Introduction to Mobile Robotics

Techniques for 3D Mapping

Wolfram Burgard, Maren Bennewitz,

Diego Tipaldi, Luciano Spinello

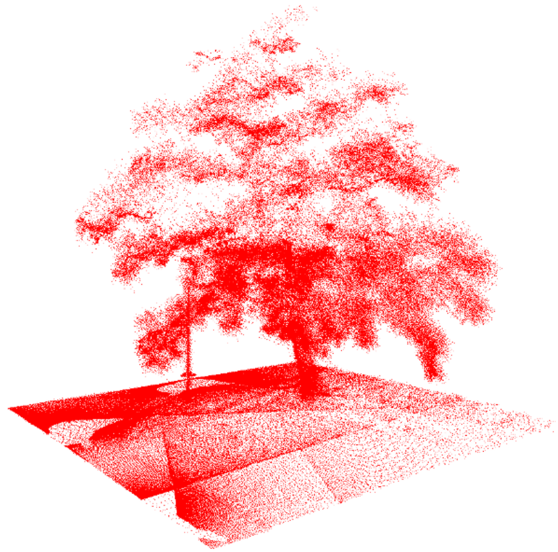
Why 3D Representations

- Robots live in the 3D world.
- 2D maps have been applied successfully for navigation tasks such as localization.
- Reliable collision avoidance and path planning, however, requires accurate 3D models.
- How to represent the 3D structure of the environment?

Popular Representations

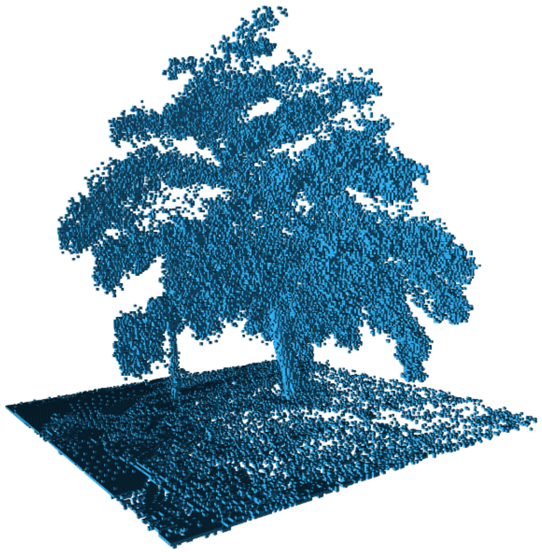
- Point clouds
- Voxel grids
- Surface maps
- Meshes
- …

Point Clouds



- Pro:
  - No discretization of data
  - Mapped area not limited
- Contra:
  - Unbounded memory usage
  - No direct representation of free or unknown space

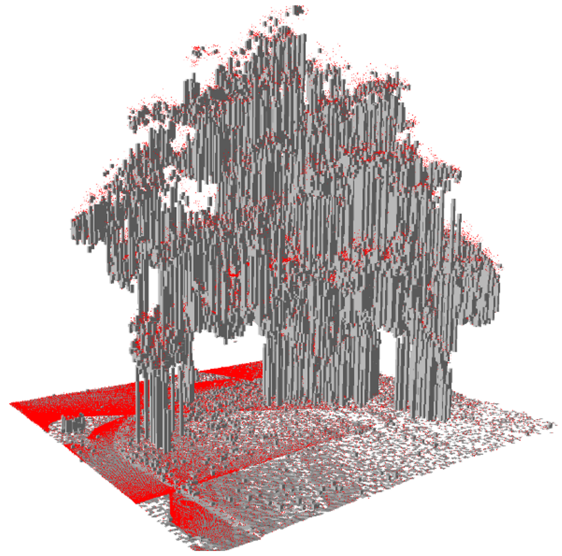
3D Voxel Grids



- Pro:
  - Volumetric representation
  - Constant access time
  - Probabilistic update
- Contra:
  - Memory requirement: Complete map is allocated in memory
  - Extent of the map has to be known/guessed
  - Discretization errors

2.5D Maps: "Height Maps"

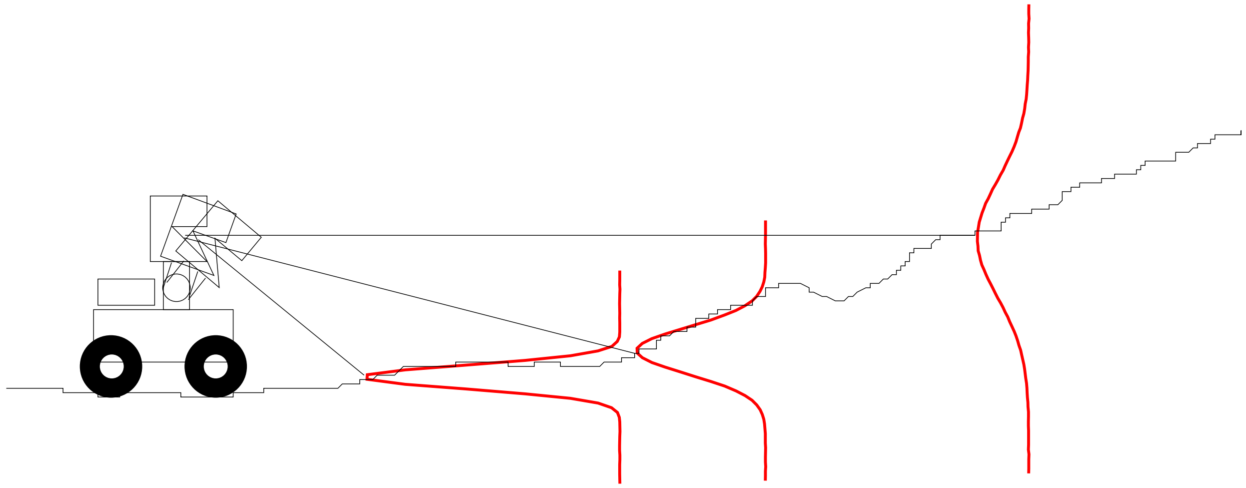
Average over all scan points that fall into a cell



- Pro:
  - Memory efficient
  - Constant time access
- Contra:
  - Non-probabilistic
  - No distinction between free and unknown space

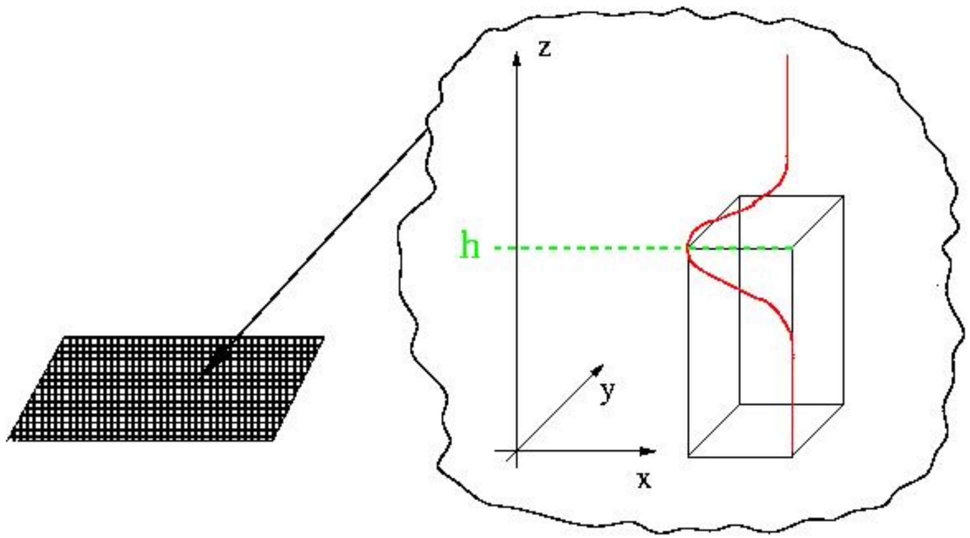
Elevation Maps

- 2D grid that stores an estimated height (elevation) for each cell
- Typically, the uncertainty increases with measured distance



Elevation Maps

- 2D grid that stores an estimated height (elevation) for each cell
- Typically, the uncertainty increases with measured distance
- Kalman update to estimate the elevation

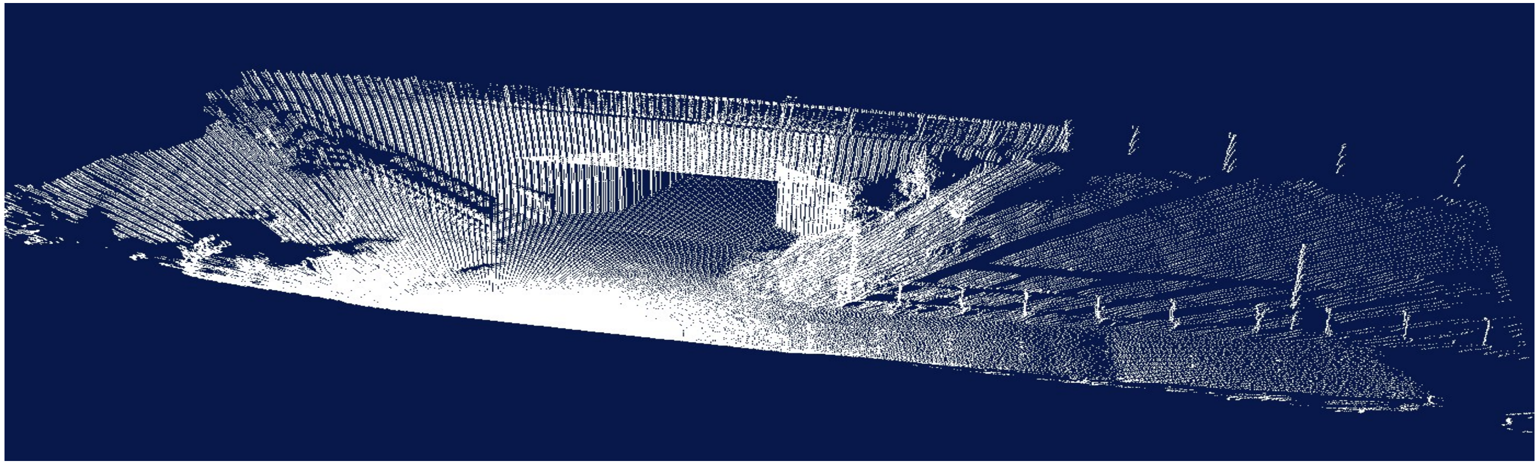


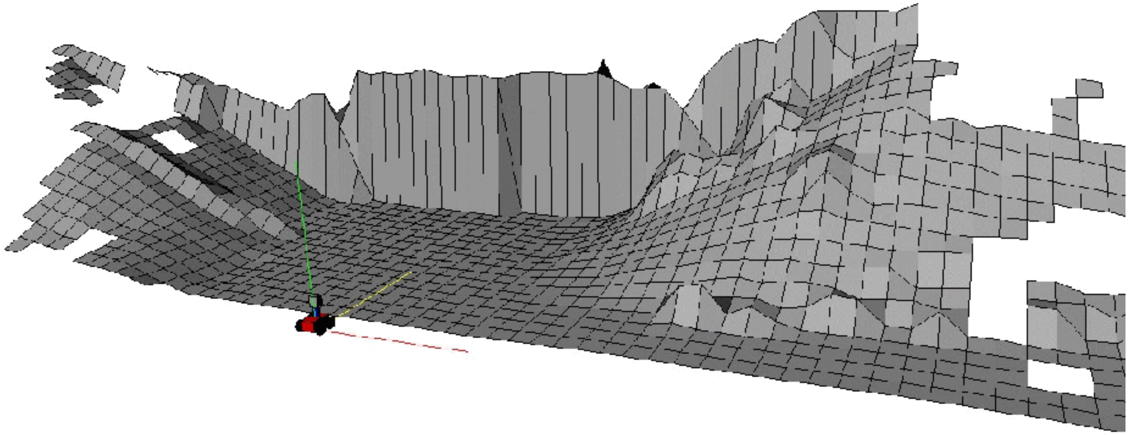
Elevation Maps

- Pro:
  - 2.5D representation (vs. full 3D grid)
  - Constant time access
  - Probabilistic estimate about the height

- Contra:
  - No vertical objects
  - Only one level is represented

Typical Elevation Map

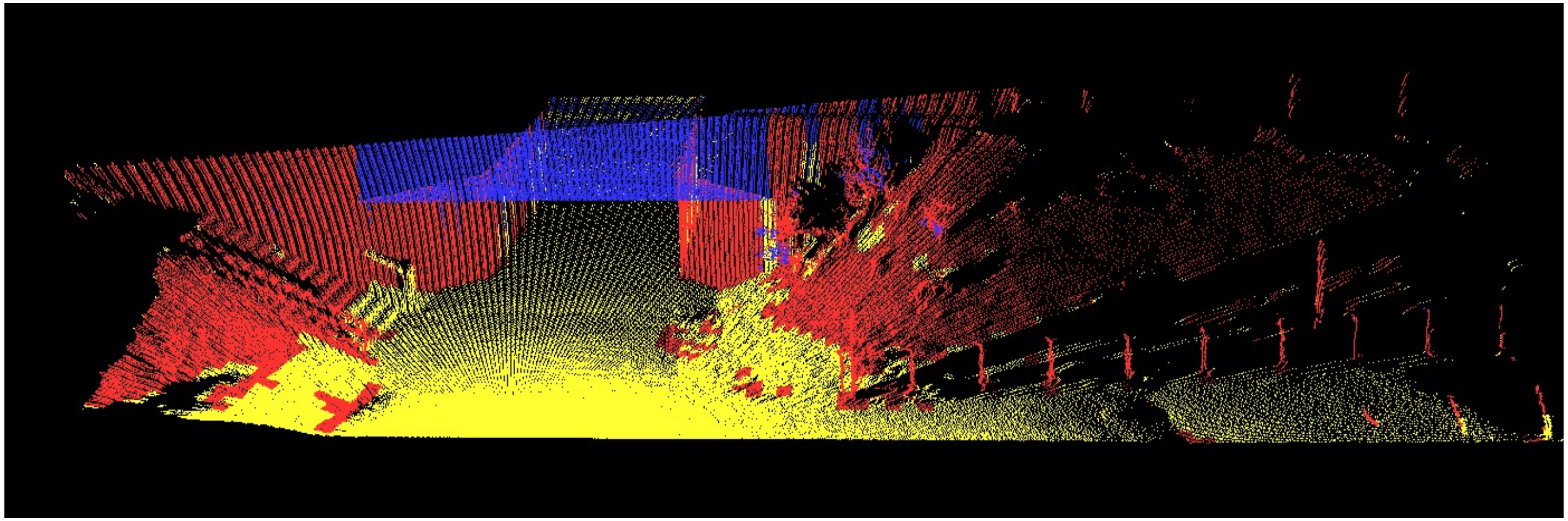




Extended Elevation Maps

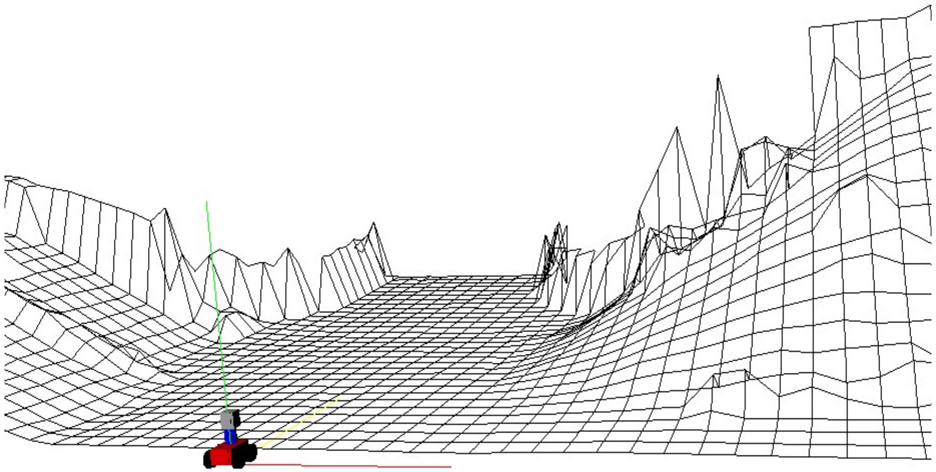
- Identify
  - Cells that correspond to vertical structures
  - Cells that contain gaps
- Check whether the variance of the height of all data points is large for a cell
- If so, check whether the corresponding point set contains a gap exceeding the height of the robot (“gap cell”)

Example: Extended Elevation Map



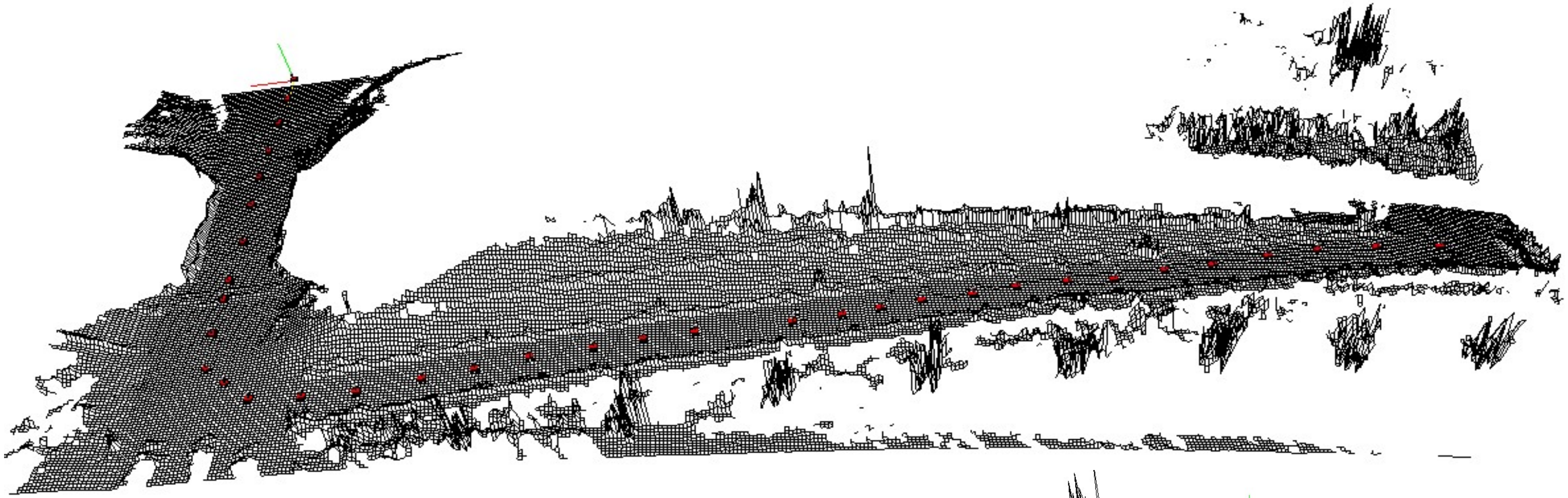
- Cells with vertical objects (red)
- Data points above a big vertical gap (blue)
- Cells seen from above (yellow)

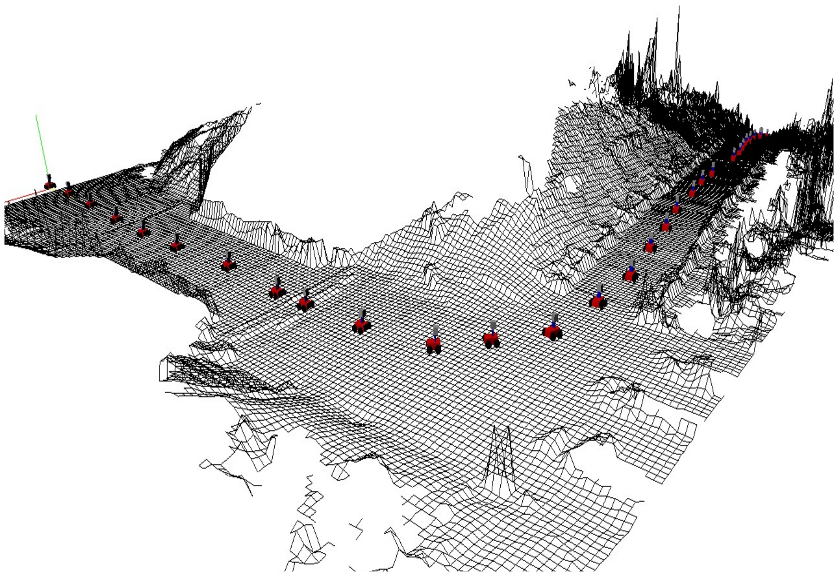
→ use gap cells to determine traversability

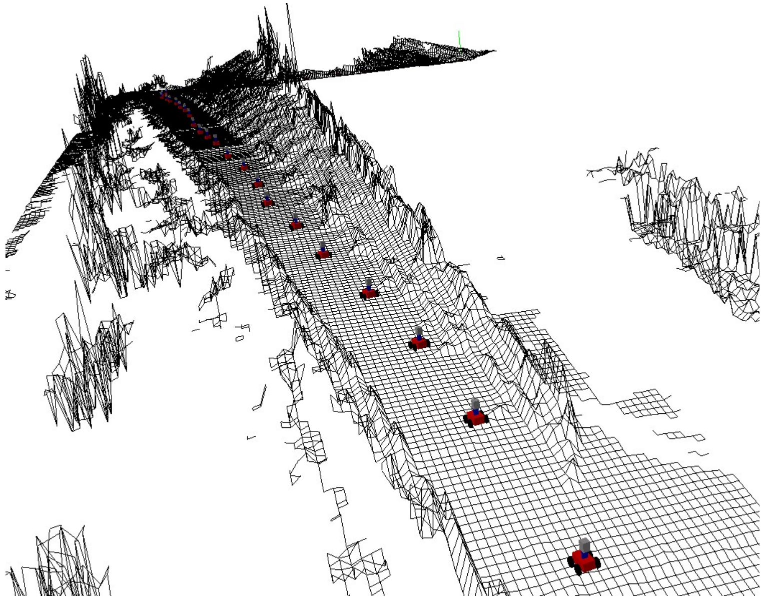


extended elevation map

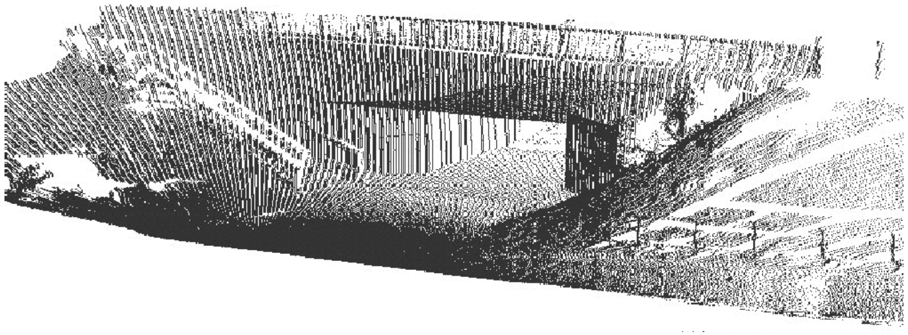
Example Elevation Maps



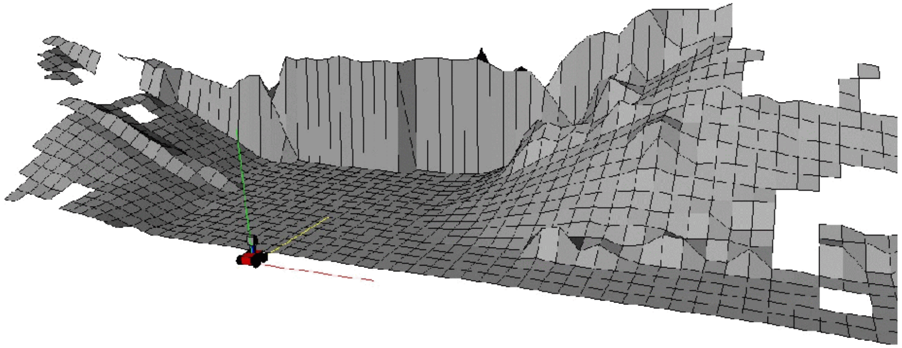




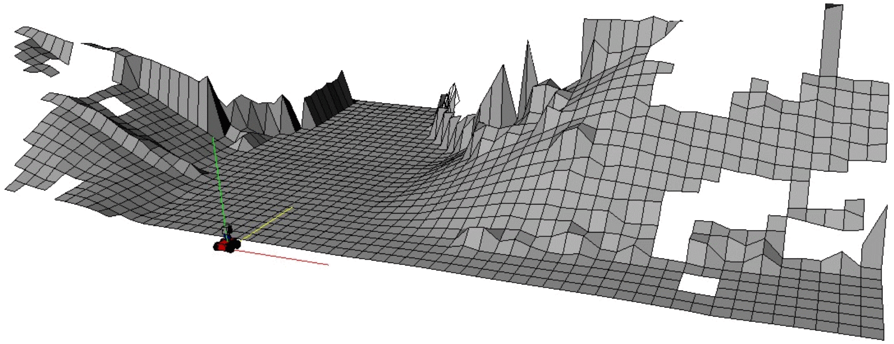
Types of Terrain Maps



Point cloud

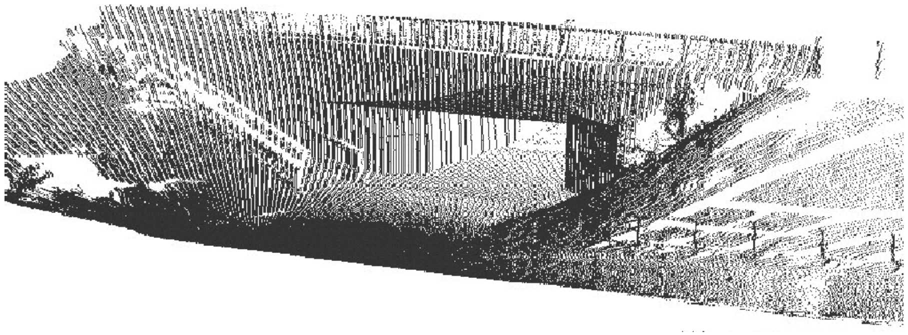


Standard elevation map

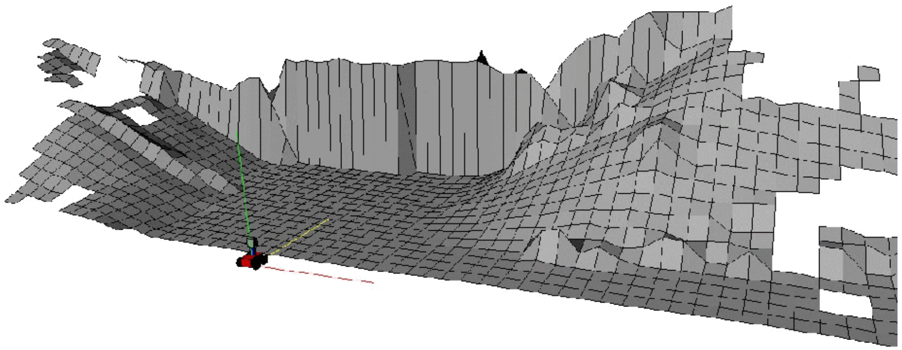


Extended elevation map

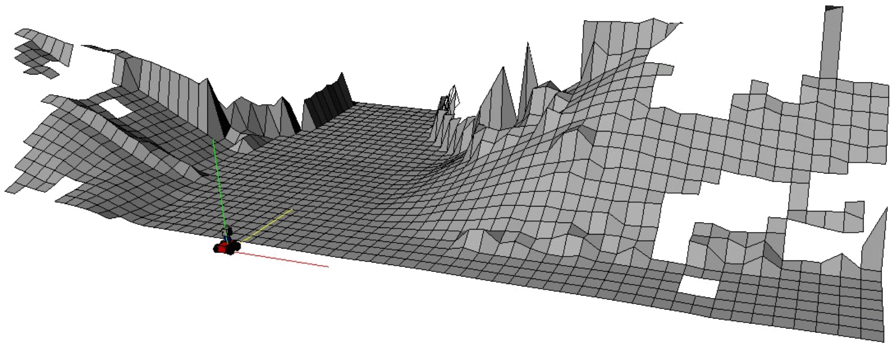
Types of Terrain Maps



Point cloud



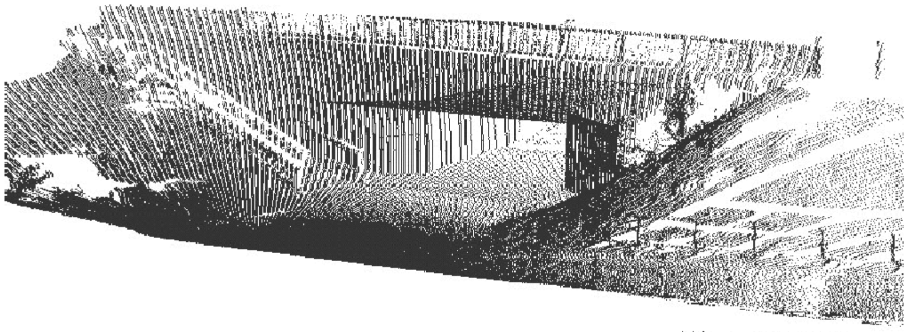
Standard elevation map



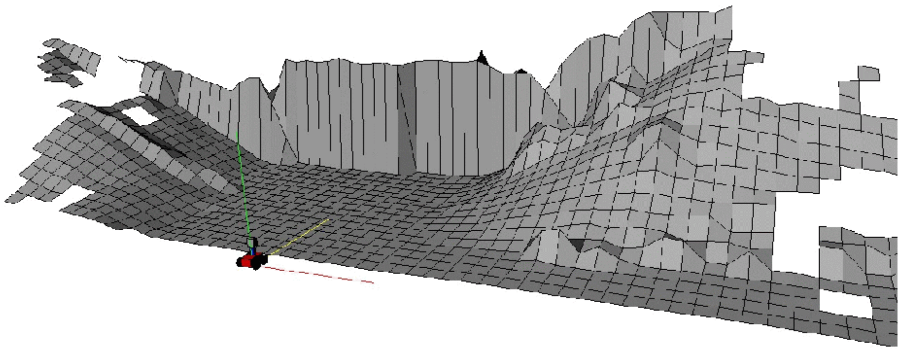
Extended elevation map

+ Planning with underpasses possible (cells with vertical gaps)
− No paths passing under **and** crossing over bridges possible (only one level per grid cell)

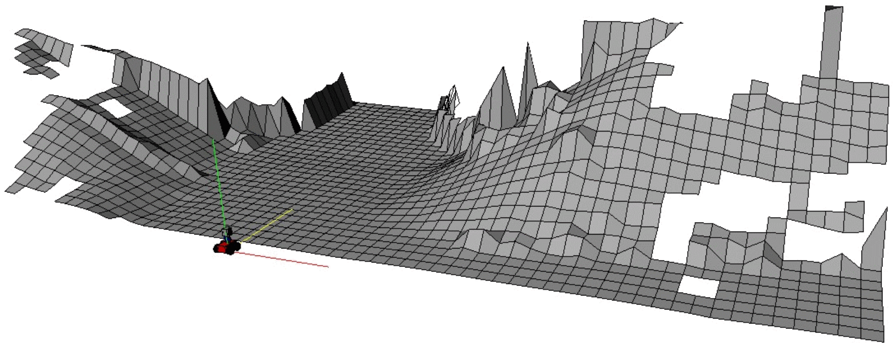
Types of Terrain Maps



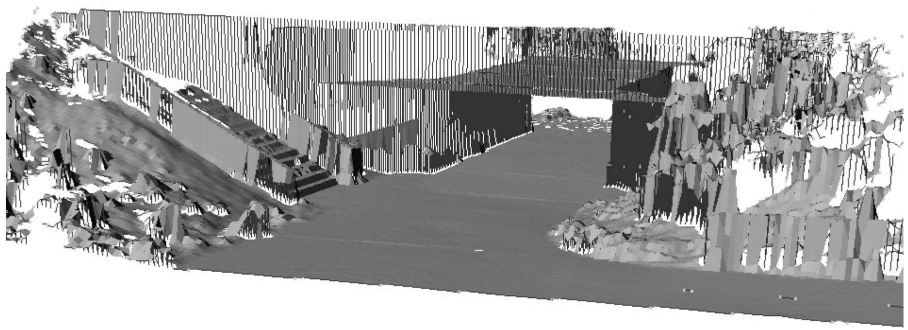
Point cloud



Standard elevation map

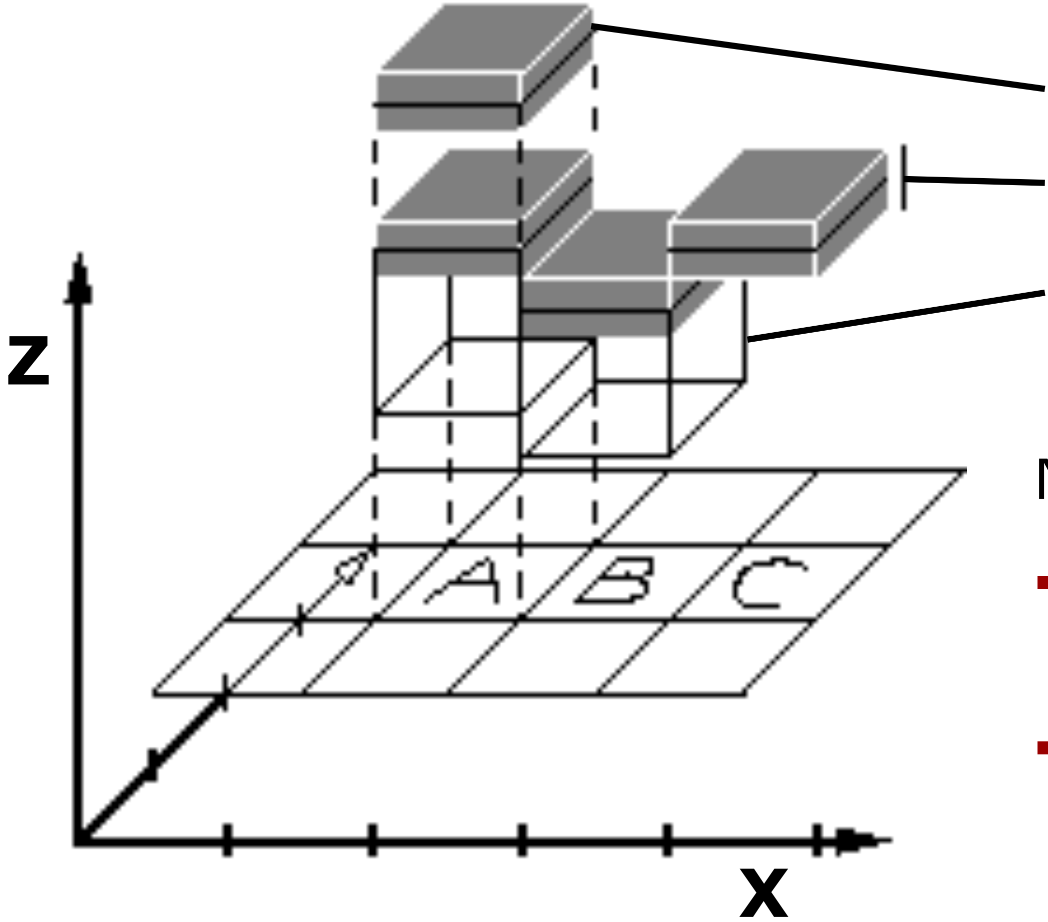


Extended elevation map



Multi-level surface map

MLS Map Representation



Each 2D cell stores various patches consisting of:

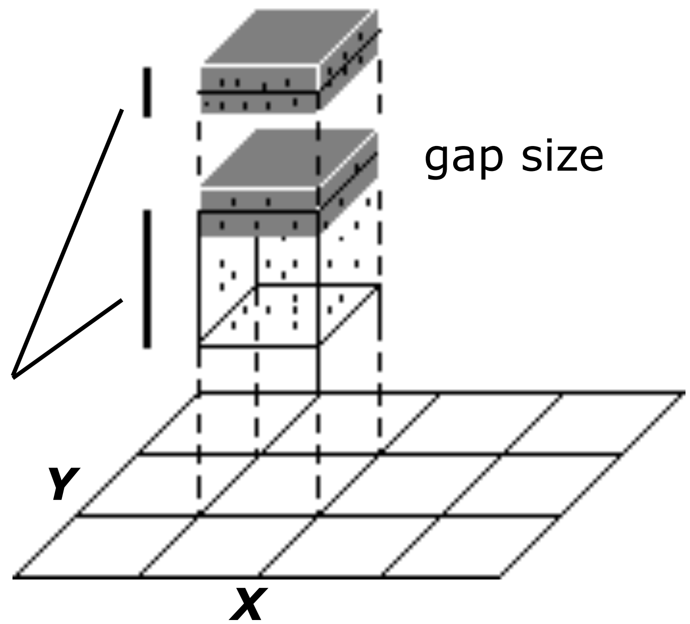
- The height mean μ
- The height variance σ
- The depth value *d*

Note:

- A patch can have no depth (flat objects, e.g., floor)
- A cell can have one or many patches (vertical gap cells, e.g., bridges)

From Point Clouds to MLS Maps

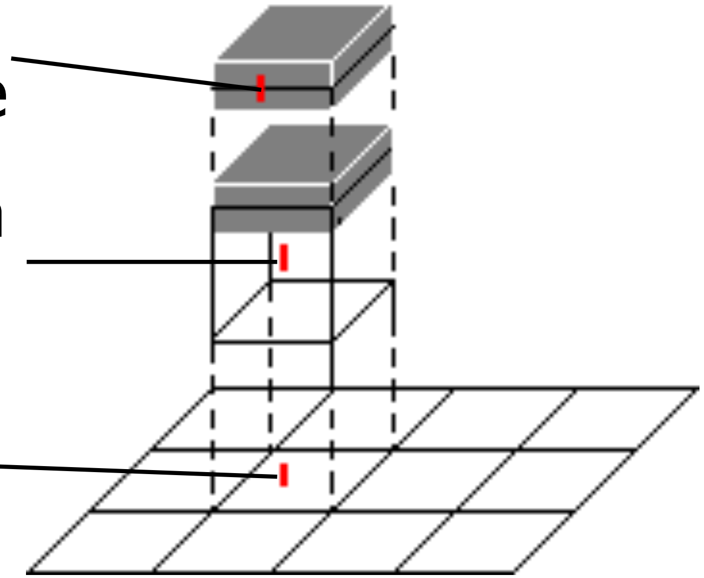
- Determine the cell for each 3D point
- Compute vertical intervals
- Classify into vertical (>10cm) and horizontal intervals



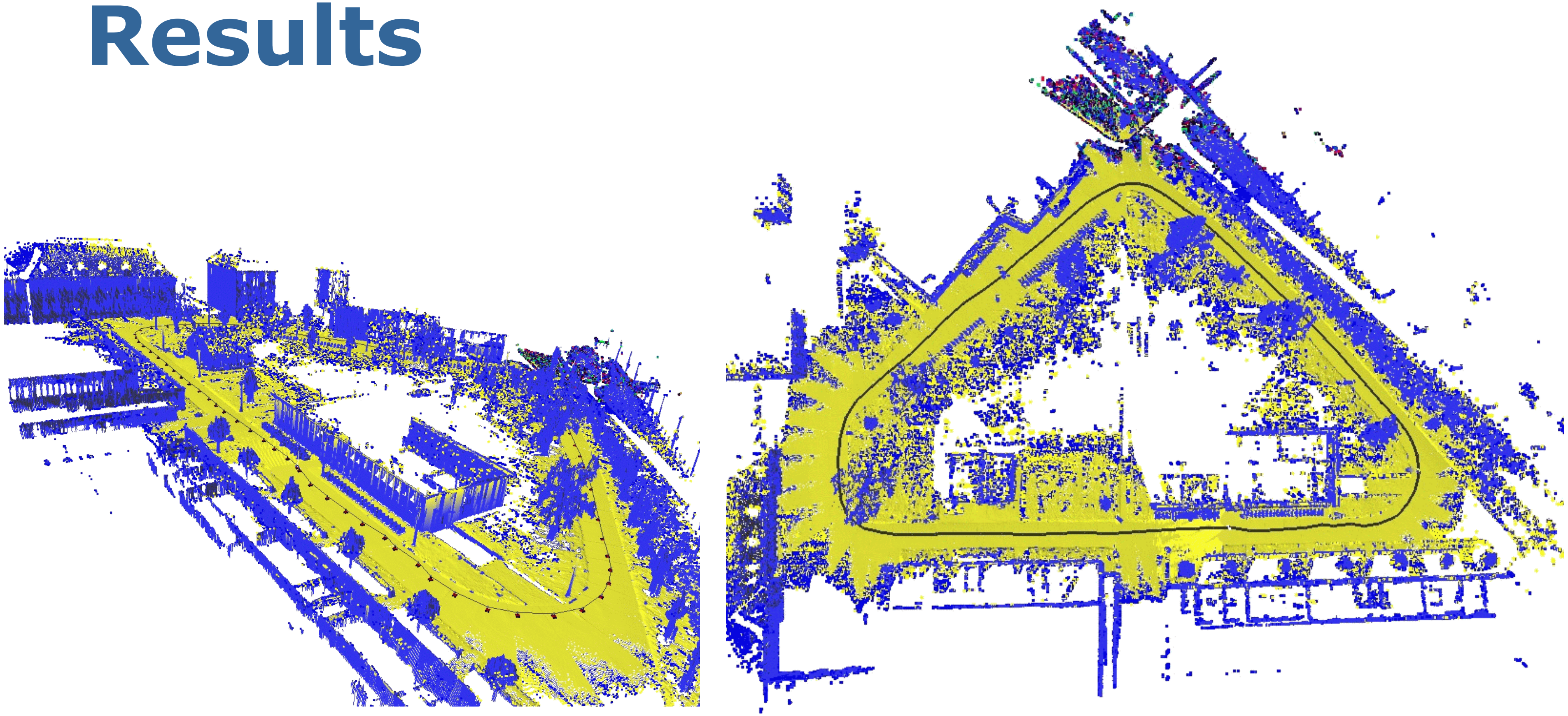
- Apply Kalman update to estimate the height based on all data points for the horizontal intervals
- Take the mean and variance of the highest measurement for the vertical intervals

Map Update

- Given: new measurement $z=(\mathbf{p},\sigma)$ with variance
- Determine the corresponding cell for z
- Find closest surface patch in the cell
- If z is inside 3 variances of the patch, do Kalman update
- If z is in occupied region of a surface patch, disregard it
- Otherwise, create a new surface patch from z

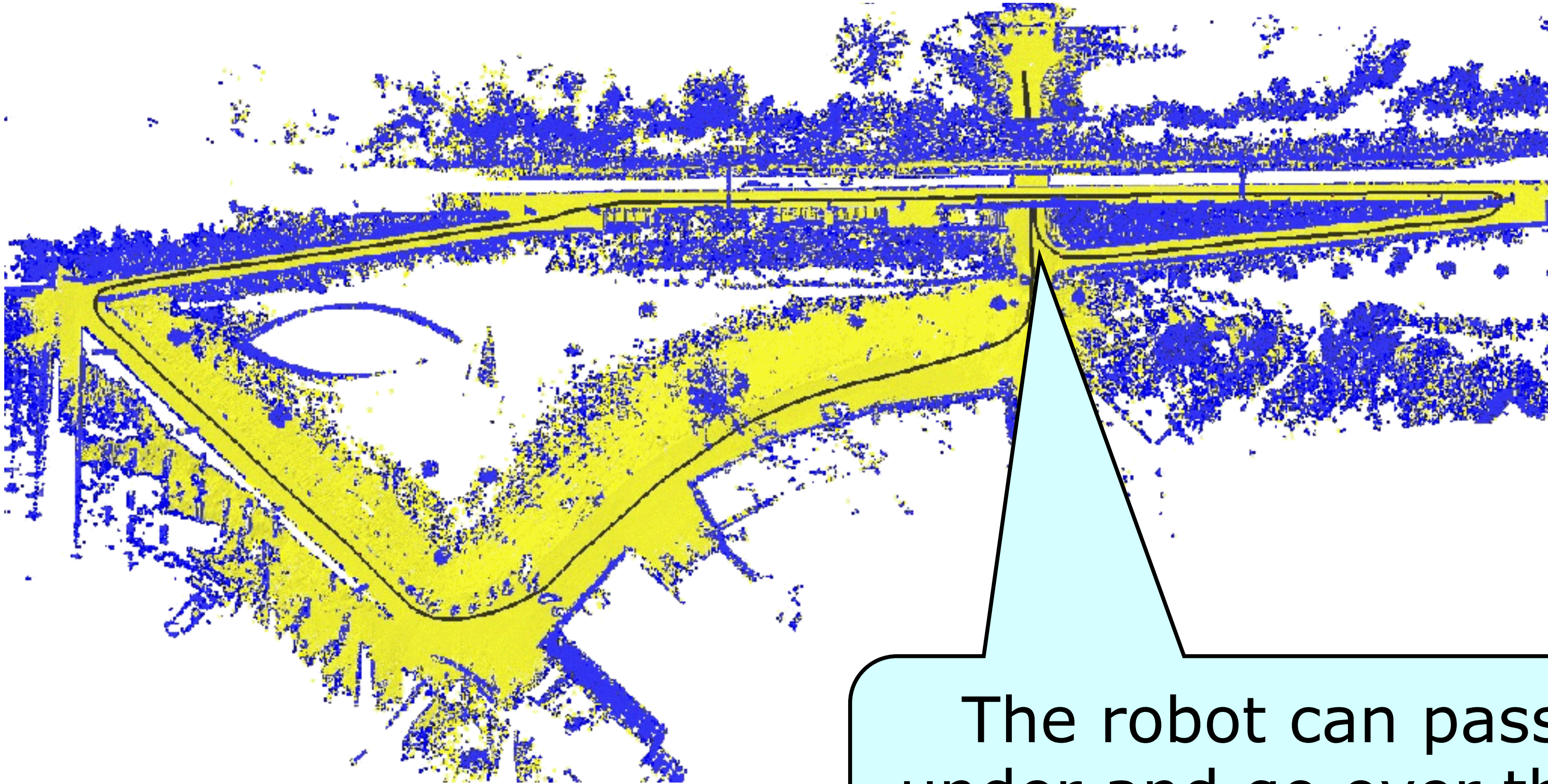


Results



- Map size: 195 by 146 *m*
- Cell resolution: 10 *cm*
- Number of data points: 20,207,000

Results



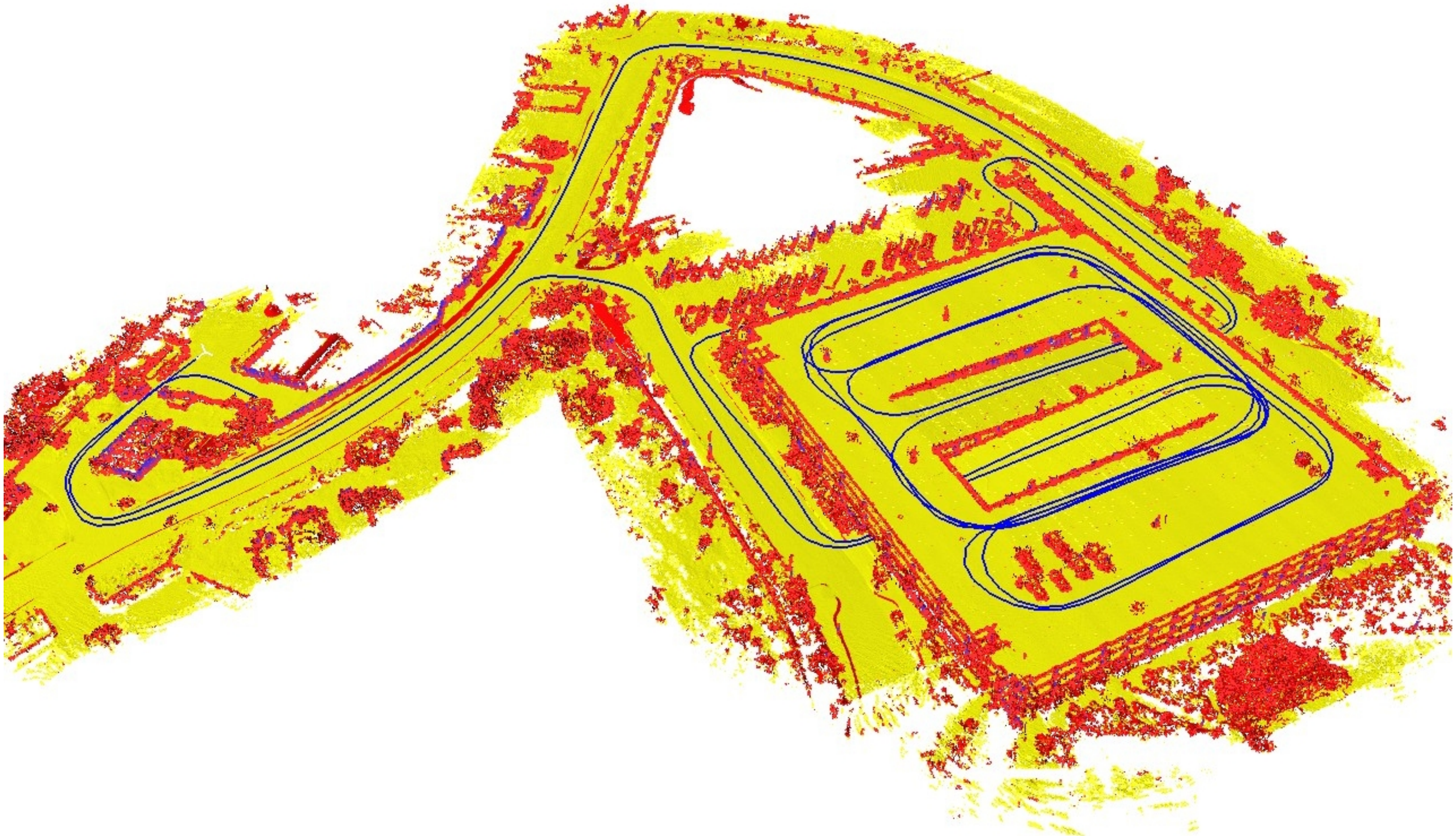
The robot can pass under and go over the bridge

- Map size: 299 by 147 *m*
- Cell resolution: 10 *cm*
- Number of data points: 45,000,000

Experiments with a Car

- Task: Reach a parking spot on the upper level

MLS Map of the Parking Garage

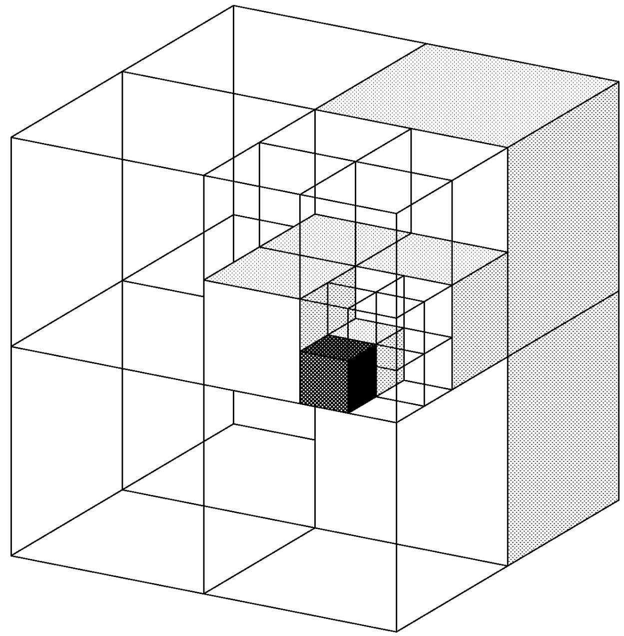


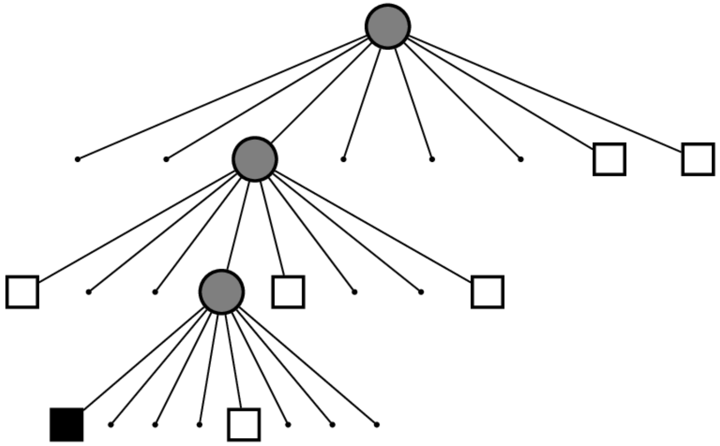
MLS Maps

- Pro:
  - Can represent multiple surfaces per cell
- Contra:
  - No representation of unknown areas
  - No volumetric representation but a discretization in the vertical dimension
  - Localization in MLS maps is not straightforward

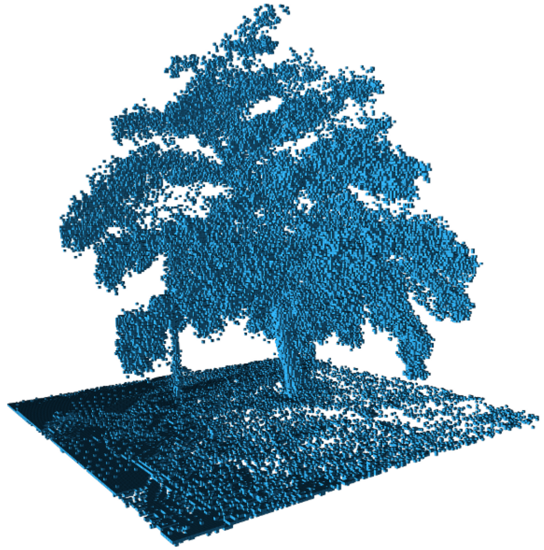
Octree-based Representation

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





Octrees



- Pro:
  - Full 3D model
  - Probabilistic
  - Inherently multi-resolution
  - Memory efficient
- Contra:
  - Implementation can be tricky (memory, update, map files, …)

OctoMap Framework

- Based on octrees
- Probabilistic, volumetric representation of occupancy including unknown
- Supports multi-resolution map queries
- Memory efficient
- Compact map files
- Open source implementation as C++ library available at http://octomap.sf.net

Probabilistic Map Update

- Occupancy modeled as recursive binary Bayes filter [Moravec '85]

$$Bel(m_t^{[xyz]}) = \left[1 + \frac{1 - P(m_t^{[xyz]} | z_t, u_{t-1})}{P(m_t^{[xyz]} | z_t, u_{t-1})} \cdot \frac{P(m_t^{[xyz]})}{1 - P(m_{t-1}^{[xyz]})} \frac{1 - Bel(m_{t-1}^{[xyz]})}{Bel(m_t^{[xyz]})} \right]^{-1}$$

- Efficient update using log-odds notation

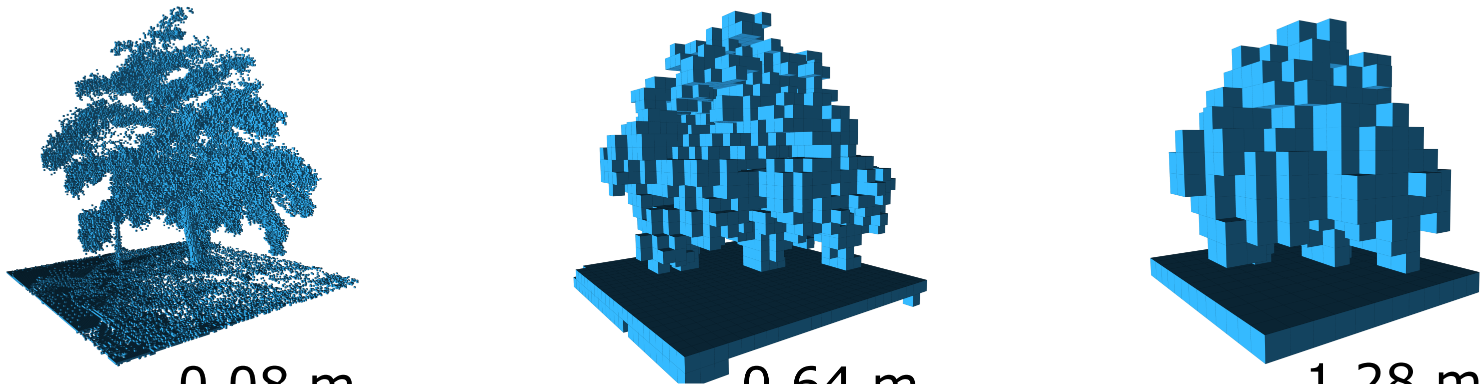
Probabilistic Map Update

- Clamping policy ensures updatability [Yguel '07]

$$Bel(m_t^{[xyz]}) \in [l_{\text{min}}, l_{\text{max}}]$$

- Multi-resolution queries using

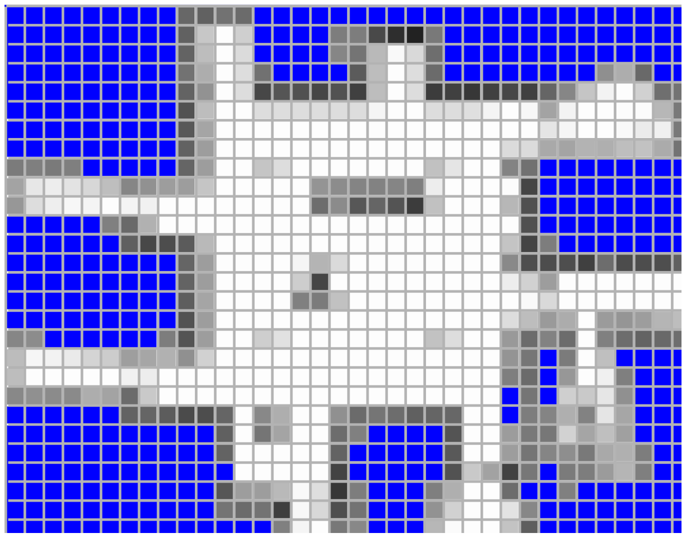
$$Bel(n) = \max_{i=1\ldots8} Bel(n_i), n_i \in \text{children}(n)$$

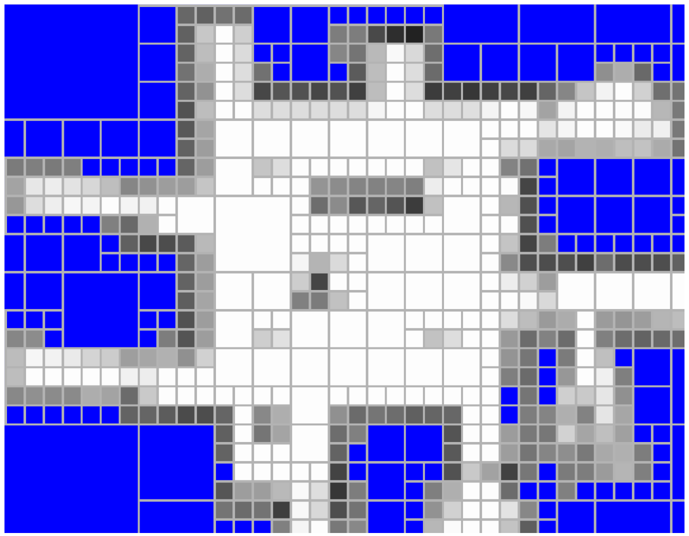


0.08 m 0.64 m 1.28 m

Lossless Map Compression

- Lossless pruning of nodes with identical children
- Can lead to high compression ratios

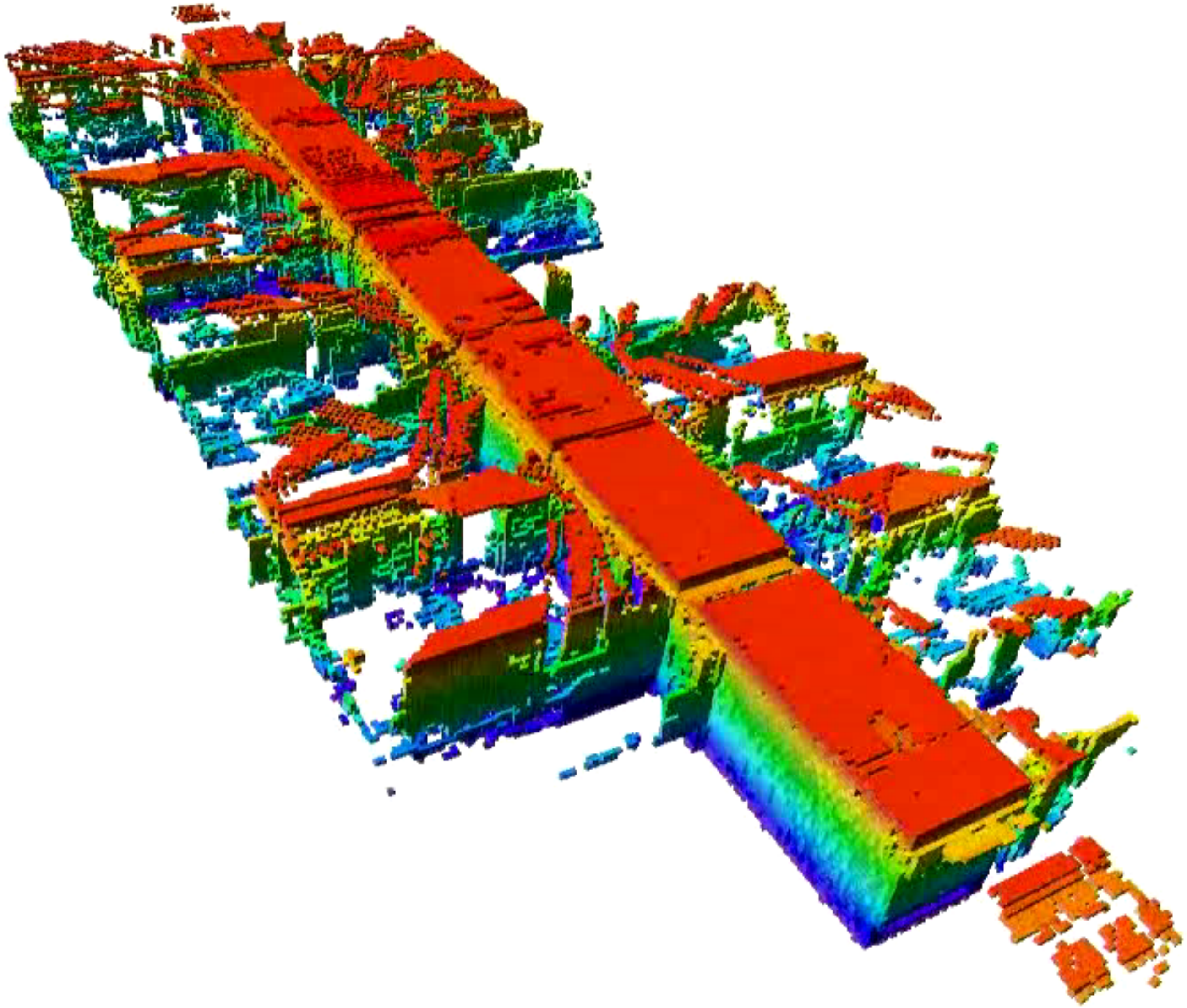




[Kraetzschmar '04]

Video: Office Building

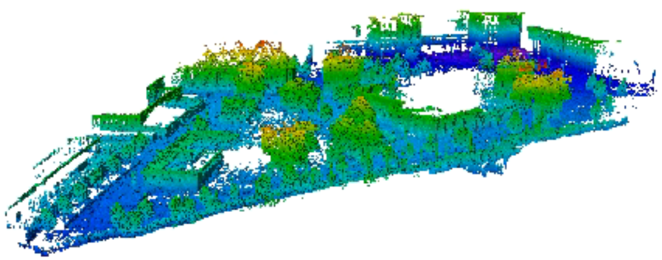
Freiburg, building 079



Video: Large Outdoor Areas

Freiburg computer science campus

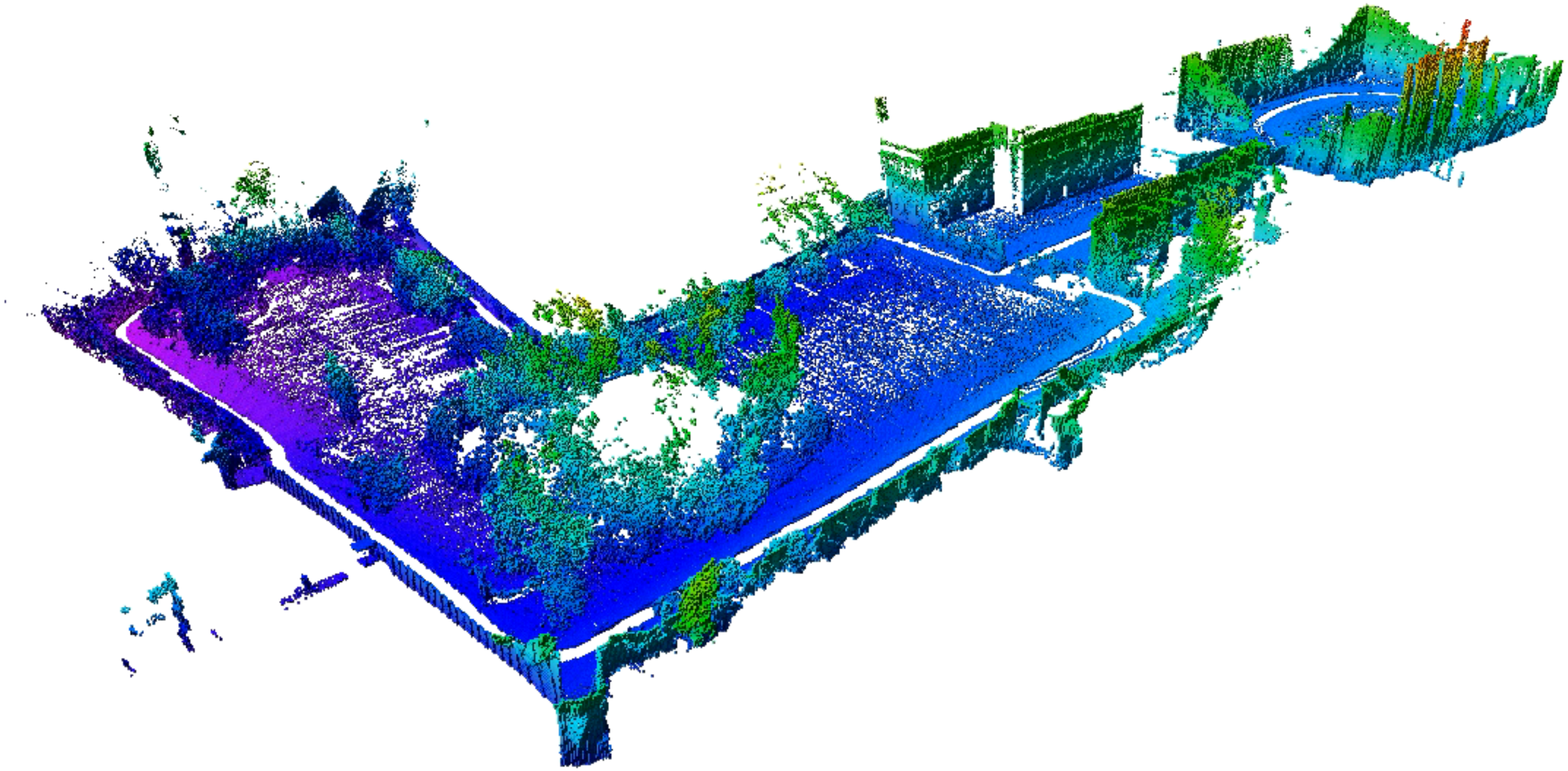
(292 x 167 x 28 m^3, 20 cm resolution)



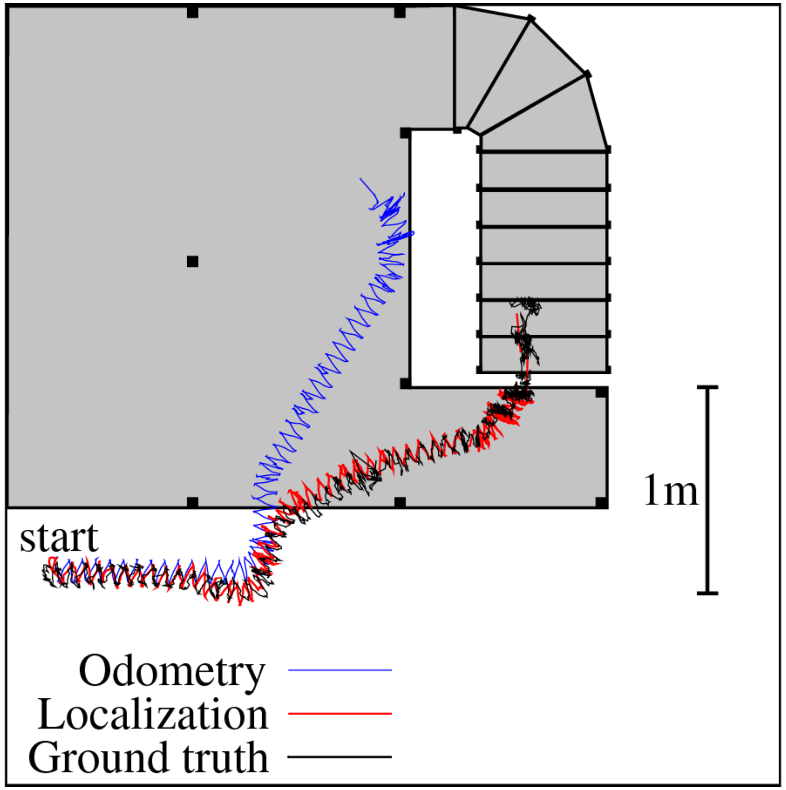
Video: Large Outdoor Areas

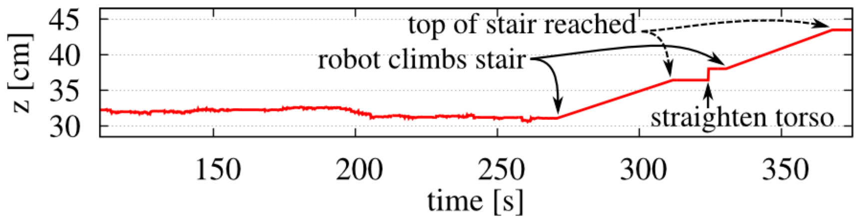
- Oxford *New College* dataset (Epoch C)

  (250 x 161 x 33 m³, 20 cm resolution)



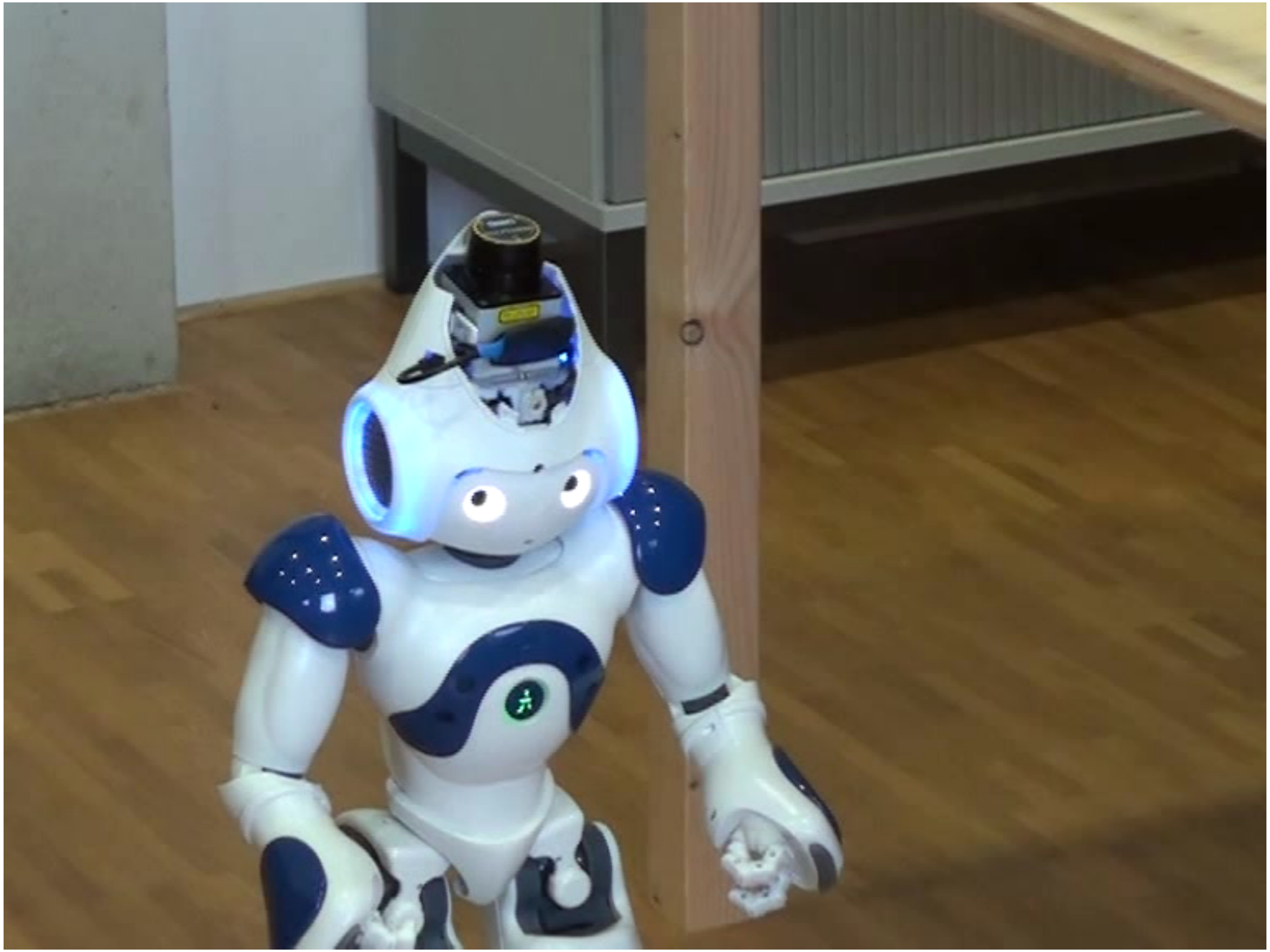
6D Localization with a Humanoid



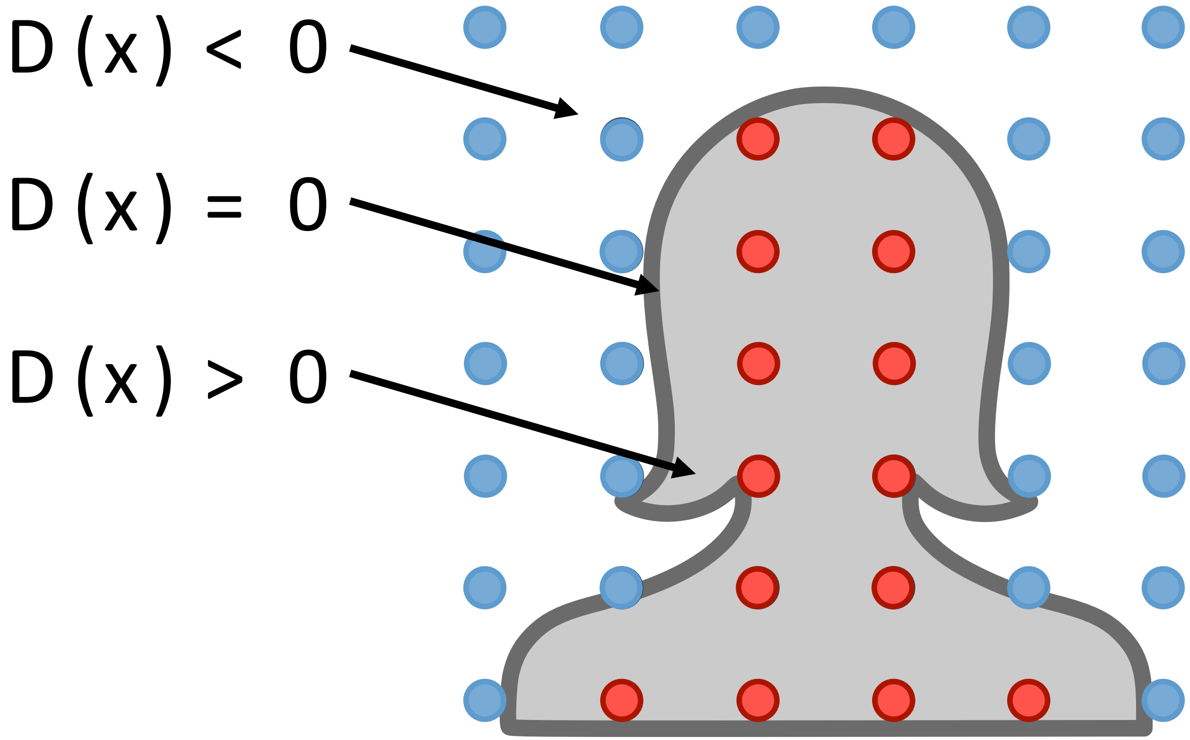


Goal: Accurate pose tracking while walking and climbing stairs

Video: Humanoid Localization



Signed Distance Function (SDF)



Negative signed distance (=outside)

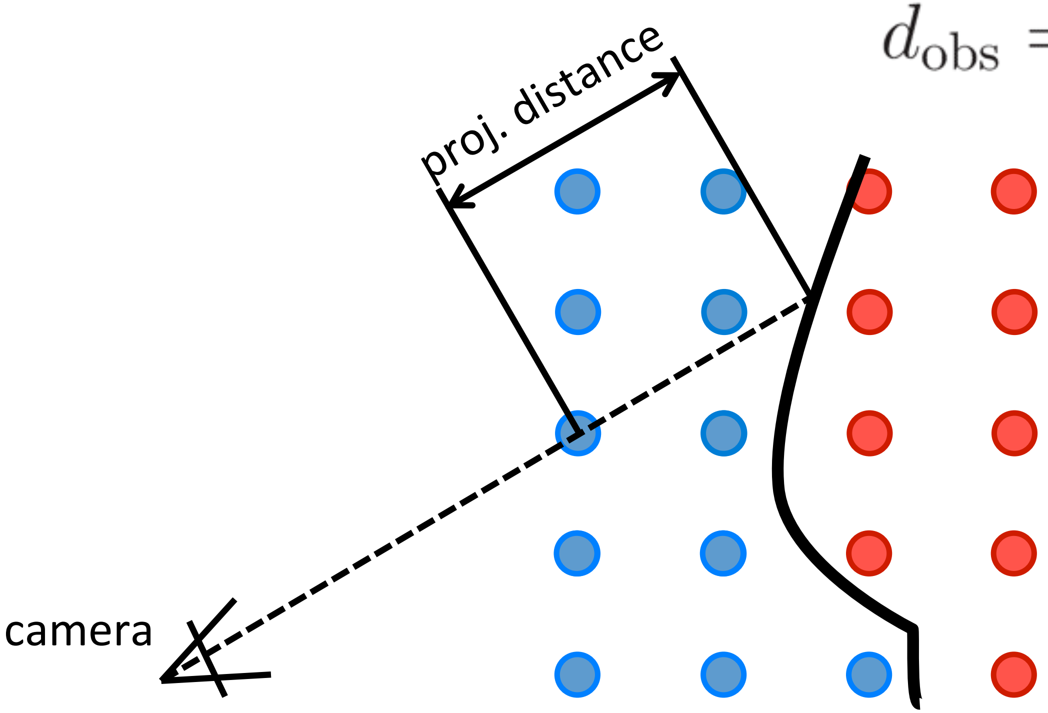
Positive signed distance (=inside)

begin slides courtesy of Jürgen Sturm]

Signed Distance Function (SDF)

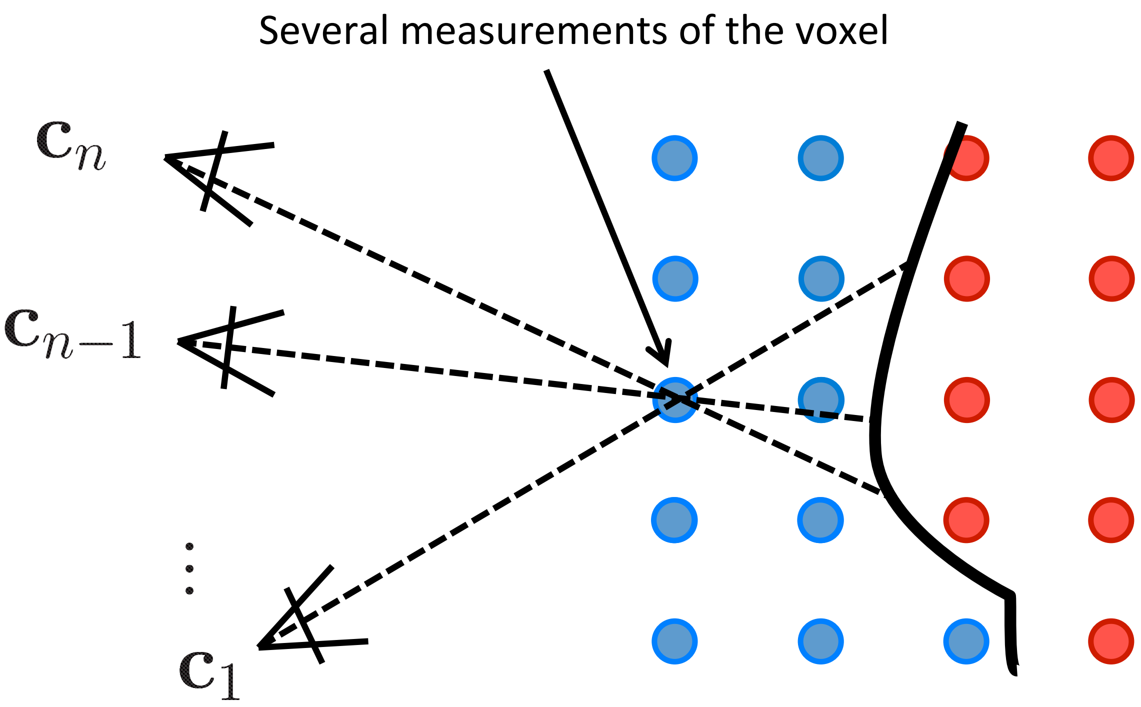
- Compute SDF from a depth image
- Measure distance of each voxel to the observed surface
- Can be done in parallel for all voxels (→ GPU)
- Becomes very efficient by only considering a small interval around the endpoint (truncation)

$$d_{\text{obs}} = z - I_Z(\pi(x, y, z))$$



Signed Distance Function (SDF)

- Calculate weighted average over all measurements for every voxel
- Assume known camera poses



$$D \leftarrow \frac{WD + wd}{W + w}$$

$$W \leftarrow W + w$$

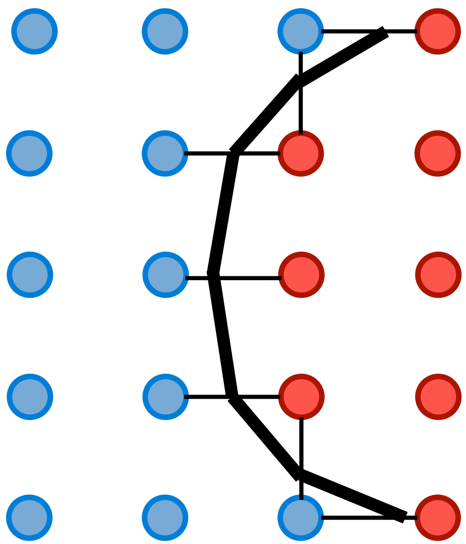
Visualizing Signed Distance Fields

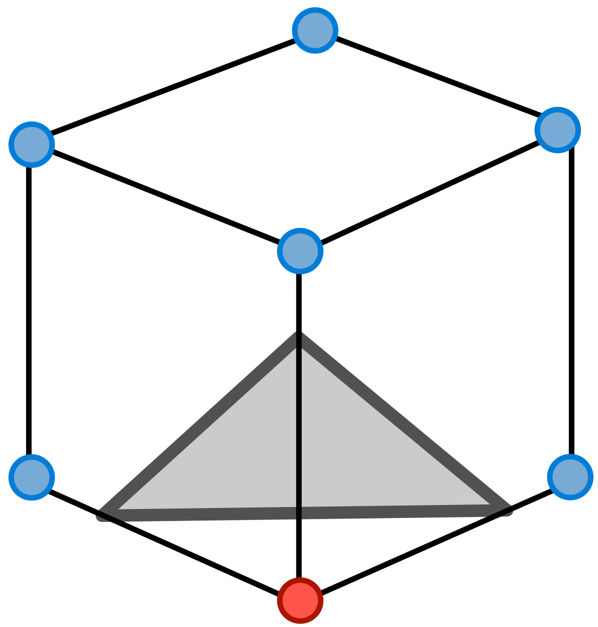
Common approaches to iso surface extraction:

1. Ray casting (GPU, fast)
   For each camera pixel, shoot a ray and search for zero crossing
2. Poligonization (CPU, slow)
   E.g., using the marching cubes algorithm
   Advantage: outputs triangle mesh

Mesh Extraction using Marching Cubes

- Find zero-crossings in the signed distance function by interpolation

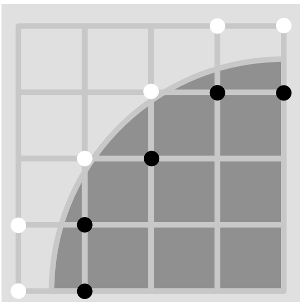


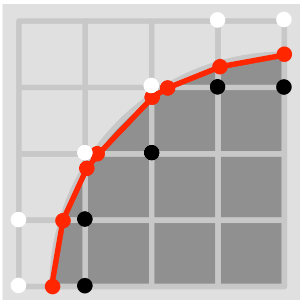


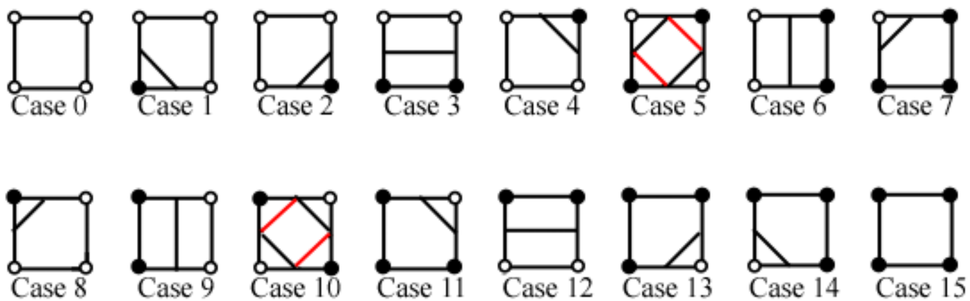
Marching Cubes

If we are in 2D: **Marching squares**

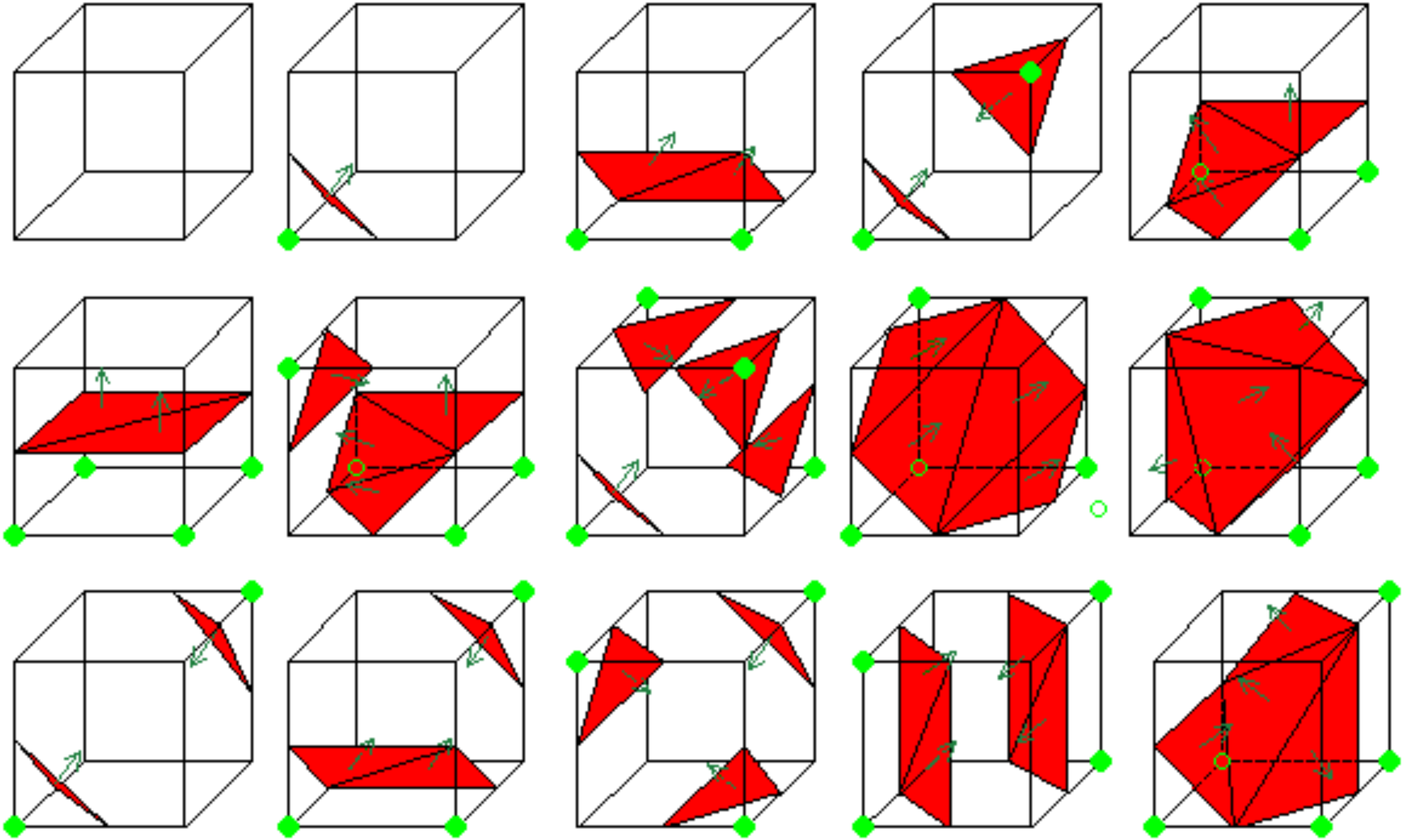
- Evaluate each cell separately
- Check which edges are inside/outside
- Generate triangles according to 16 lookup tables
- Locate vertices using least squares







Marching Cubes (3D)



KinectFusion

- SLAM based on projective ICP (see next section) with point-to-plane metric
- Truncated signed distance function (TSDF)
- Ray Casting

An Application

[Sturm, Bylow, Kahl, Cremers; GCPR 2013], end courtesy by Jürgen Sturm]

Signed Distance Functions

- Pro:
  - Full 3D model
  - Sup-pixel accuracy
  - Fast (graphics card) implementation

- Contra:
  - Space consuming voxel grid

Summary

- Different 3D map representations exist
- The best model always depends upon the corresponding application
- We discussed surface models and voxel representations
- Surface models support a traversability analysis
- Voxel representations allow for a full 3D representation
- Octrees are a probabilistic representation. They are inherently multi-resolution.
- Signed distance functions also use three-dimensional grids but allow for a sub-pixel accuracy representation of the surface.
- Note: there also is a PointCloud Library for directly dealing with point clouds (see also next chapter).