Visual Navigation for Flying Robots

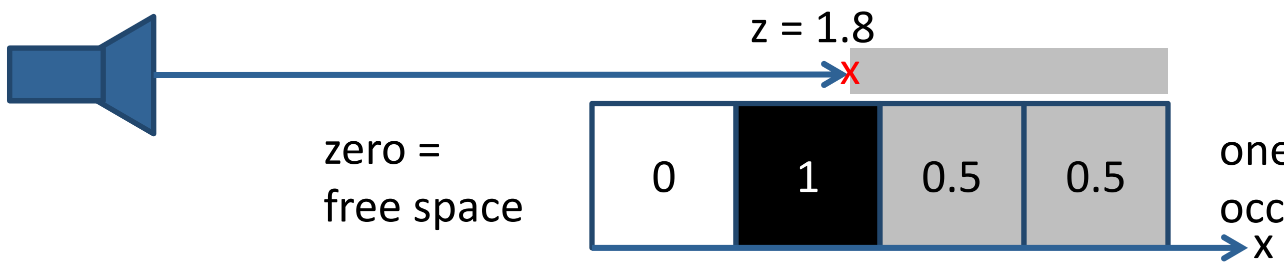
Dense Reconstruction in Sparse Data Structures

Frank Steinbrücker

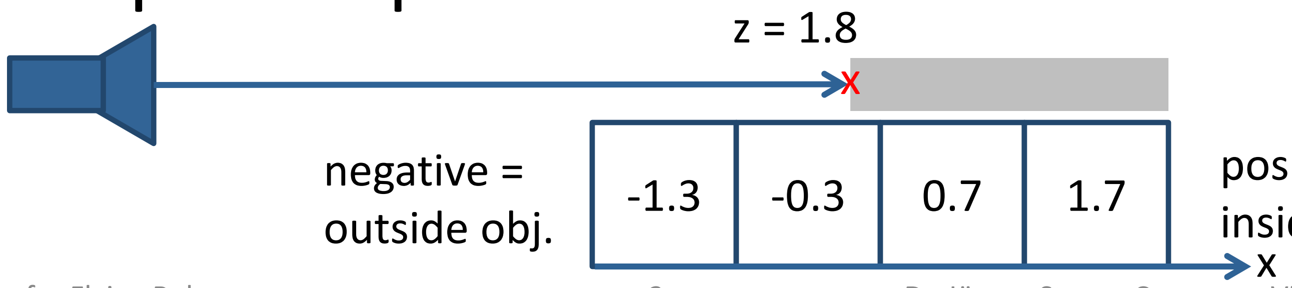
Recap: Signed Distance Field (SDF)

[Curless and Levoy, 1996]

- **Idea:** Instead of representing the cell occupancy, represent the distance of each cell to the surface
- Occupancy grid maps: explicit representation



- SDF: implicit representation

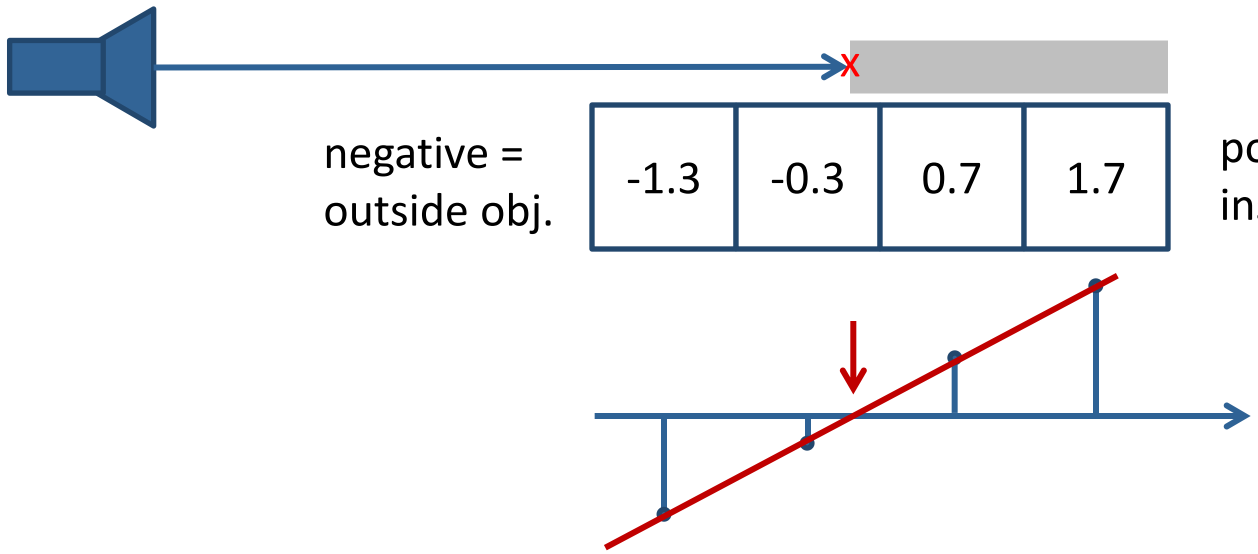


Recap: Signed Distance Field (SDF)

[Curless and Levoy, 1996]

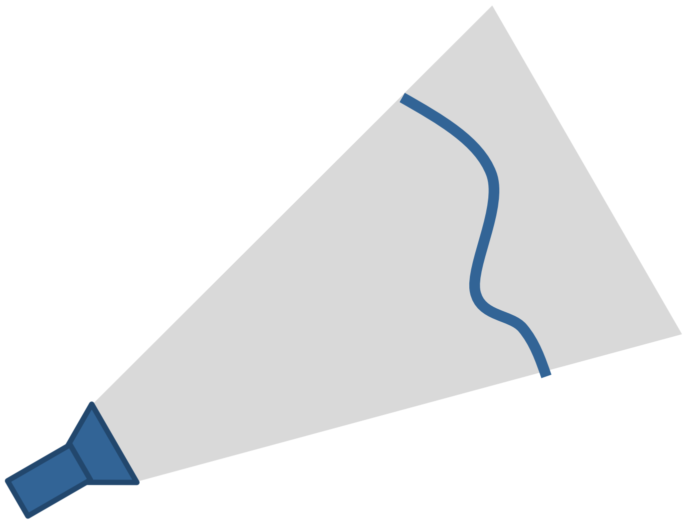
Algorithm:

1. Estimate the signed distance field
2. Extract the surface using interpolation (surface is located at zero-crossing)



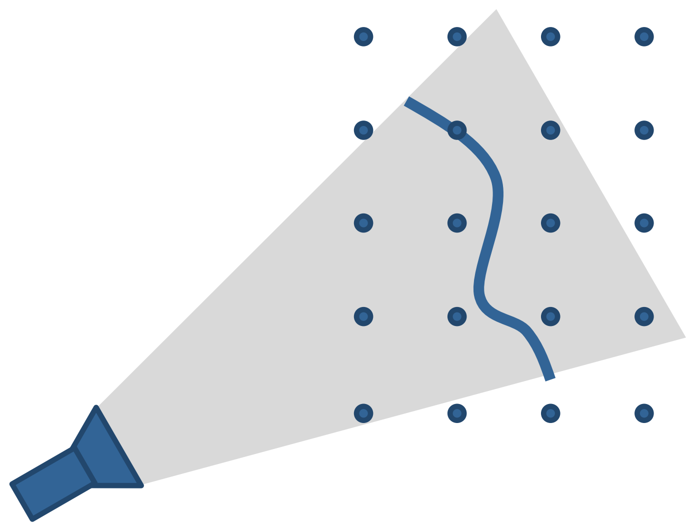
Recap: Dense Mapping: 2D Example

- Camera with known pose



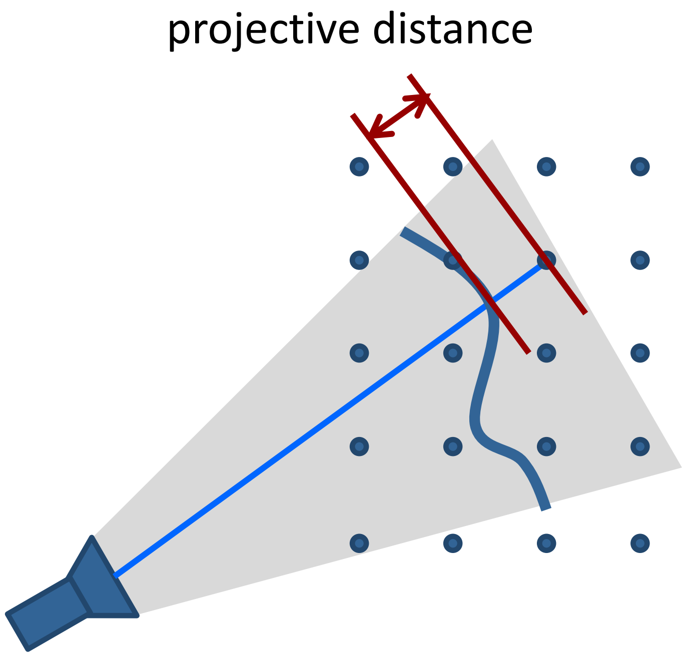
Recap: Dense Mapping: 2D Example

- Camera with known pose
- Grid with signed distance function



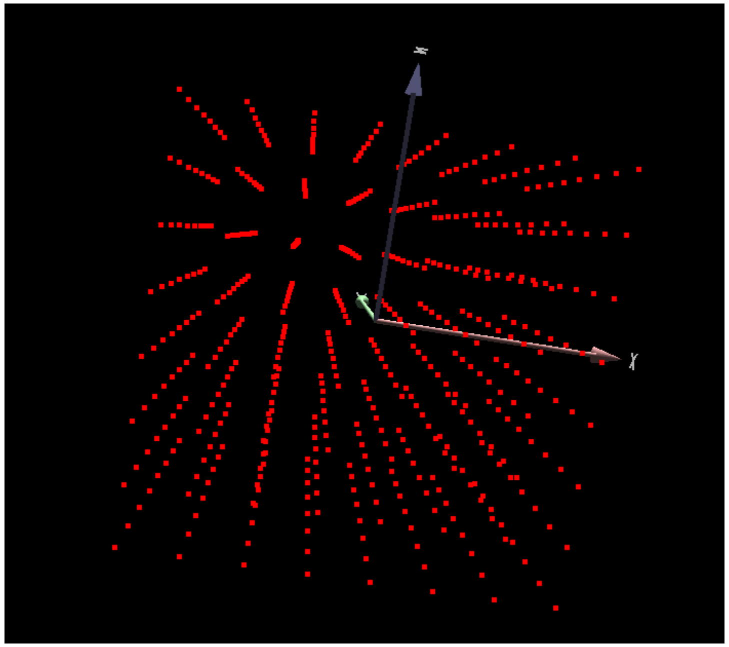
Recap: Dense Mapping: 2D Example

- For each grid cell, compute its projective distance to the surface



Recap: Dense Mapping: 3D Example

- Generalizes directly to 3D
- But: memory usage is cubic in side length



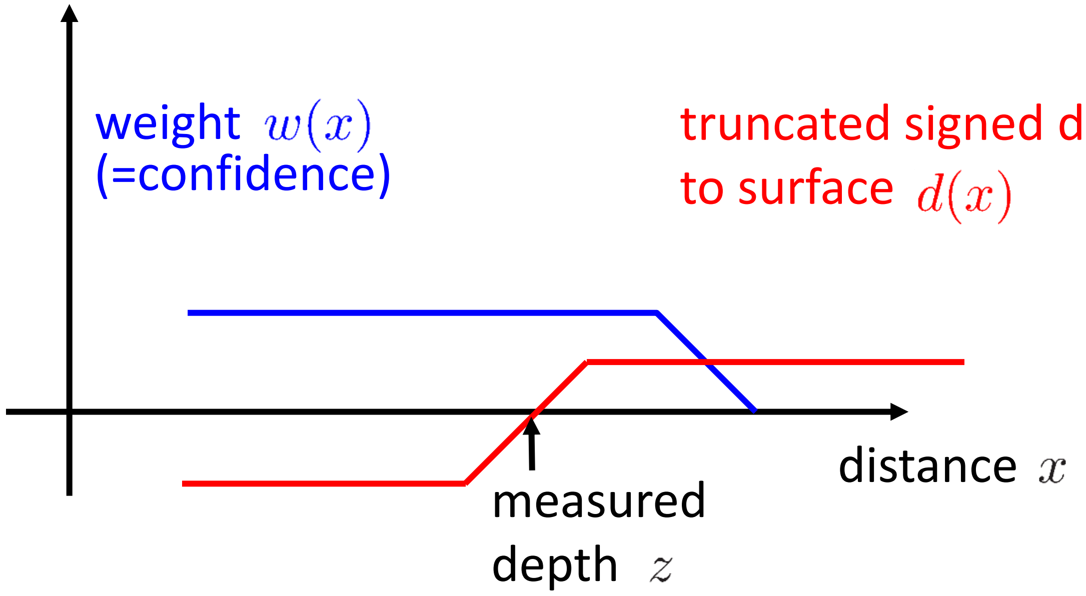
Recap: Data Fusion

- **Idea:** Compute weighted average
- Each voxel cell x in the SDF stores two values
  - Weighted sum of signed distances $D_t(\mathbf{x})$
  - Sum of all weights $W_t(\mathbf{x})$
- When new range image arrives, update every voxel cell according to

$$D_{t+1}(\mathbf{x}) = D_t(\mathbf{x}) + w_{t+1}(\mathbf{x}) d_{t+1}(\mathbf{x})$$
$$W_{t+1}(\mathbf{x}) = W_t(\mathbf{x}) + w_{t+1}(\mathbf{x})$$

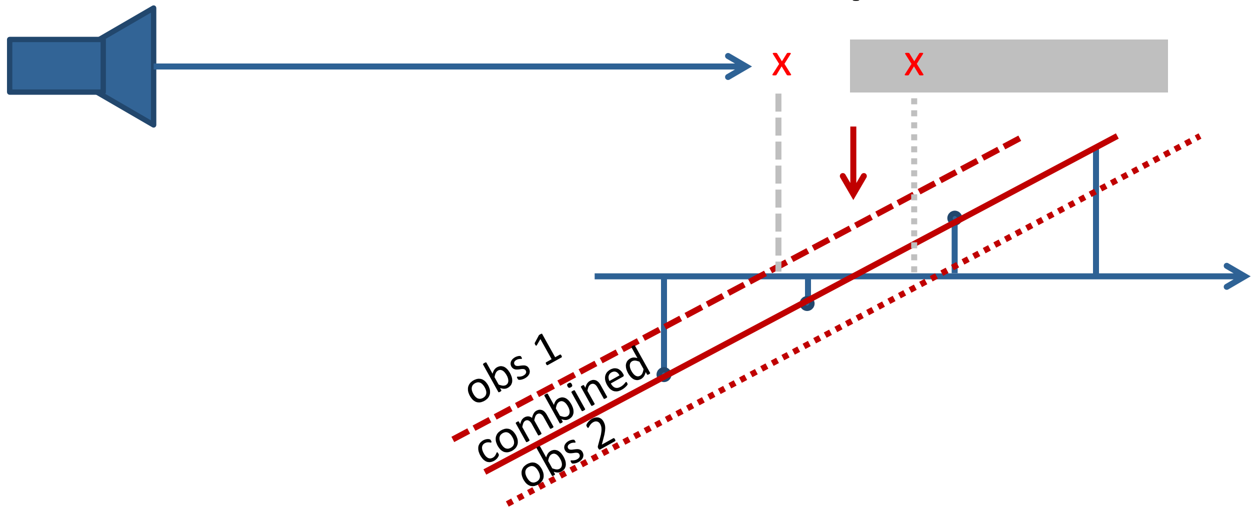
Recap: Distance and Weighting Functions

- Weight each observation according to its confidence
- Weight can additionally be influenced by other modalities (reflectance values, …)



Recap: Two Nice Properties

- Noise cancels out over multiple measurements



- Zero-crossing can be extracted at sub-voxel accuracy (least squares estimate)

1D Example: $$x^* = \frac{\sum D_t(x)x}{\sum W_t(x)x}$$

Recap: Memory Consumption

- Consider we want to map a building with 40x40m at a resolution of 0.05m
- How much memory do we need?

$$\left(\frac{40}{0.05}\right)^2 = 640.000 \text{ cells} = 4.88\text{mb}$$

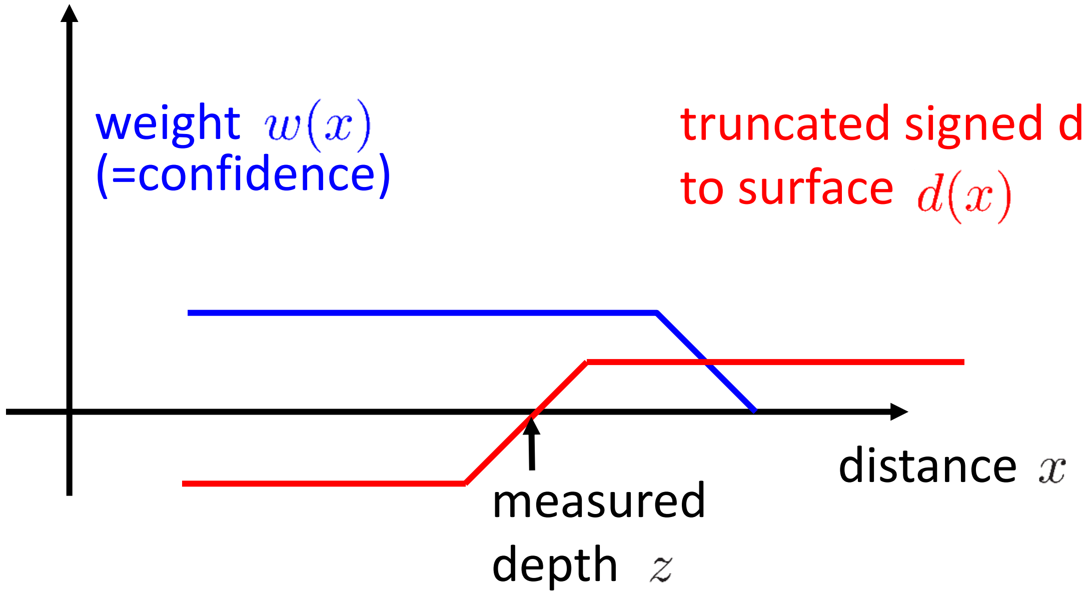
- And for 3D?

$$\left(\frac{40}{0.05}\right)^3 = 512.000.000 \text{ cells} = 3.8\text{gb}$$

- And what about a whole city?

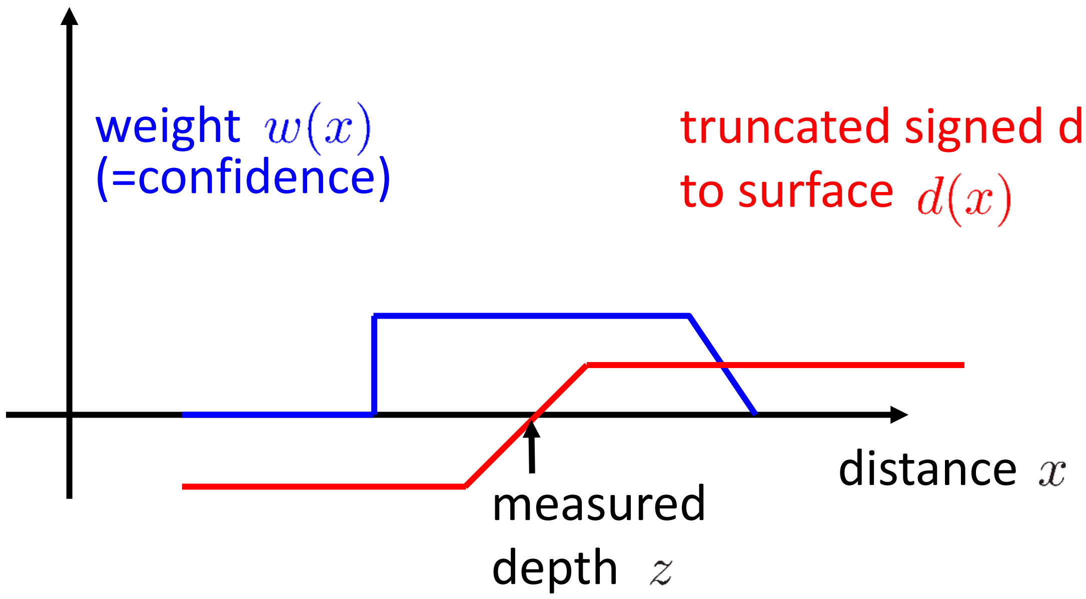
Distance and Weighting Functions

- Weight each observation according to its confidence
- Weight can additionally be influenced by other modalities (reflectance values, …)



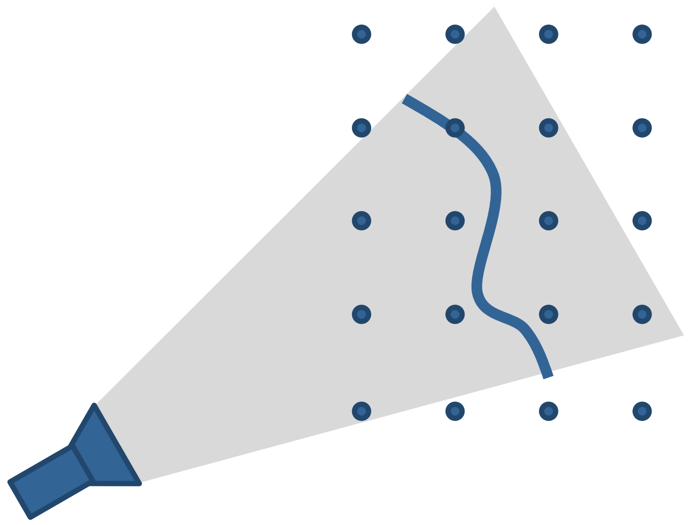
Distance and Weighting Functions

- Values outside the support of the weight function do not need to be stored


weight $w(x)$ (=confidence)
truncated signed distance to surface $d(x)$
distance x
measured depth z

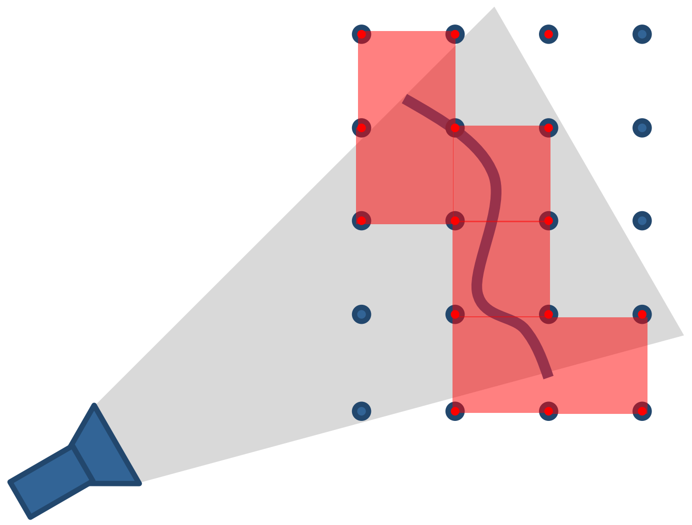
Reconstruction Band around the Surface

- Values outside the support of the weight function do not need to be stored



Reconstruction Band around the Surface

- Values outside the support of the weight function do not need to be stored
- => Hierarchical structure, voxels grouped in "bricks"



Reconstruction Band around the Surface

- Requirements
  - Low storage cost
  - Time-Efficient update of the band
  - Online-Capability, i.e. not all data is present beforehand
    - Volume “grows” when more depth images are added
    - Only bricks in the current camera frustum are touched

Structure 1: Brick List/Array

- Every brick stores a position in space
  - Low storage cost => O(1)✓
  - Time-Efficient update of the band => O(n) X
  - Online-Capability, i.e. not all data is present beforehand
    - Volume “grows” when more depth images are added ✓
    - Only bricks in the current camera frustum are touched X

Structure 2: Brick Grid

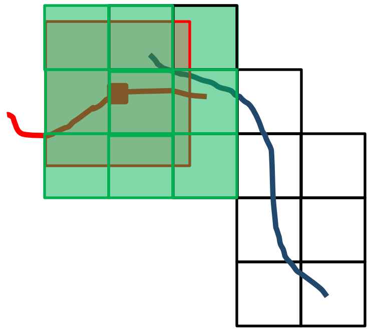
- Coarse Grid stores occupancy of the bricks
  - Low storage cost => O(n)X
  - Time-Efficient update of the band => O(1)✓
  - Online-Capability, i.e. not all data is present beforehand
    - Volume “grows” when more depth images are added X
    - Only bricks in the current camera frustum are touched ✓

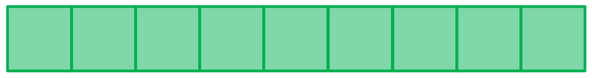
Structure 3: Tree and List

- Brick List/Array with an Octree ontop
  - Low storage cost => O(log n) ✓
  - Time-Efficient update of the band => O(log n)✓
  - Online-Capability, i.e. not all data is present beforehand
    - Volume “grows” when more depth images are added ✓
    - Only bricks in the current camera frustum are touched ✓

Fusion step in the Octree

- (Parallel) iteration over all valid depth pixels
  - Bounding box around the 3D point according to distance/weight-threshold
    - Intersection of the bounding box with tree leaves
      - Allocation of new leaves, where necessary
    - Addition of the the intersected leaves to a queue
  - (Parallel) processing of the queue (projection und distance update)



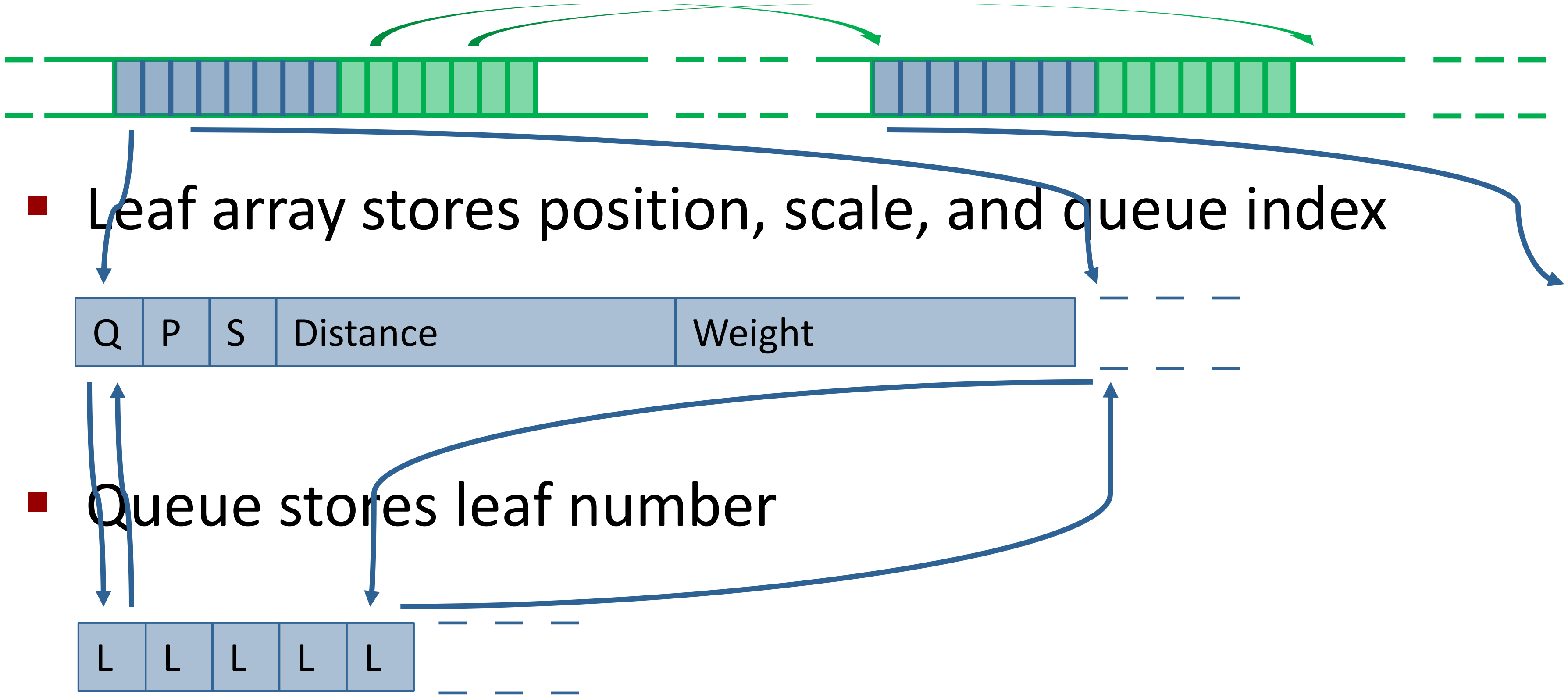


Implementation of the Tree

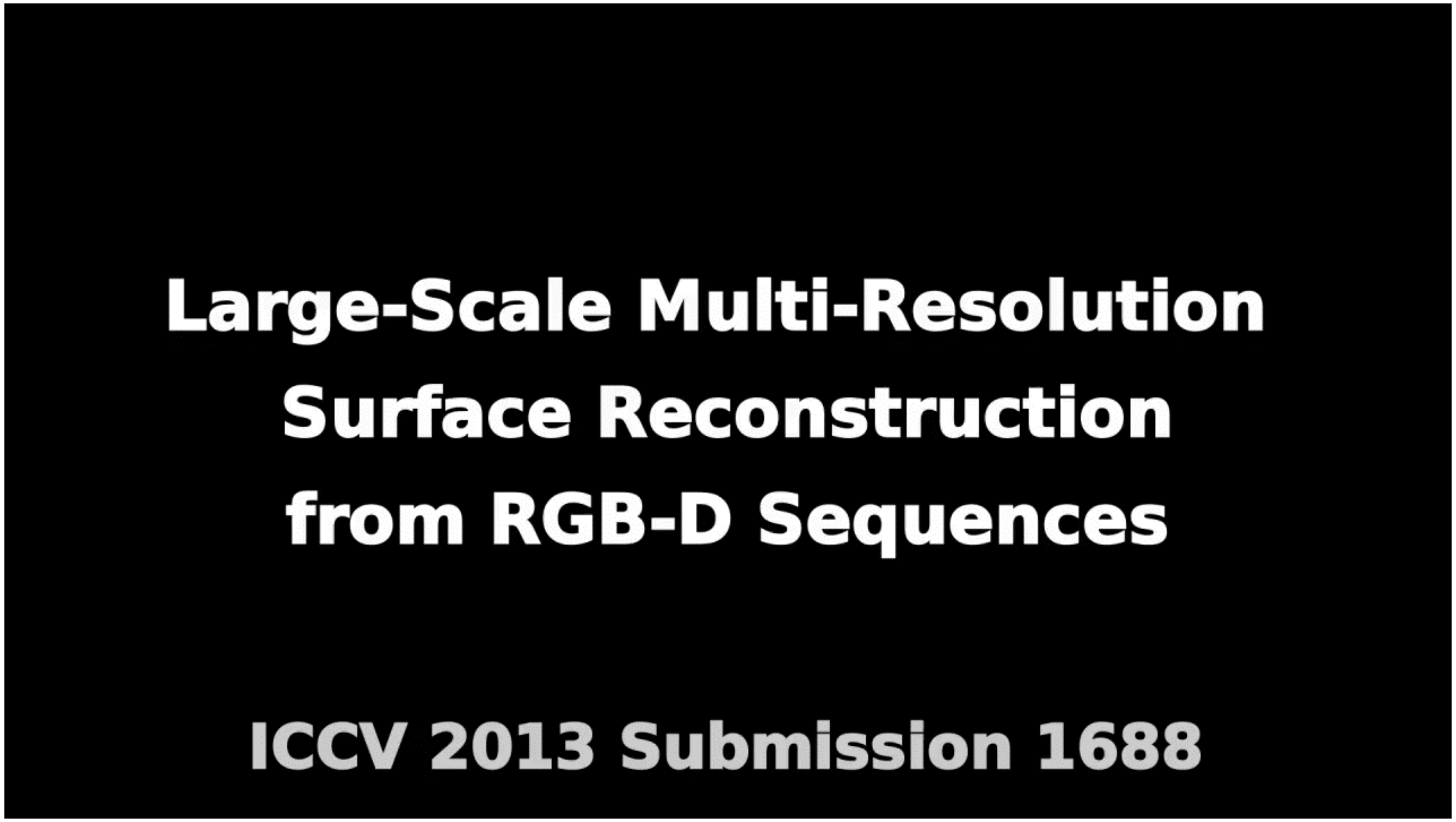
- CPU/GPU interoperability requirements
  - No explicit C-Pointers => Indices instead
  - All memory is allocated beforehand
    - Push-back operations are performed with global indices
  - Minimization of diverging code
    - => Reason for queue

Implementation of the Tree

- Branch array stores indices of leaves and subbranches
- Leaf array stores position, scale, and queue index
- Queue stores leaf number



Demo Video



Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences

ICCV 2013 Submission 1688