Friedrich-Alexander-Universität
Technische Fakultät

Lecture 11, Part 2

Surface Reconstruction 2/2

Computer Vision
Summer Semester 2023

Prof. Bernhard Egger, Prof. Tim Weyrich, Prof. Andreas Maier

shutterstock/zur_gestaltung

Surface Reconstruction – Task Statement

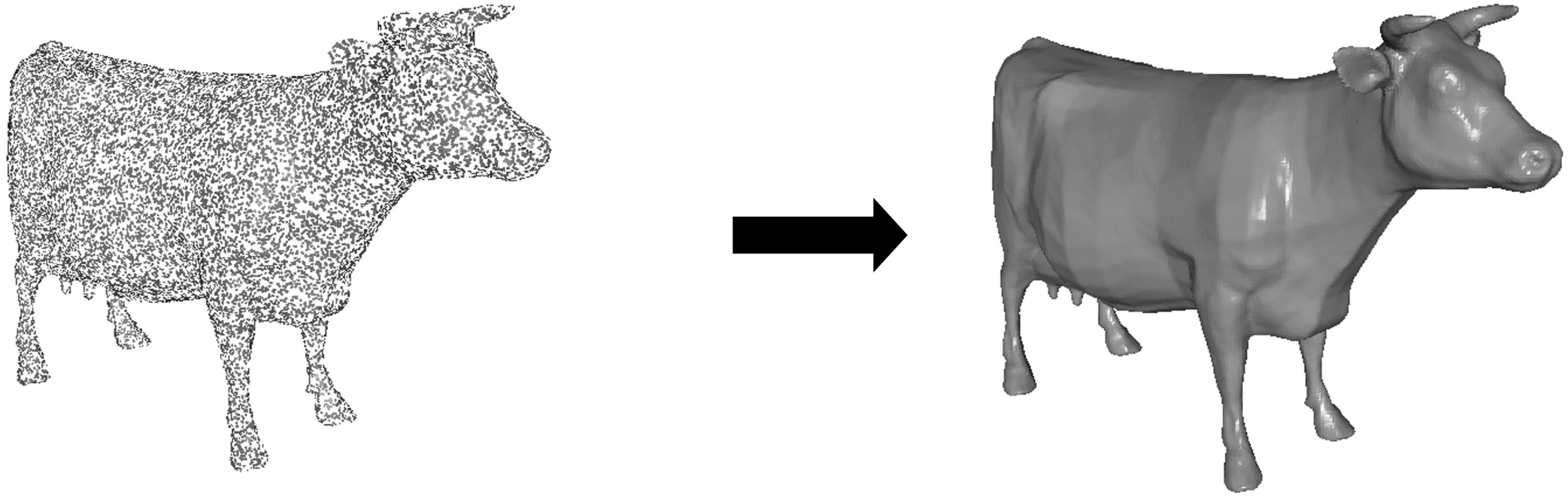
- Given: dense set of 3-D points
  - from multiple scans or
  - from a stream of depth images or
  - reconstructed from multi view stereo
  - or …

- Challenge:
  - reconstruct (closed) surface of the object

- Slides based on
  - Lecture Slides "Geometry Processing", Summer term 2017
  - Hugues Hoppe: "Poisson surface reconstruction"

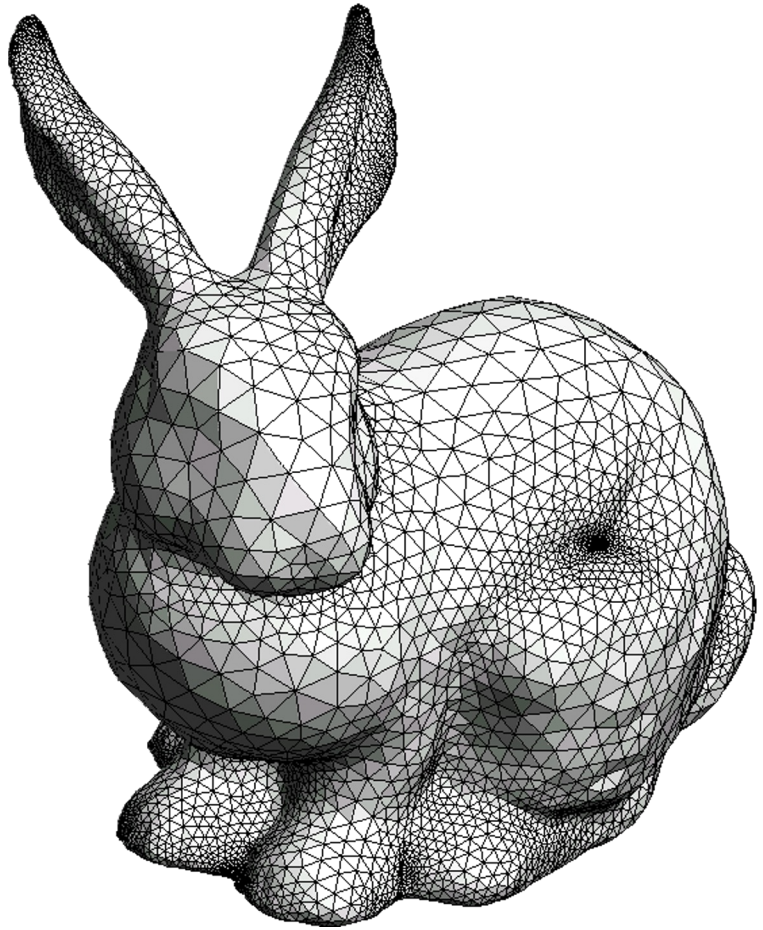
Surface Reconstruction

- Problem #1 solved: We now have registered point clouds
- Problem #2: transform (merged) point clouds to surface



Surface Reconstruction

- Ideally, we want a triangle mesh
- **First approach:**
  - use samples as vertices
  - generate triangles from these

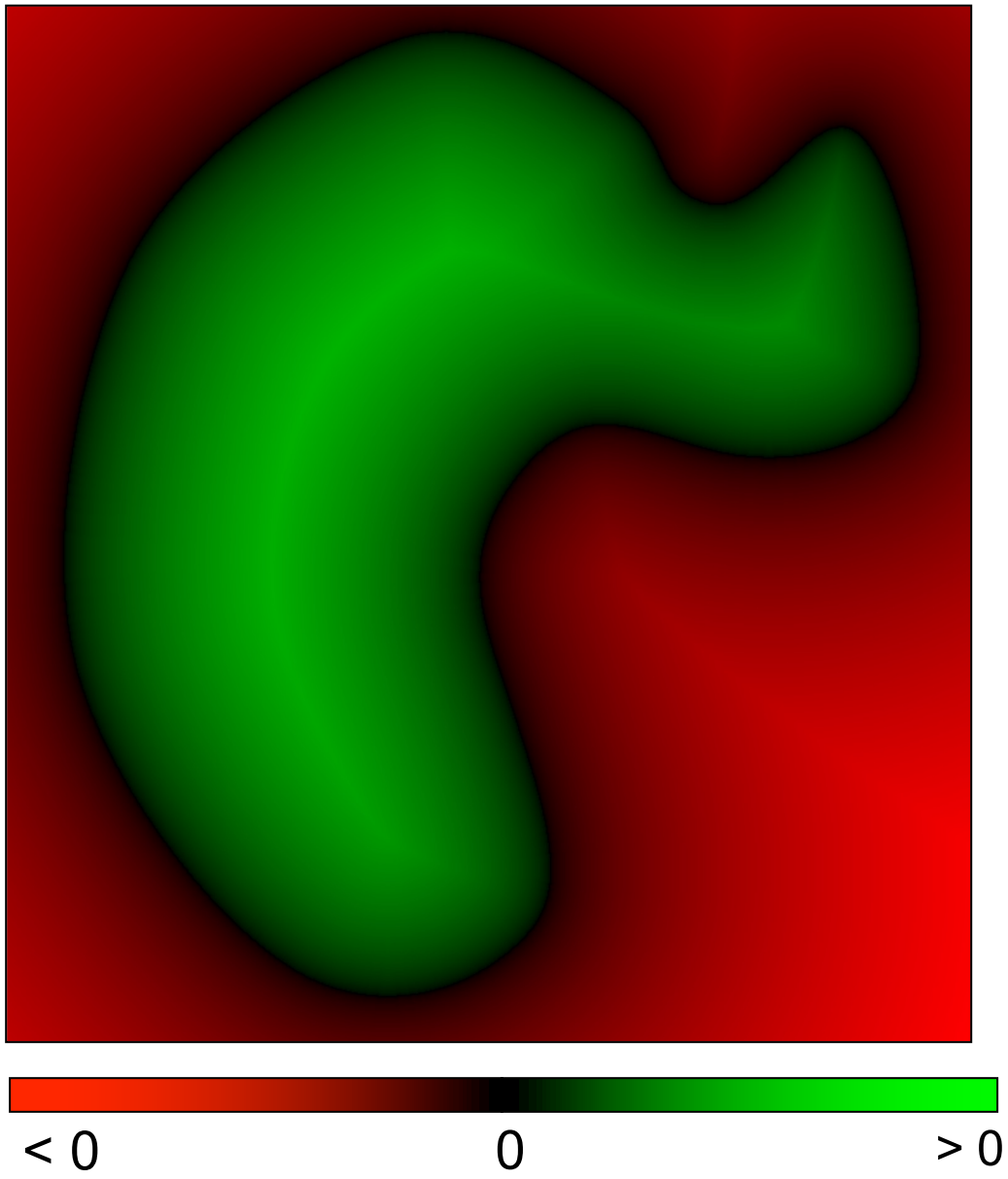


- Interpolates measured samples
  - but very tricky and difficult, e.g. due to noise in the samples
  - Ideally, samples should be *approximated*, not interpolated…

Implicit Functions

- **Second, better approach:** use implicit functions

- Define a function with value less than zero outside the model and greater than zero inside
  → **Implicit Function**

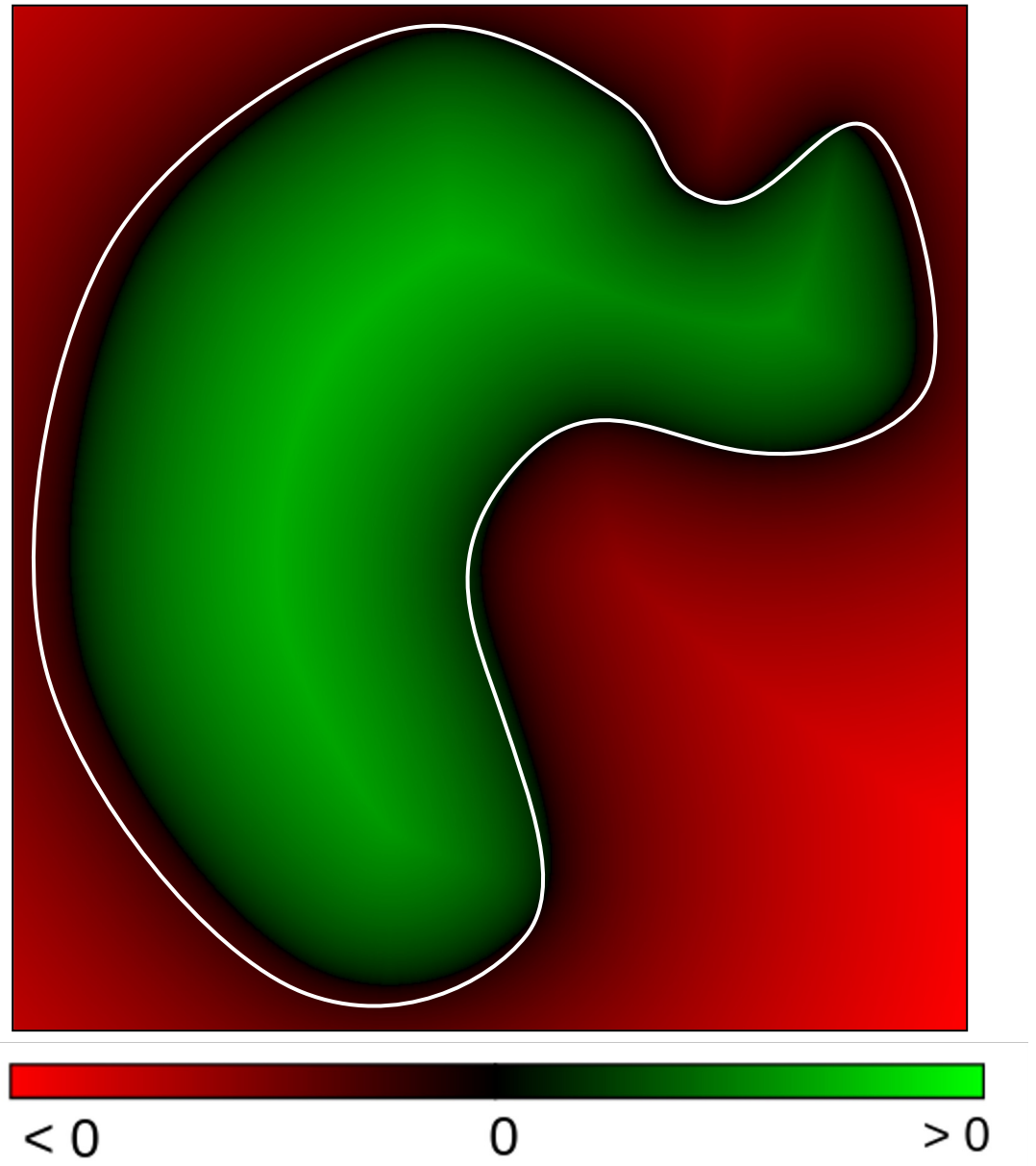


Implicit Functions

- **Second, better approach:** use implicit functions

- Define a function with value less than zero outside the model and greater than zero inside
  → **Implicit Function**

- Extract the zero-set



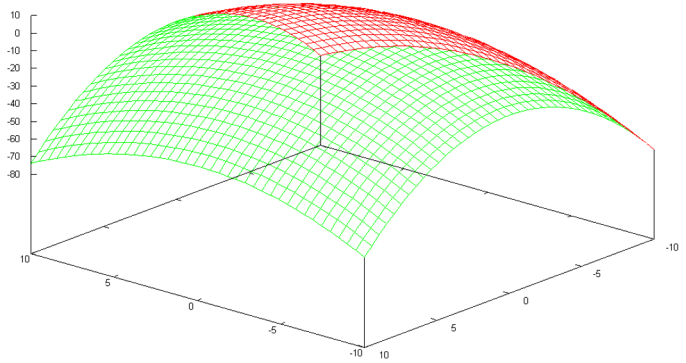
Implicit Functions

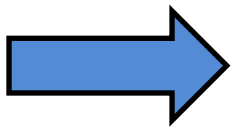
- Extracting the zero-surface of an implicit function is a well-known problem, e.g. also in Visualization
- We first look into this problem for 2D implicit functions describing curves

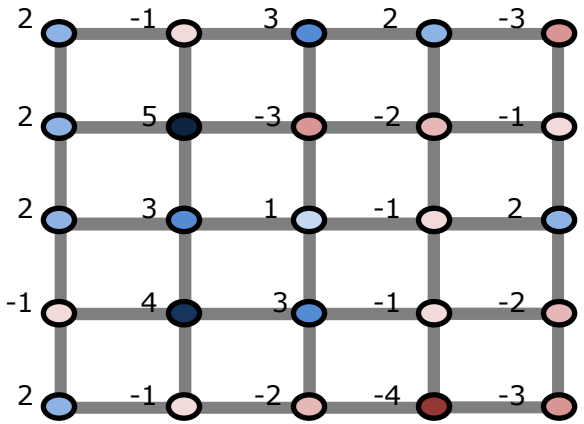
Interlude – Surface Representations

- Converting iso-lines to polygons
  - Given bi-variate scalar function $f(x, y)$: $\mathbb{R}^2 \rightarrow \mathbb{R}$ and a constant c
  - Iso-line is defined by set of points $\{ (x_i, y_i) \mid \ f(x_i, y_i) = c \}$

- Algorithm
  1. Sample function on uniform grid

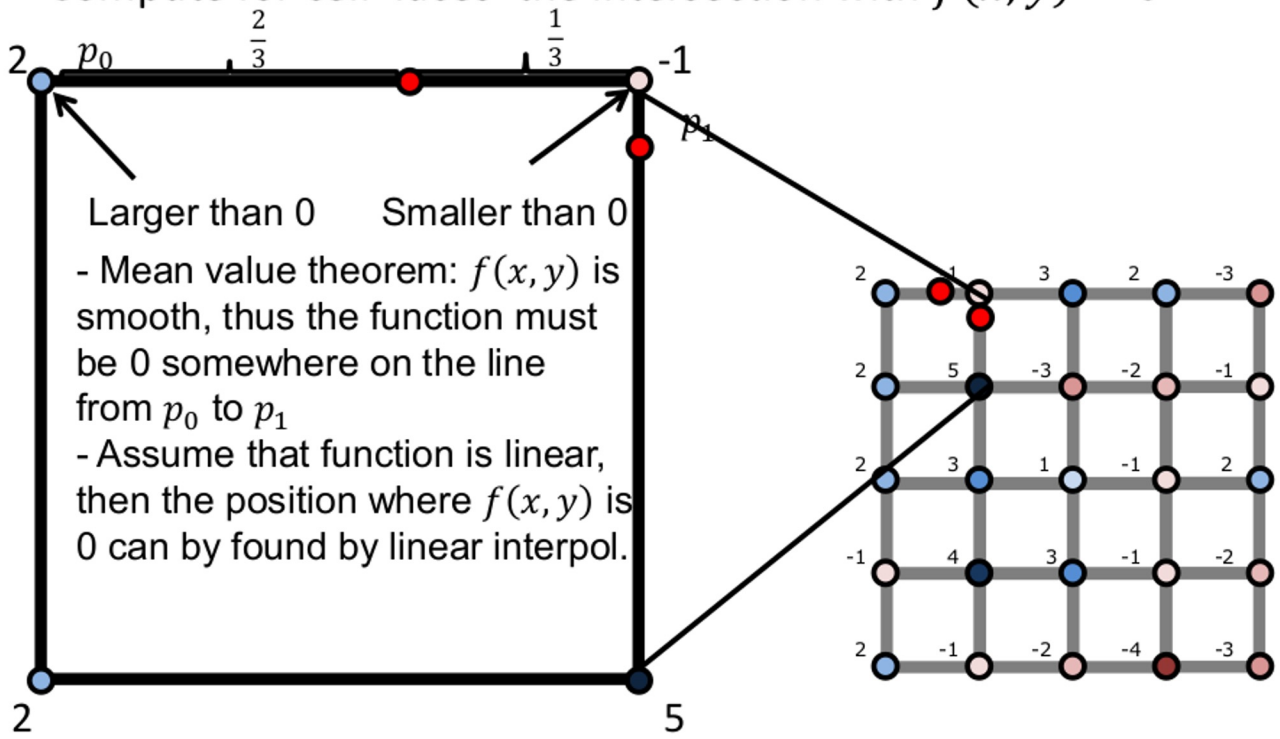






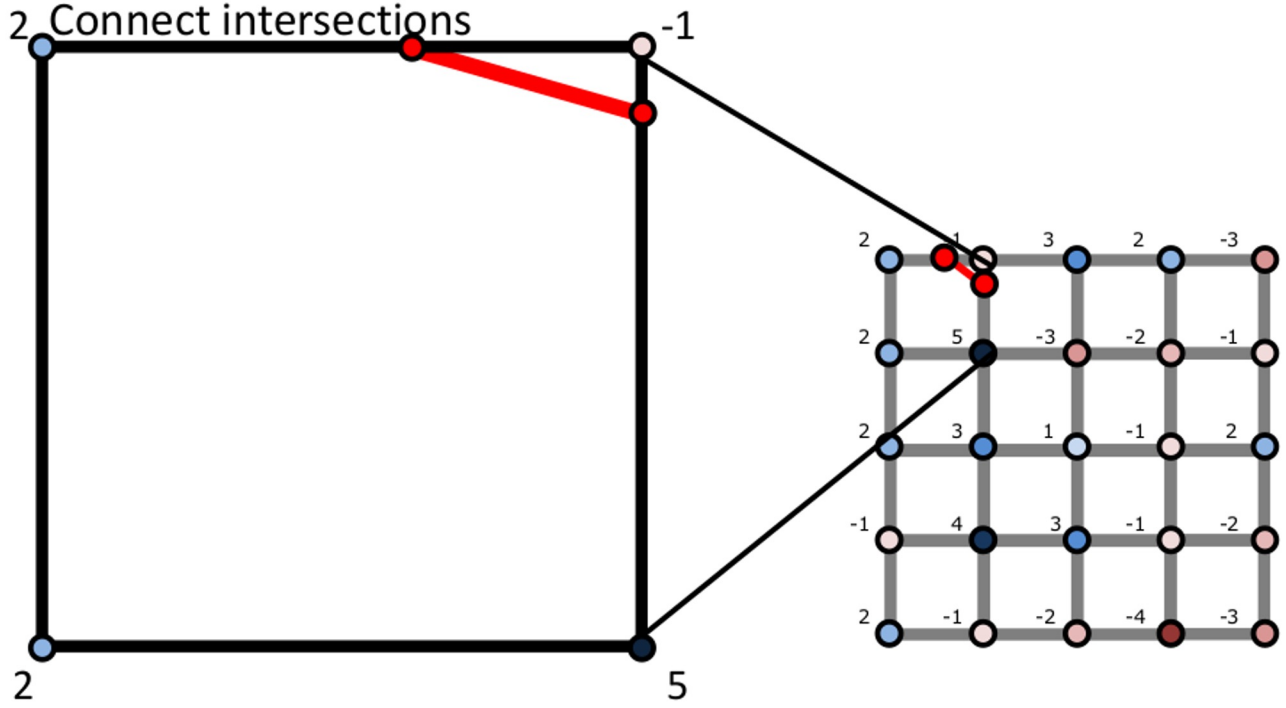
Interlude – Surface Representations

- Algorithm (We look for the $f(x, y) = 0$ iso-line)
  1. Sample function on uniform grid
  2. Check for each cell whether it intersects the iso-line
     - Compute for cell 'faces' the intersection with $f(x, y) = 0$



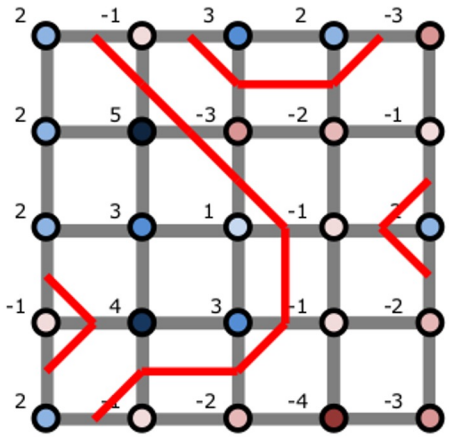
Interlude – Surface Representations

- Algorithm (We look for the $f(x, y) = 0$ iso-line)
  1. Sample function on uniform grid
  2. Check for each cell whether it intersects the iso-line
     - Compute for cell 'faces' the intersection with $f(x, y) = 0$
     - Connect intersections



Interlude – Surface Representations

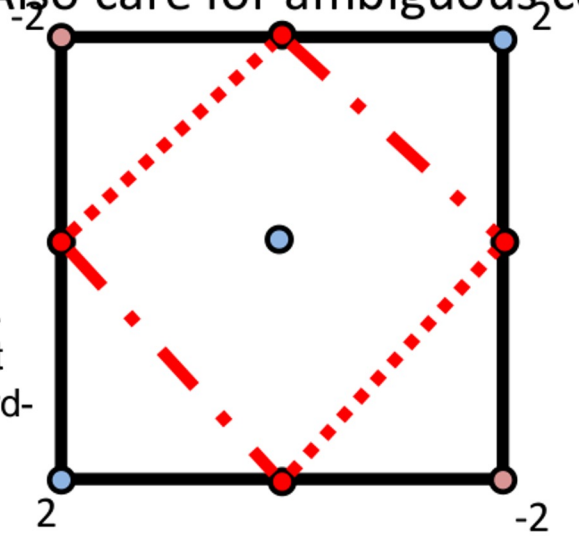
- Algorithm (We look for the $f(x, y) = 0$ iso-line)
  1. Sample function on uniform grid
  2. Check for each cell whether it intersects the iso-line
     - Compute for cell 'faces' the intersection with $f(x, y) = 0$
     - Connect intersections
  3. Repeat for all cells

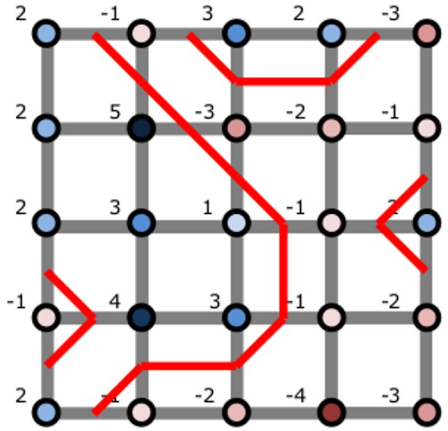


Interlude – Surface Representations

- Algorithm (We look for the $f(x, y) = 0$ iso-line)
  1. Sample function on uniform grid
  2. Check for each cell whether it intersects the iso-line
     - Compute for cell 'faces' the intersection with $f(x, y) = 0$
     - Connect intersections
  3. Repeat for all cells
  4. Also care for ambiguous configurations

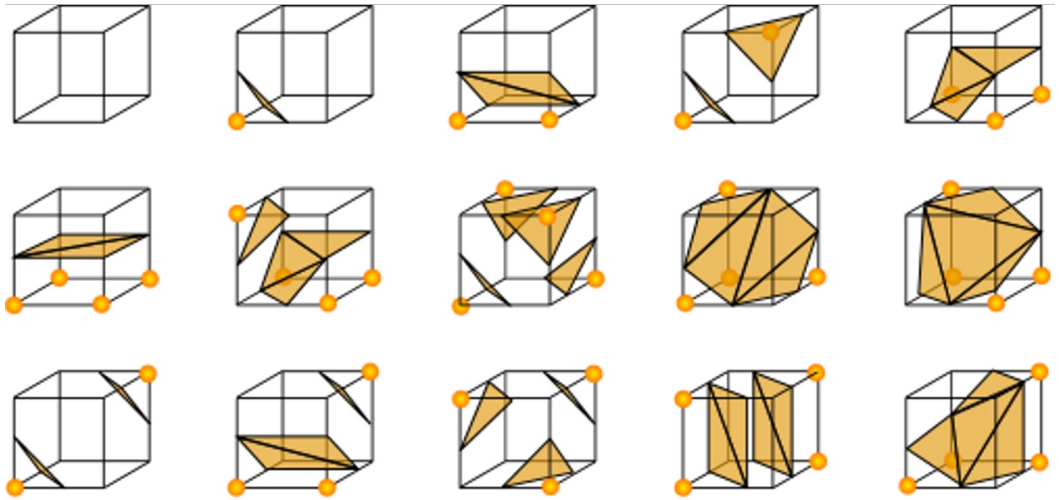
Evaluate func. at center point connect accord-ingly



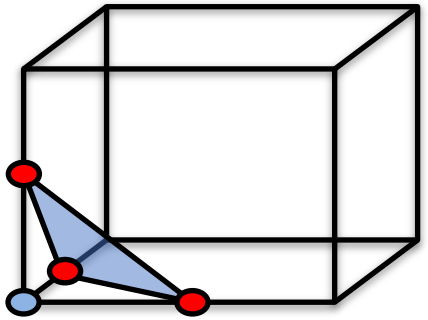


Interlude – Surface Representations

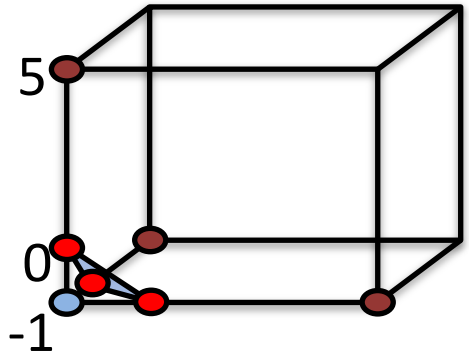
- Algorithm for iso-surfaces (**Marching Cubes**)
  - Sample function on uniform grid
  - For each cell in grid
    - Mark corners whether they are smaller or larger iso-value
    - Cell has 8 vertices, therefore, there are 256 different +/- configurations (due to symmetry, cases may be reduced to 15)
    - Determine correct case, use lookup table to find triangulation
    - Adjust vertex positions according to linear interpolation (5, -1)



Marching Cubes table



From table



Real intersections

Interlude – Surface Representations

- Idea:
  - generate an implicit surface description from the point cloud
  - generate surface from this using Marching Cubes

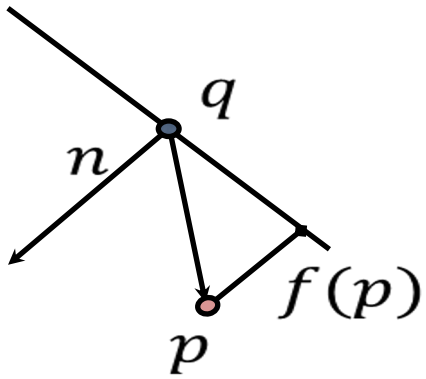
→ **implicit surface reconstruction**

Try – Hoppe's Method

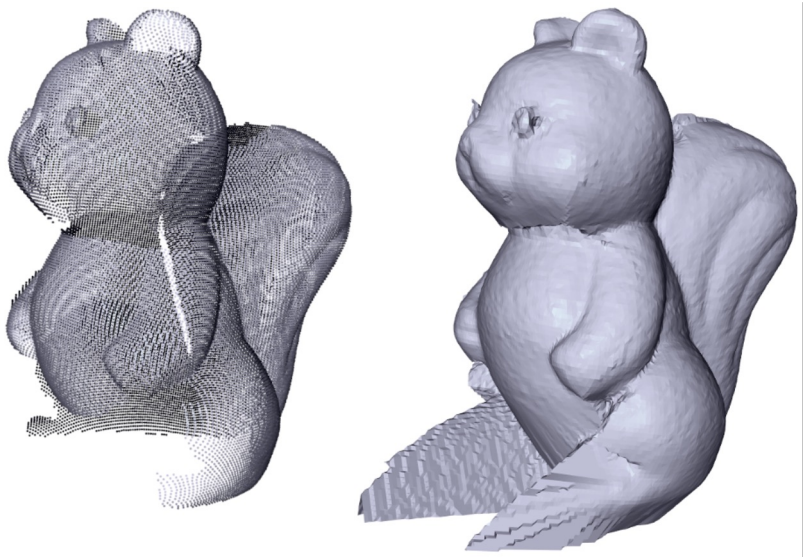
- Hoppe'92 proposed following simple algorithm
  - When evaluating the signed distance function at p
    1. Find closest point q with normal n
    2. Compute distance as distance to tangent plane of q

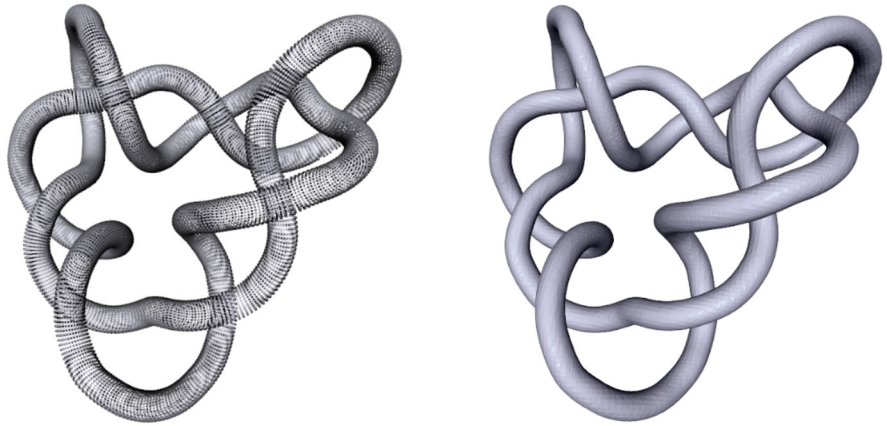
$$f(p) = (p - q) \cdot n$$

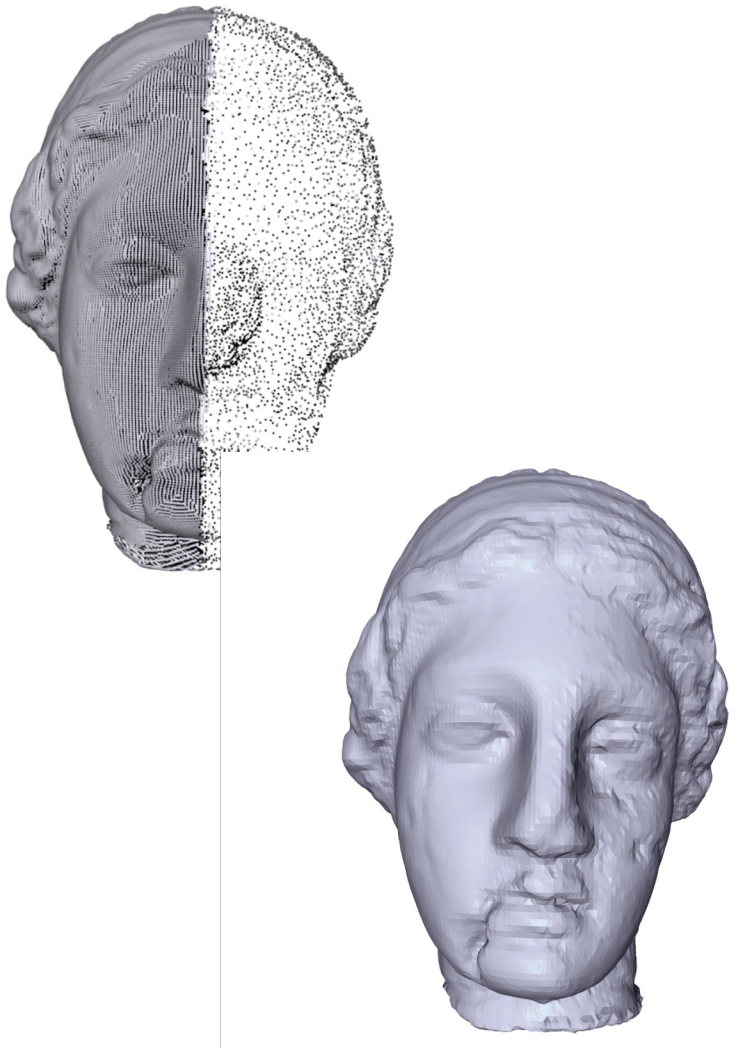
  - Evaluate on uniform grid = volume
  - Run marching cubes on volume to extract $f(x) = 0$



Hoppe's Method – Results



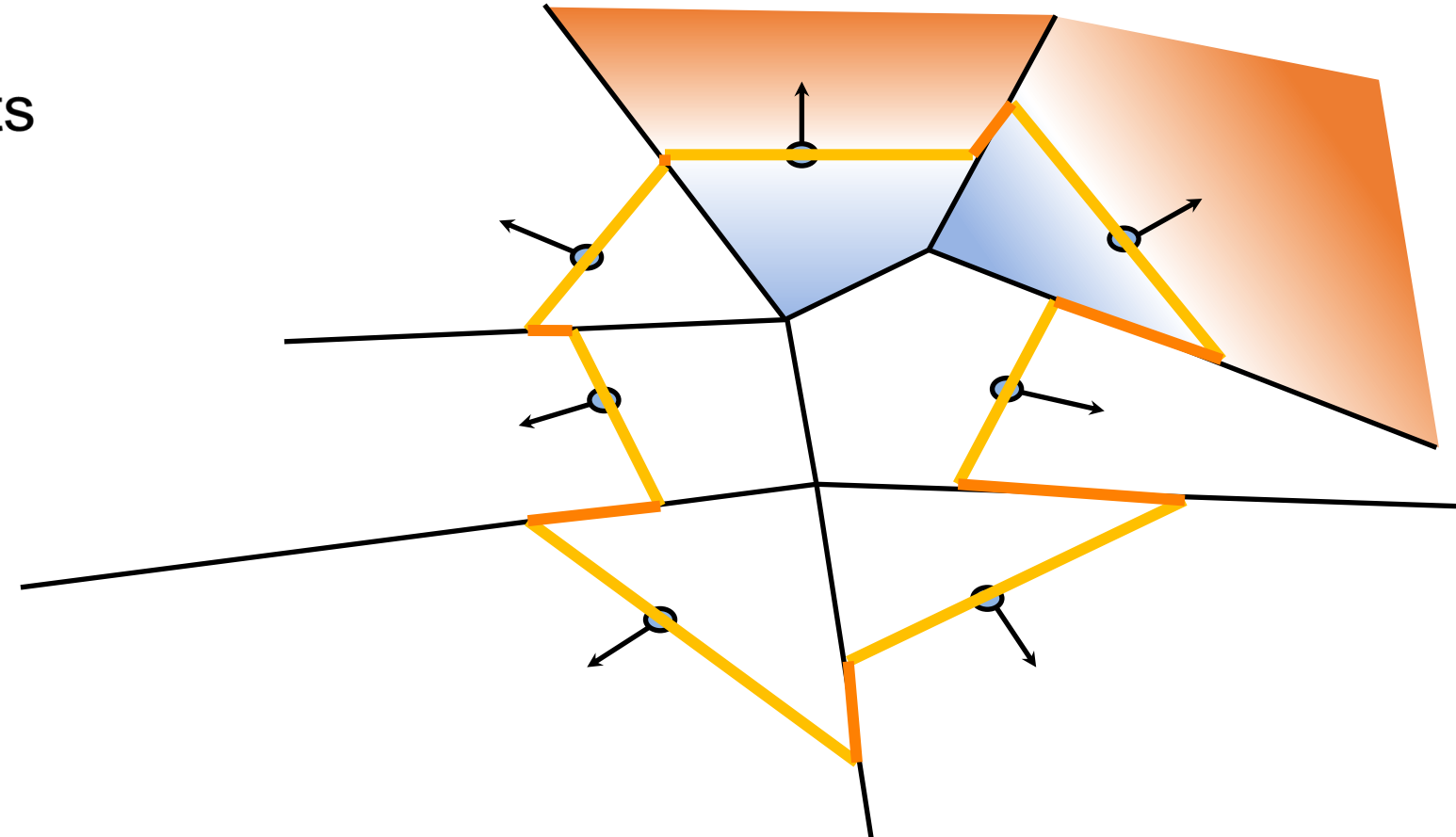




Hoppe's Method – Analysis

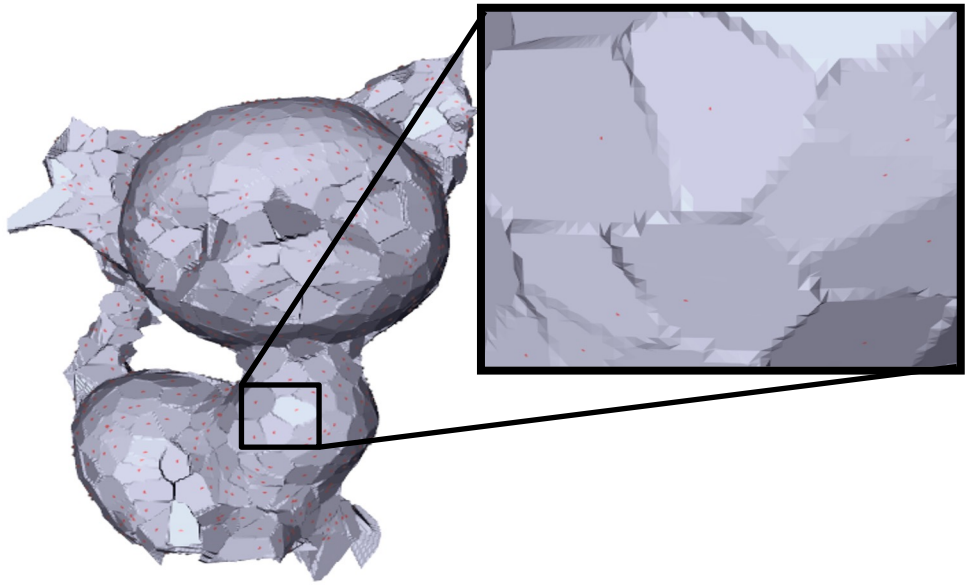
- Function $f(x)$ is $f(x) = (x - q_i) \cdot n_i$ if q_i is the point closest to x
- $f(x)$ is piecewise linear, defined on the Voronoi diagram of the input points
- Discontinuous along Voronoi Edges
- Marching cubes makes it manifold again

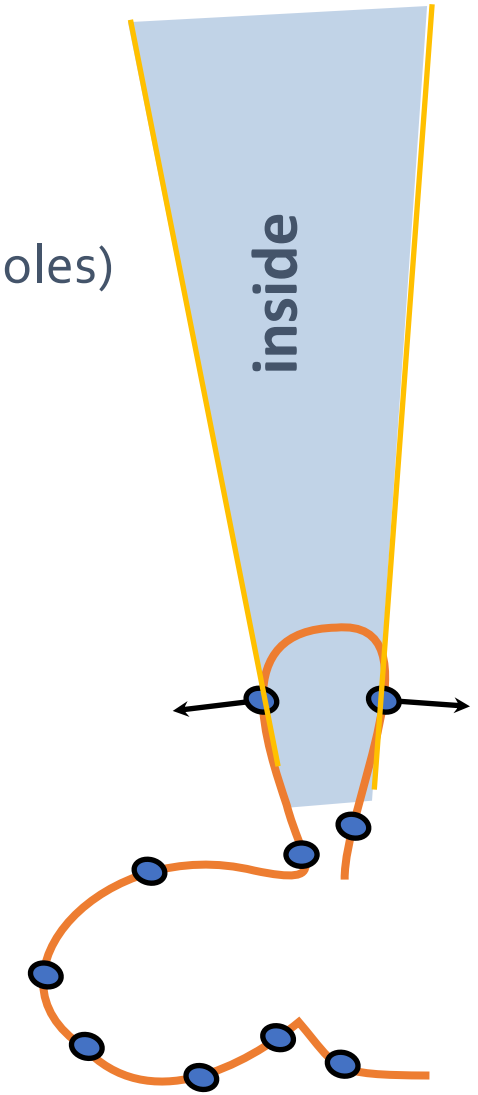
linear



Hoppe's Method – Analysis

- Easy to implement
  - works well for clean data (uniformly sampled, no noise, no outliers, no holes)
- Piecewise linear causes piecewise linear artifacts
- May cause artifact if data set is undersampled





Partition of Unity Implicits

- Better way:
  To evaluate the implicit function at some point p
  - look for the k nearest samples x_i
  - compute their distance just as before: $d_i = (x_i - p) \cdot n_i$
  - blend these, e.g., based on their distance to p:

$$f(p) = \frac{\sum_i w(\|x_i - p\|) d_i}{\sum_i w(\|x_i - p\|)}$$

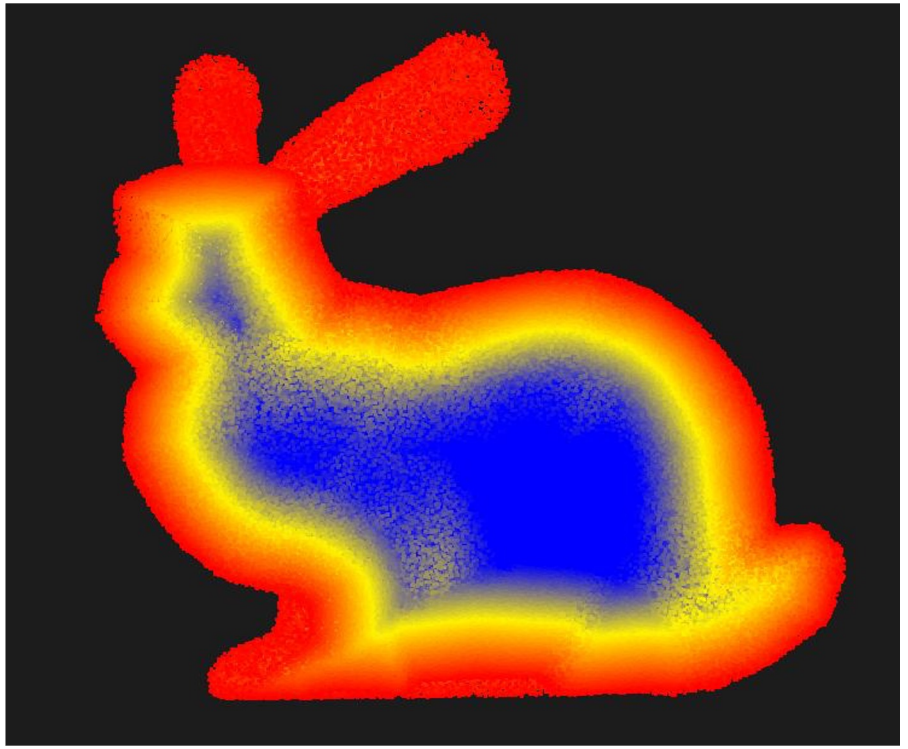
  - smoother result

Variant for Range Images: Structured Depth Grid

- Curless/Levoy: "A Volumetric Method for Building Complex Models from Range Images"

- Method to fuse noisy depth images from known camera paositions

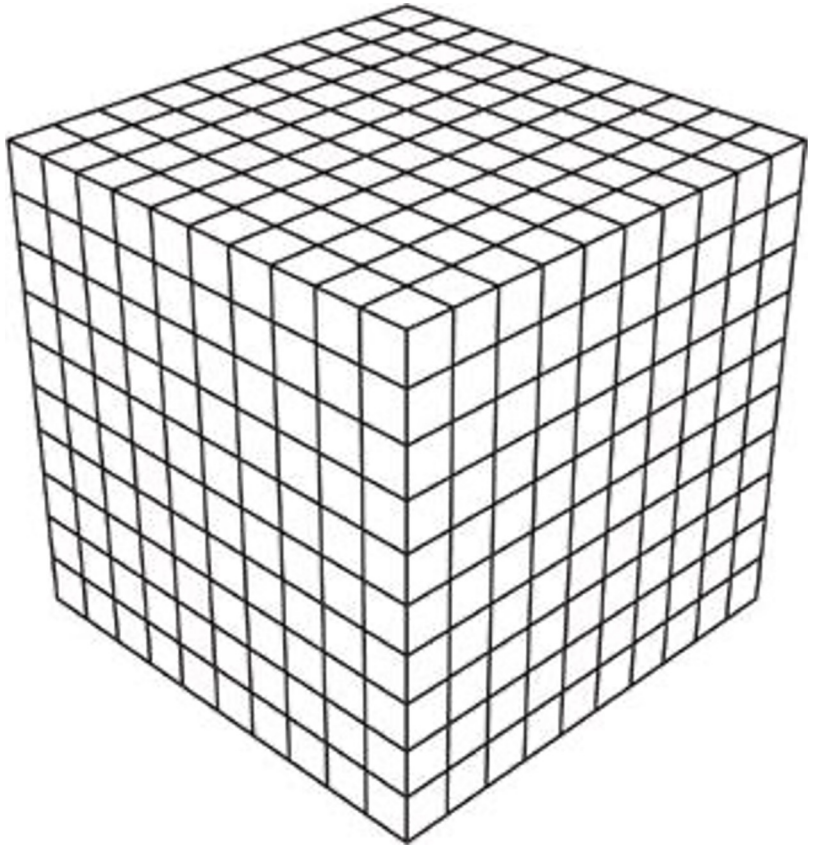
- Idea:
  - define **signed distance field** as implicit function
  - value of implicit function is distance to surface
  - negative distance ⟶ inside



http://blog.blackredking.org

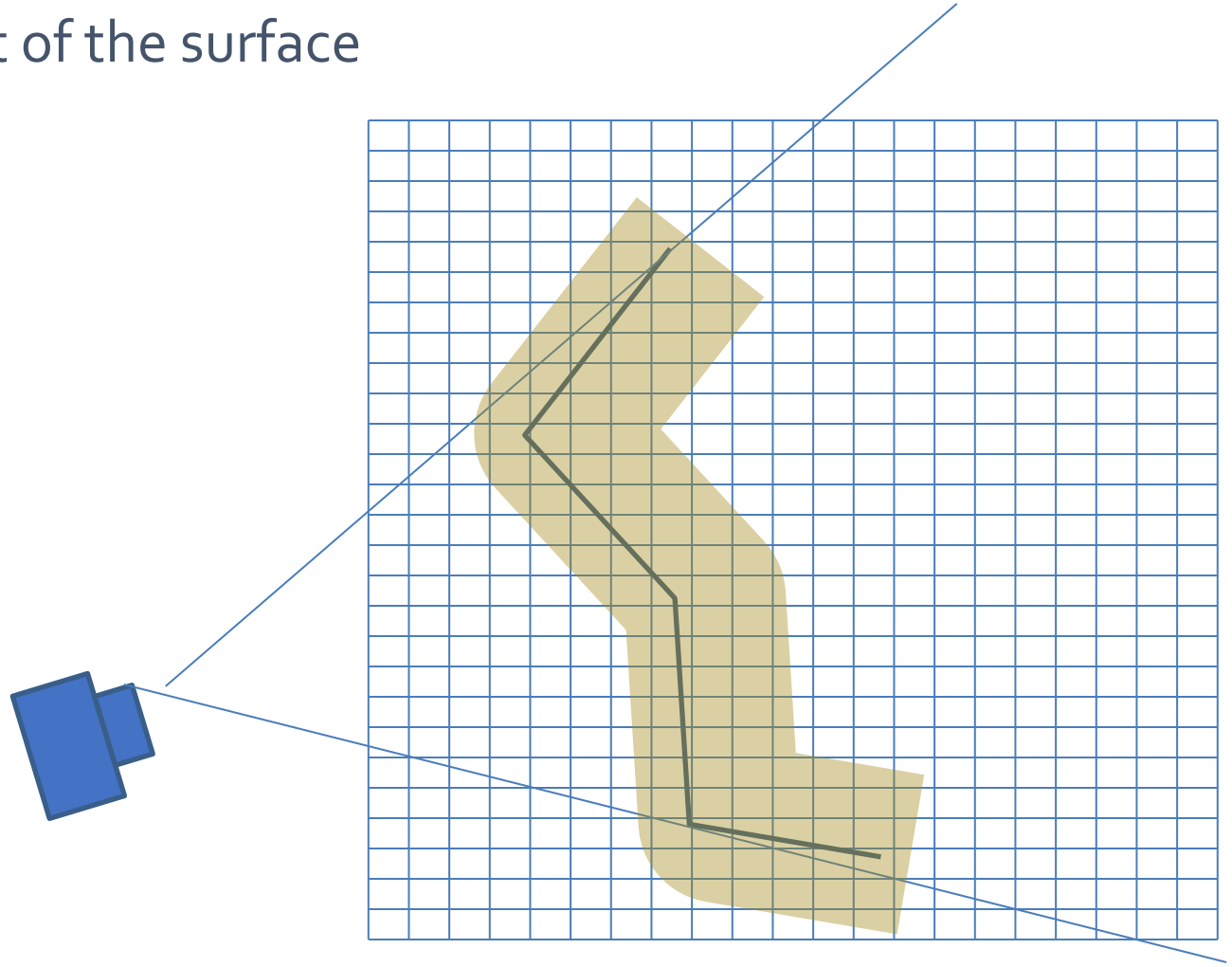
Truncated Signed Distance Fields

- Store this function on a 3D grid
- Truncated → store values only nearby surface

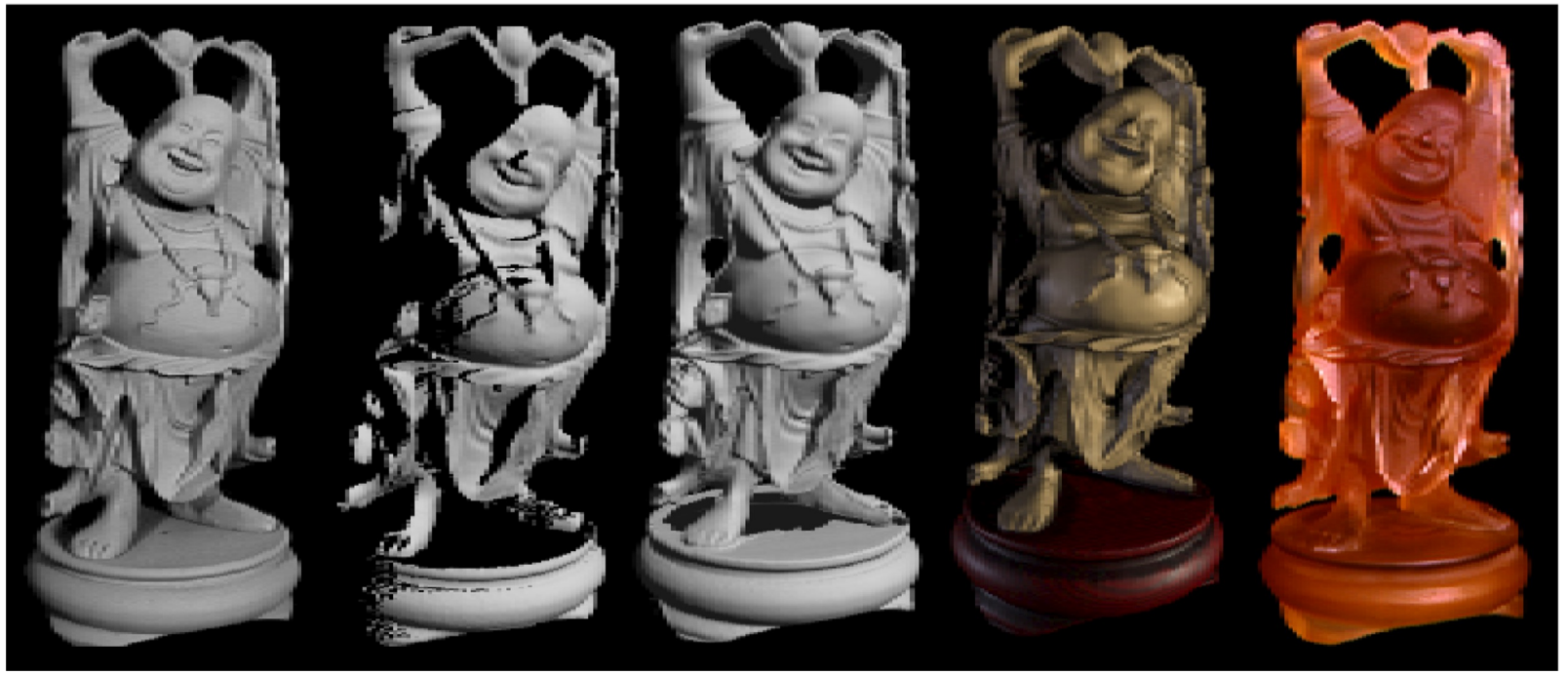


Truncated Signed Distance Fields

- Each depth image defines a part of the surface
- for each depth image
  - in neighborhood of this surface ⟶ set signed distance values in grid (*truncated* SDF)
  - if value is already set, merge values ⟶ smoothing out noise
  - "erase" area between camera and surface
- Finally, reconstruct using Marching Cubes



Example: Buddha reconstruction



Bringing all together – in Real-Time !

- Kinect Fusion
- See also: Kinect Fusion

End of Computer Vision Course for Summer 2023