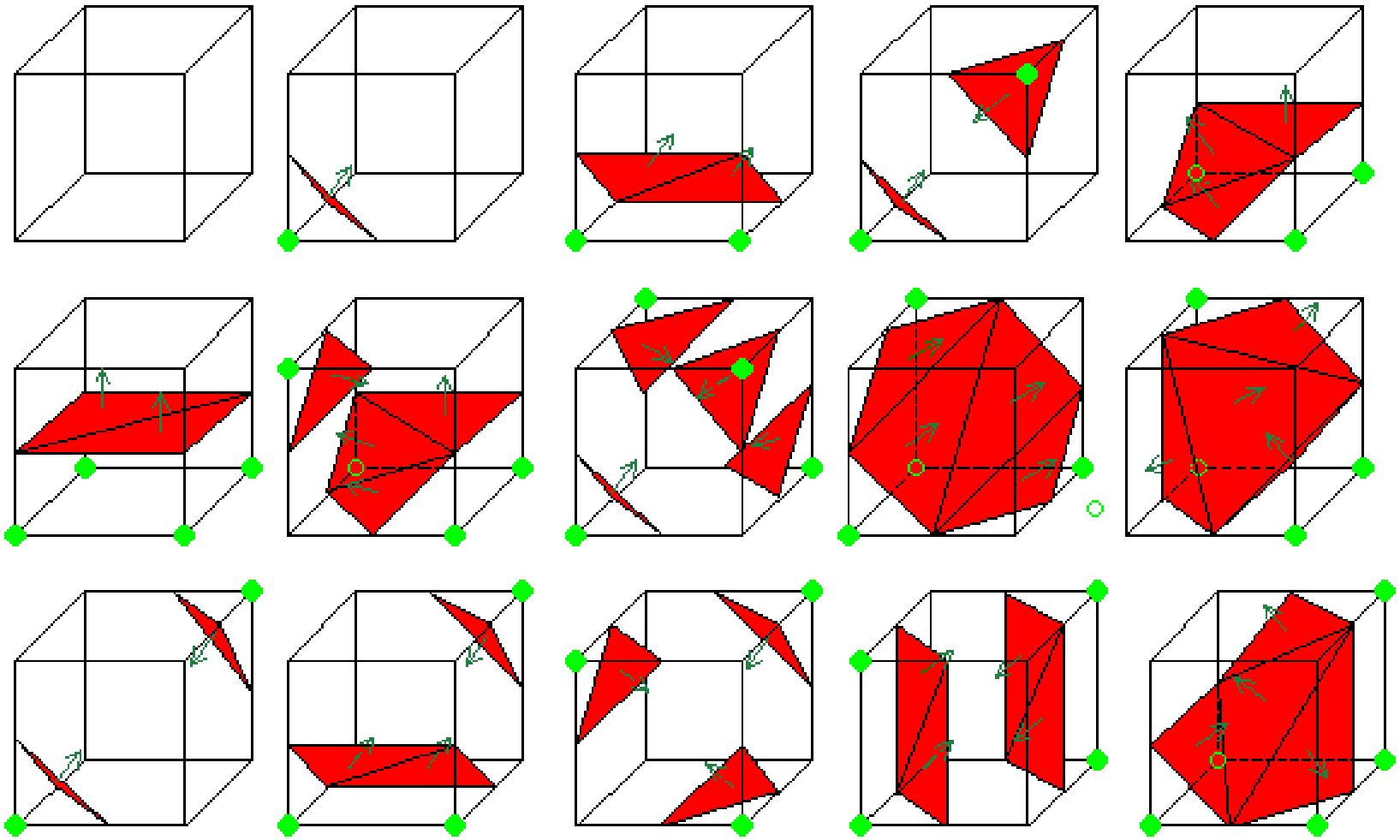
Marching Squares (2D)

- Evaluate each cell separately
- Check which vertices are inside/outside
- Generate triangles according to 16 lookup tables





Marching Cubes (3D)



http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html

Signed Distance Functions

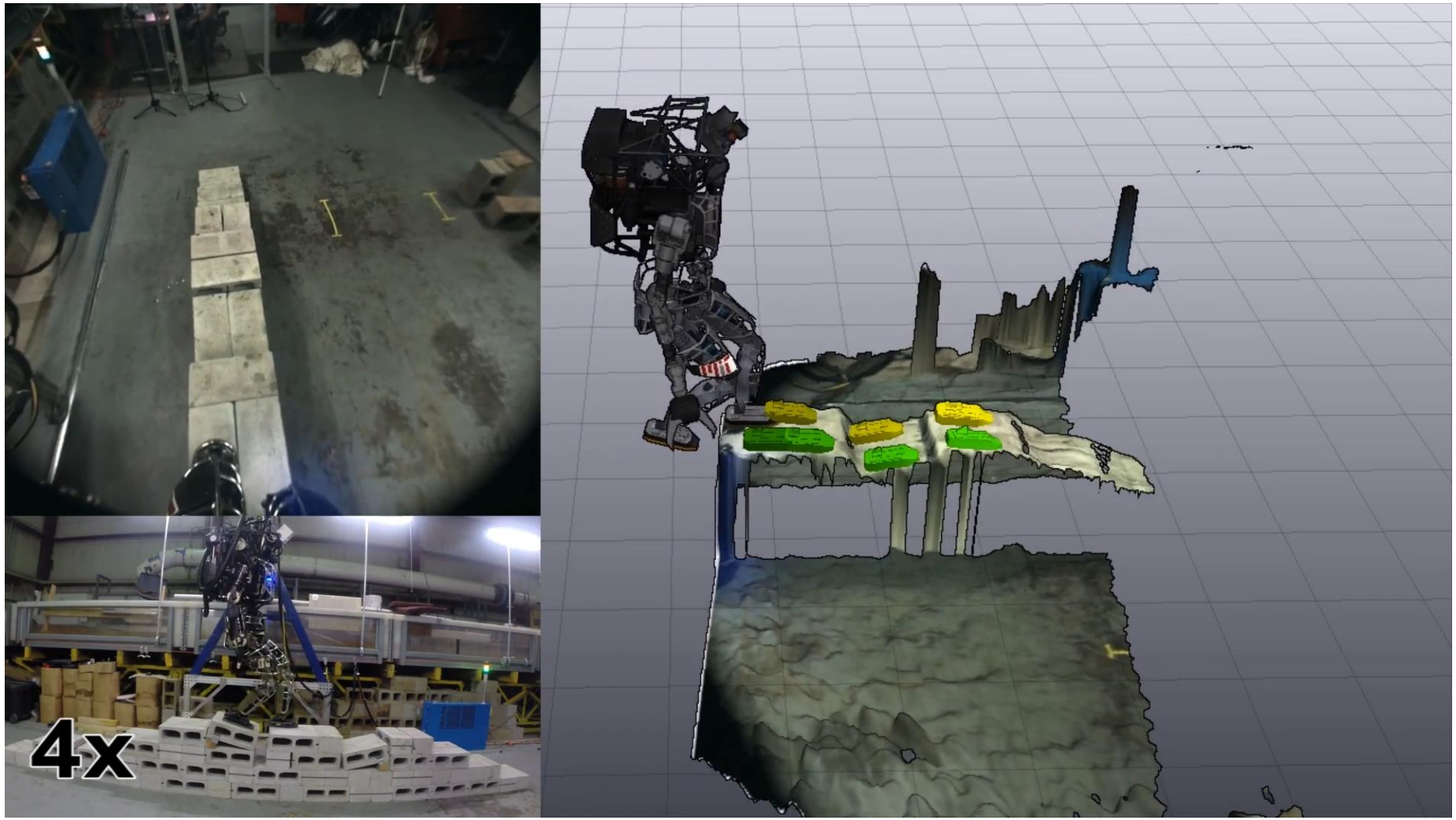
Pros

- Full 3D model
- Sub-voxel accuracy
- Supports fast GPU implementations

Cons

- Space-consuming voxel grid
- Polygonization: slow

Application: Estimation of Traversable Terrain using TSDF



[Fallon, Deits, Whelan, Antone, McDonald, and Tedrake, Humanoids 2015]

Application: Learning Accurate 3D Models

[Sturm, Bylow, Kahl, Cremers; GCPR 2013]

Multi-Layer Mapping

Multi-Layer Mapping

- All the 3D world representations studied till now have geometric features
  - Occupancy probability
  - Weight and Signed Distance Field
- However, scene understanding need other features in addition to metric features
  - Semantics i.e. class labels
  - Instance ID
  - Uncertainty/Confidence

Semantic Aware Volumetric Mapping

Key idea:

- Combine semantic scene understanding with geometric information to produce semantic aware volumetric mapping
- Semantics about objects/environment enable better abstraction and long term interaction

2D to 3D projection of Semantics

- Typically metric-semantic maps have 2 separate layers:
  - Metric: Occupancy, TSDF
  - Semantics: Class label, Instance label

- Multi-view integration of semantic labels
  - Majority voting: Label with highest votes for a voxel wins
  - Recursive Bayesian update: Requires probabilistic Bayesian neural networks for semantic prediction

Deep Learning based Scene Understanding

- **Semantic Segmentation**
  - Pixel level class labeling
  - No explicit object detection
  - Different instances of same class grouped together
  - No foreground/background differentiation
  - Application: Geo-sensing, autonomous driving

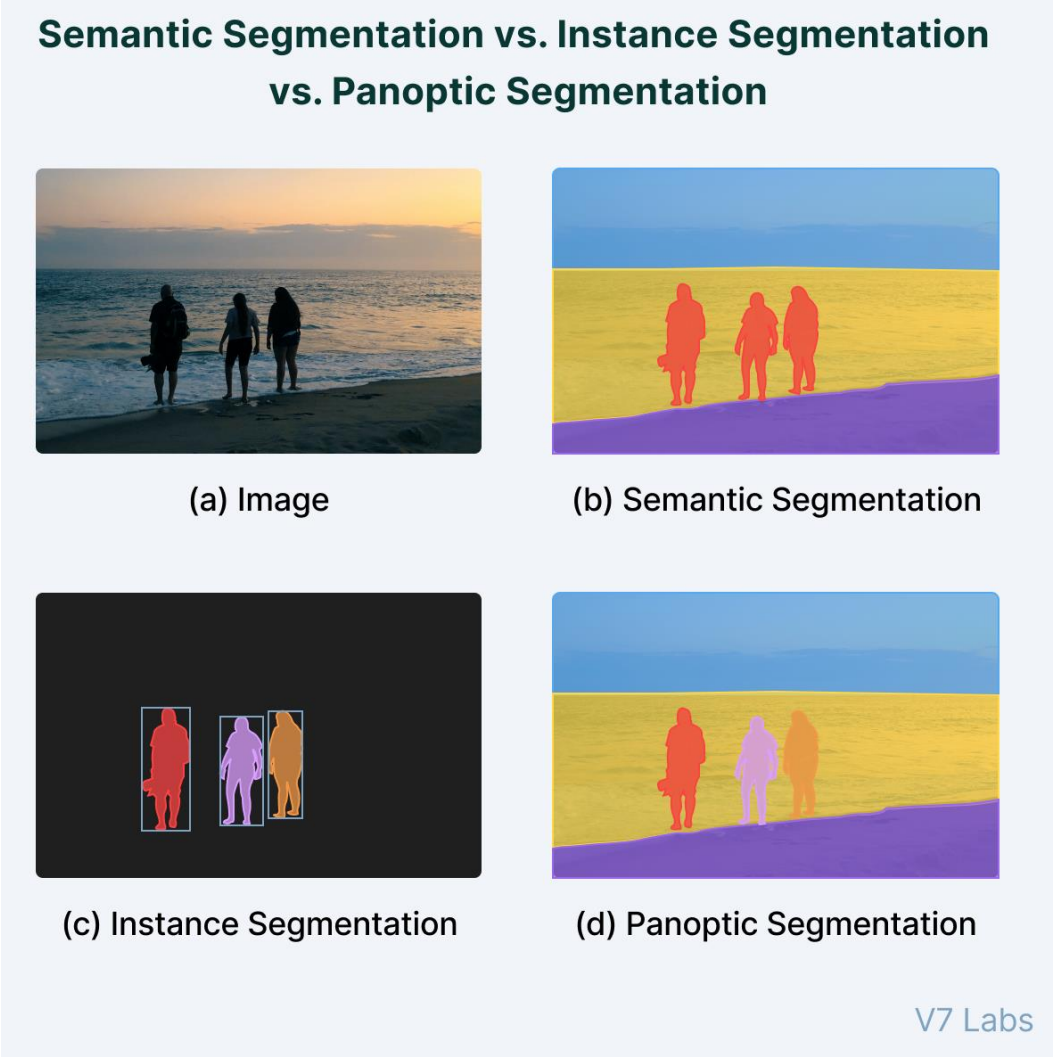


Figure from V7 Labs: https://www.v7labs.com/blog/panoptic-segmentation-guide

Deep Learning based Scene Understanding

- **Instance Segmentation**
  - Hybrid of object detection and semantic segmentation
  - Unique instance id for two different objects of same class
  - Mostly used for countable foreground objects like humans, chairs, cars
  - Application: Grasping, object tracking

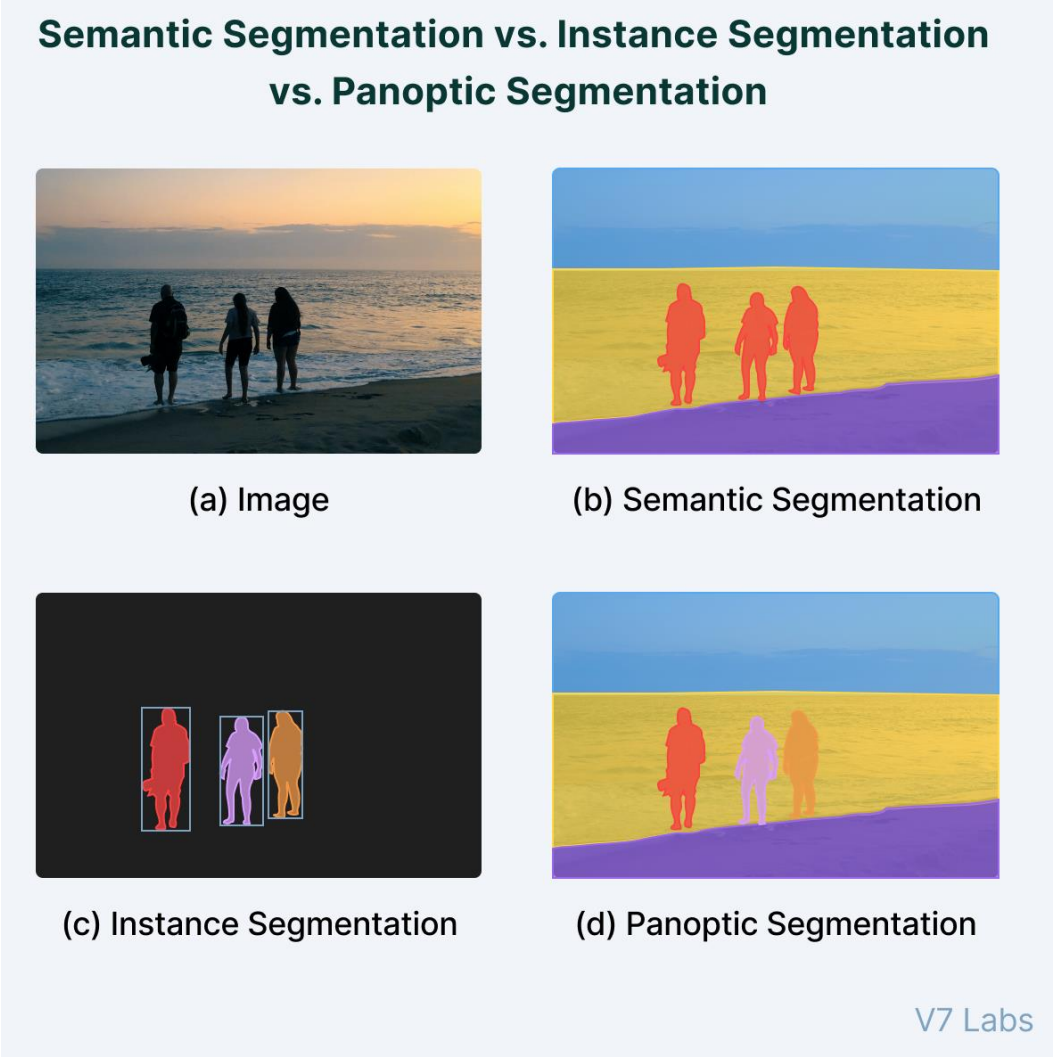


Figure from V7 Labs: https://www.v7labs.com/blog/panoptic-segmentation-guide

Deep Learning based Scene Understanding

- **Panoptic Segmentation**
  - Combination of semantic and instance segmentation
  - Foreground (things) with instance ids
  - Background (stuff) with no instance ids
  - Richer semantic information
  - Long term object interaction

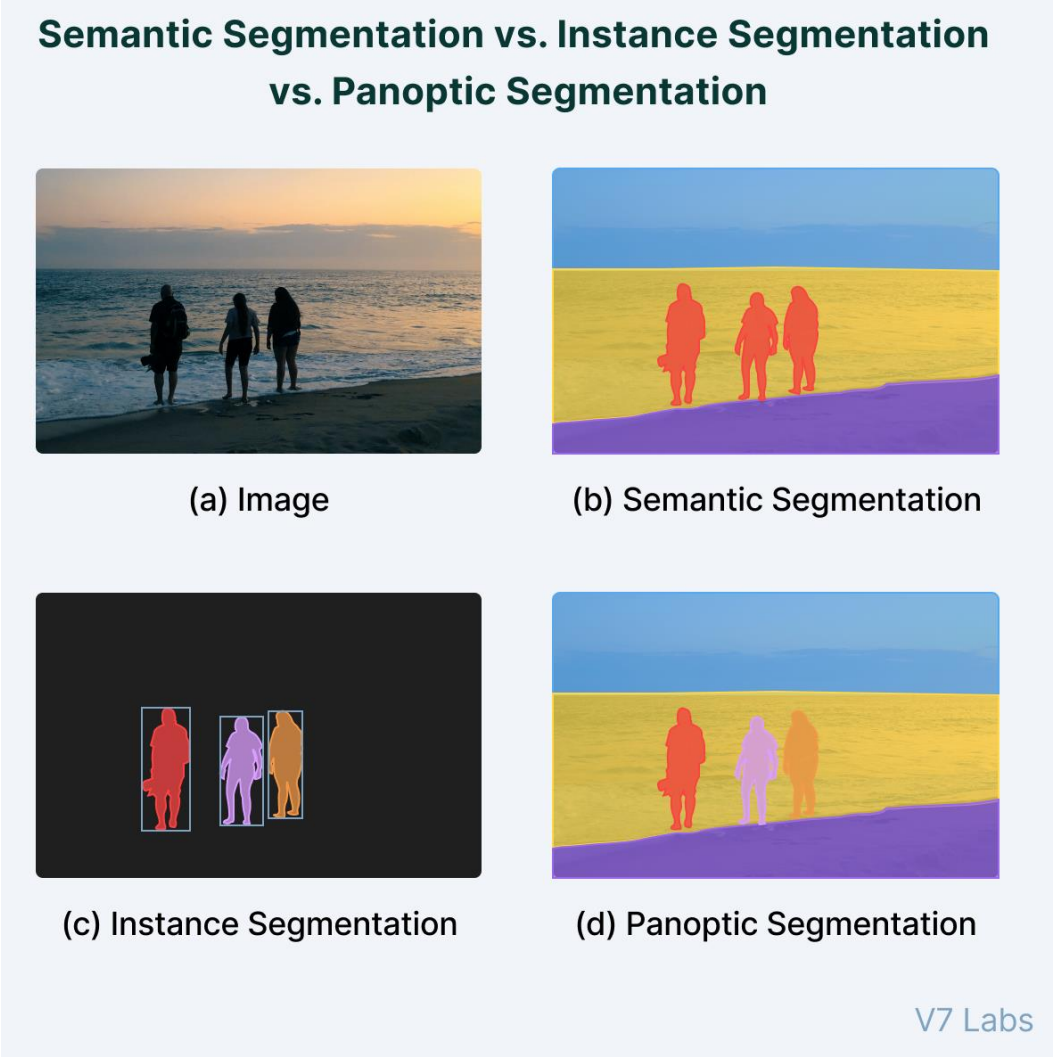


Figure from V7 Labs: https://www.v7labs.com/blog/panoptic-segmentation-guide

Majority Voting for Semantic Labels

- Let K be the number of classes for each semantic voxel v
- Each voxel v maintains a vote vector $\boldsymbol{c} = [c_1, c_2, \ldots, c_k, \ldots, c_{K-1}, c_K]$
- At time t, let measurement assign class k to the voxel v

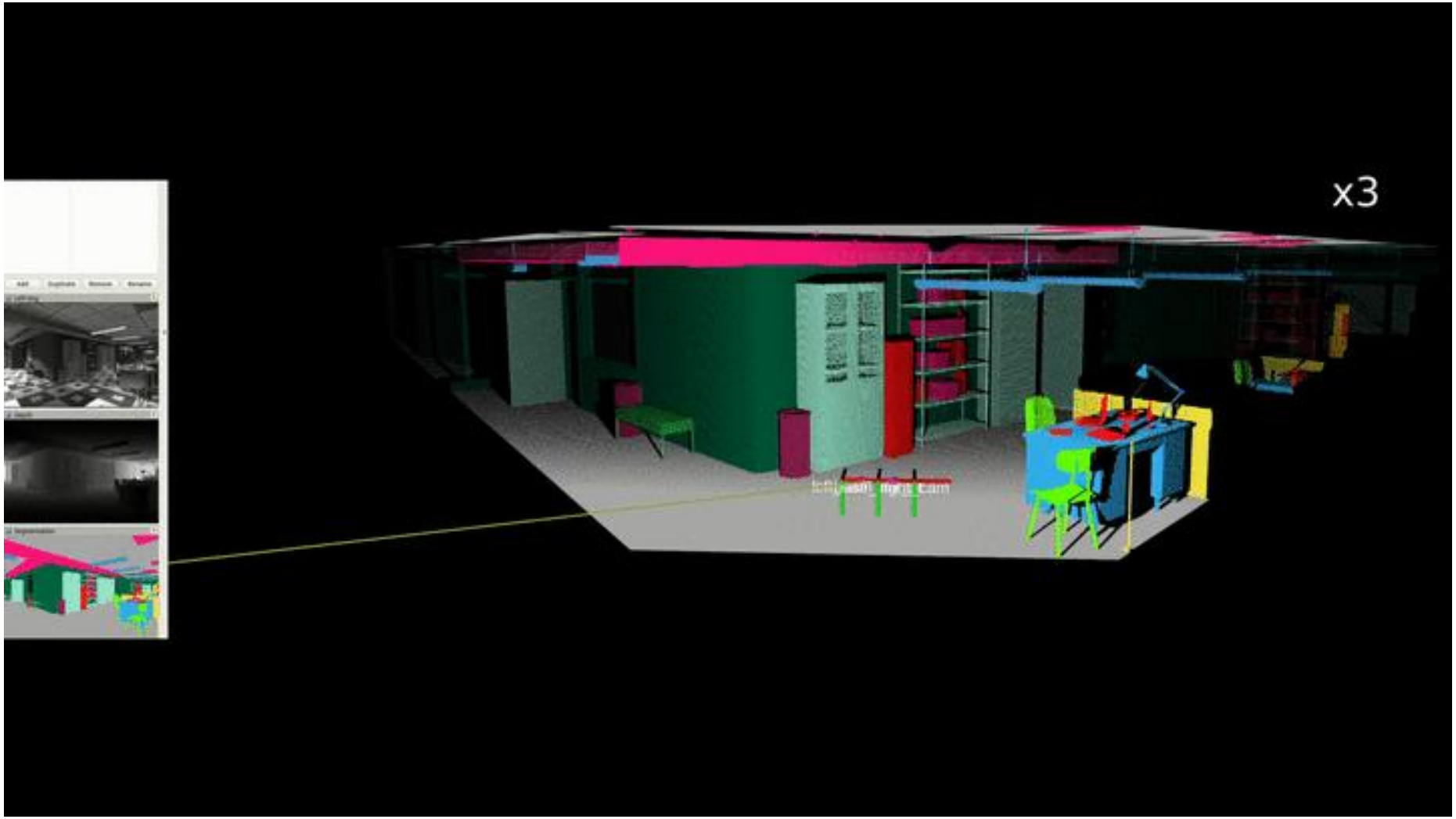
$$\boldsymbol{c_t} = \boldsymbol{c_{t-1}} + [0_1, 0_2, \ldots, 1_k, \ldots, 1_{K-1}, 1_K]$$

- For conversion from vote vector to hard label for voxel v

$$l_t^v = argmax(\boldsymbol{c_t})$$

- What happens when 2 classes have equal number of votes?

Semantic Mapping



[A. Rosinol, M. Abate, Y. Chang and L. Carlone, ICRA 2020]

Semantic Instance Volumetric Mapping

- Convolutional networks e.g. Mask RCNN, Panoptic DeepLab, enable to label objects in RGB images
- Fuse per frame semantic information from RGB images with geometric information from range data

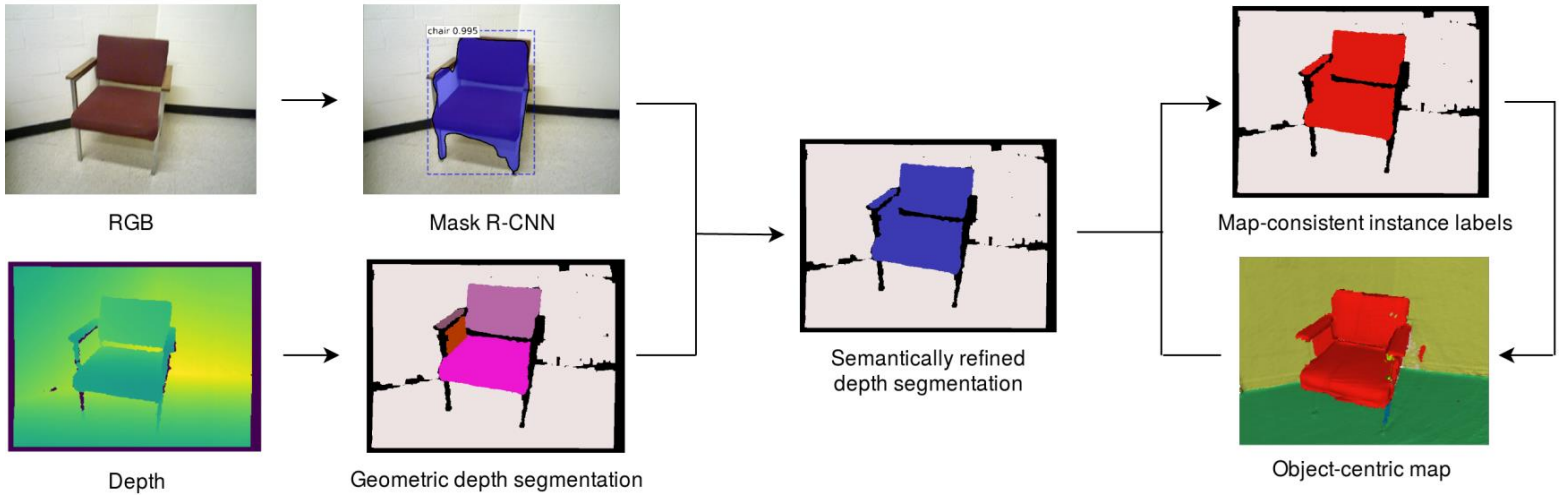


Figure from "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery " by Grinvald et al., 2019

Visual Semantic Learning for Navigation

[P. Roth, J. Nubert, F. Yang, M. Mittal and M. Hutter, ICRA 2024]

Semantic Aware Volumetric Mapping

Pros

- Semantic rich maps
- Enables semantic based planning and interaction
- Closer to how humans abstract environment

Cons

- Requires large neural network models
- Complex and expensive

Figure from "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery " by Grinvald et al., 2019

Scene Graph Representation

Hierarchical Perception Layers

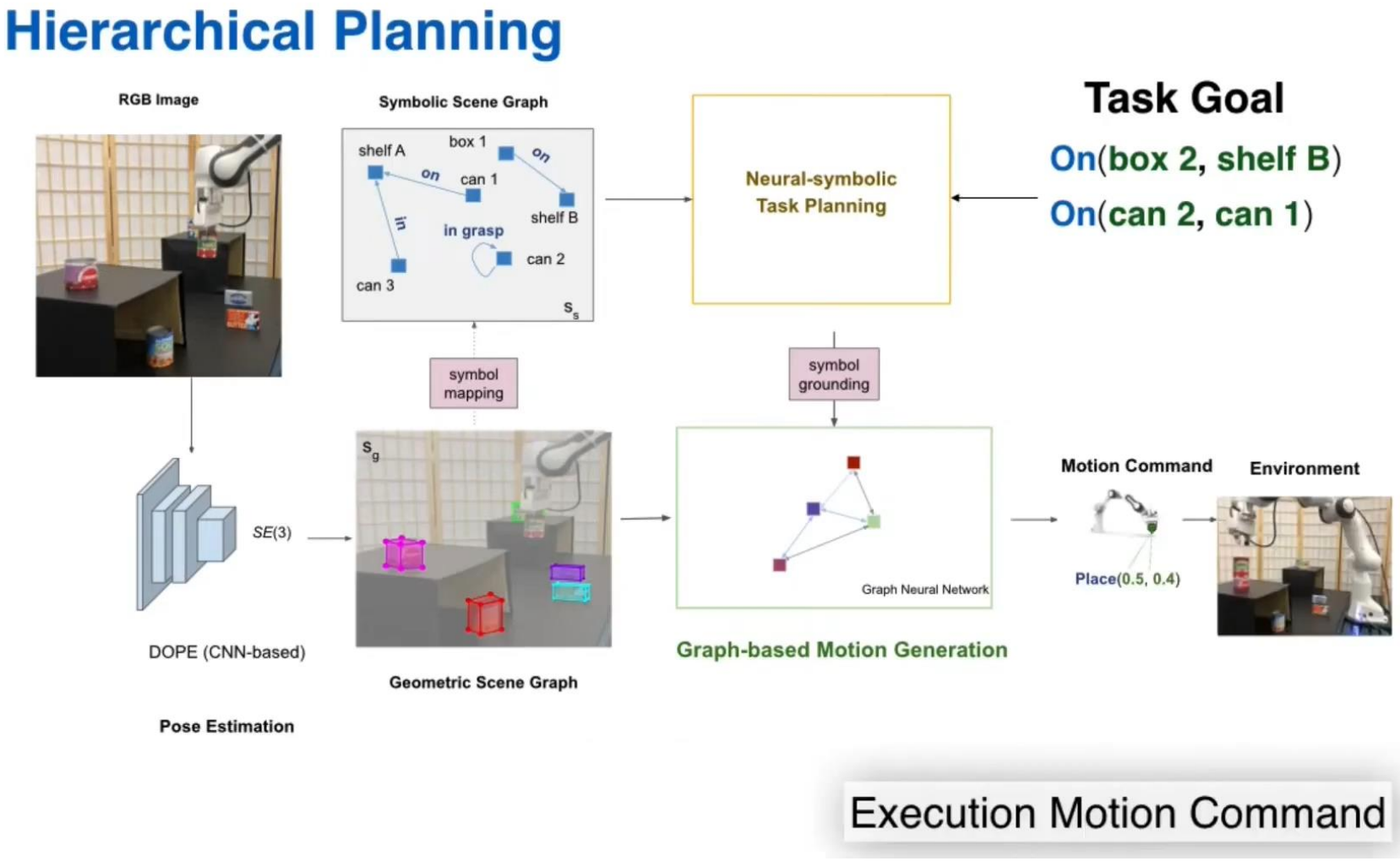
Raw Geometry (Point Clouds)

Metric Representation (TSDF, Octomap)

Semantic-Metric Representation (Semantic Maps)

Hierarchial Relationships (Scene Graphs)

Scene Graphs



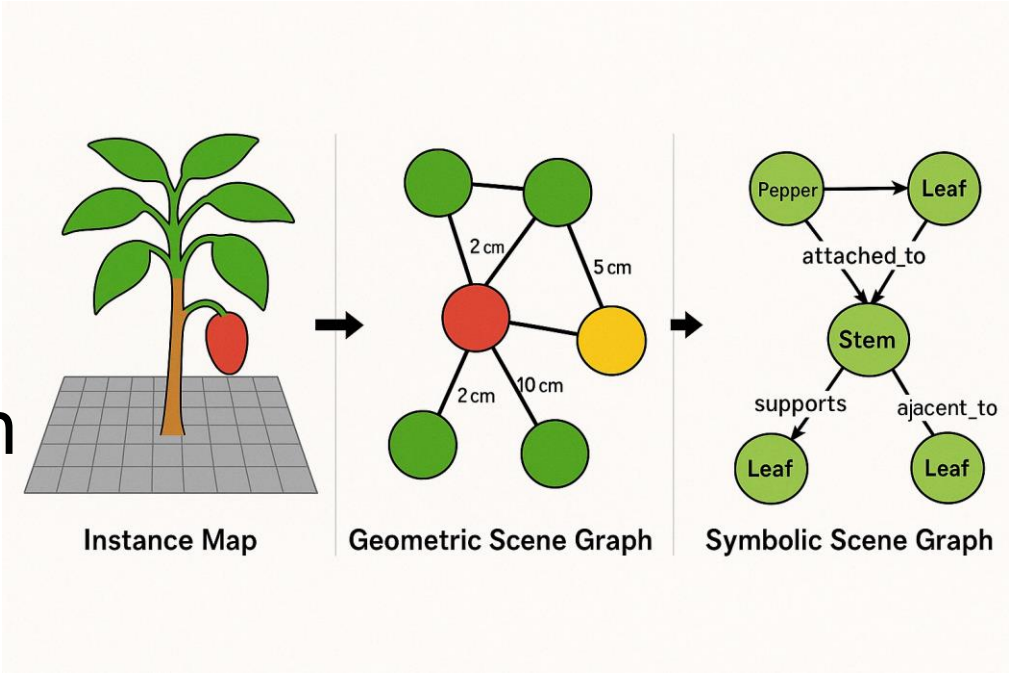
[Y. Zhu, J. Tremblay, S. Birchfield and Y. Zhu, ICRA 2021]

Scene Graph Basics

- Nodes represent objects and agents
- Edges encode spatial and semantic relations
- Structured 3D environment representation
- Enables high-level environment queries
- Incrementally updated with perception data

Example Scene Graph Generation

- Geometric scene graphs
  - Extract centroids of instances → vertices
  - Edges represent distance between centroids
- Semantic scene graphs
  - Define dictionary of relations using triplets <subject, object, relation>
  - E.g. <pepper, stem, attached_to>



Source: ChatGPT

Summary

- The best model depends on the application
- Voxel representations allow for a full 3D representation
- Octrees are a compact, inherently multi-resolution, probabilistic 3D representation
- Surface models support traversability & graspability analysis
- Signed distance functions also use 3D grids but allow for a sub-voxel accuracy representation of the surface
- Semantic aware volumetric maps enable long term interaction
- Scene graphs are abstract models → task planning

Literature 3D World Models (1)

- *Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing*,
  R. Triebel, P. Pfaff, and W. Burgard, IEEE/RSJ Int. Conf. on Int. Robots and Systems (IROS), 2006
- *OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees*,
  A. Hornung,. K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, Autonomous Robots, 2013
- *World Modeling*
  W. Burgard, M. Herbert, and M. Bennewitz, Handbook of Robotics (2nd edition), Chapter 45, Springer, 2016.
- *Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions*,
  E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, Robotics: Science and Systems (RSS), 2013
- *Real-time navigation in 3D environments based on depth camera data*
  D. Maier, A. Hornung and M. Bennewitz, Humanoids 2012
- *Continuous Humanoid Locomotion over Uneven Terrain using Stereo Fusion*,
  M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, Humanoids 2015

Literature 3D World Models (2)

- *Survey of 3D modeling using depth cameras*
  Xu, Hantong & Xu, Jiamin & Xu, Weiwei. (2019), Virtual Reality & Intelligent Hardware. 1. 483-499. 10.1016/j.vrih.2019.09.003.
- *Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery*,
  M. Grinvald, F. Furrer, T. Novkovic, J.J. Chung, C. Cadena, R. Siegwart, & J. Nieto, IEEE Robotics and Automation Letters, 2019
- *Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping*
  A. Rosinol, M. Abate, Y. Chang and L. Carlone, ICRA 2020
- *ViPlanner: Visual Semantic Imperative Learning for Local Navigation*
  P. Roth, J. Nubert, F. Yang, M. Mittal and M. Hutter, ICRA 2024
- *Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graph*,
  Y. Zhu, J. Tremblay, S. Birchfield and Y. Zhu, ICRA 2021

Acknowledgment

- Previous versions of parts of the slides have been created by Wolfram Burgard, Cyrill Stachniss, and Jürgen Sturm