A Soft Union based Method for Virtual Restoration and 3D Printing of Cultural Heritage Objects

R. Gregor, P. Mavridis, A. Wiltsche & T. Schreck

Problem Statement

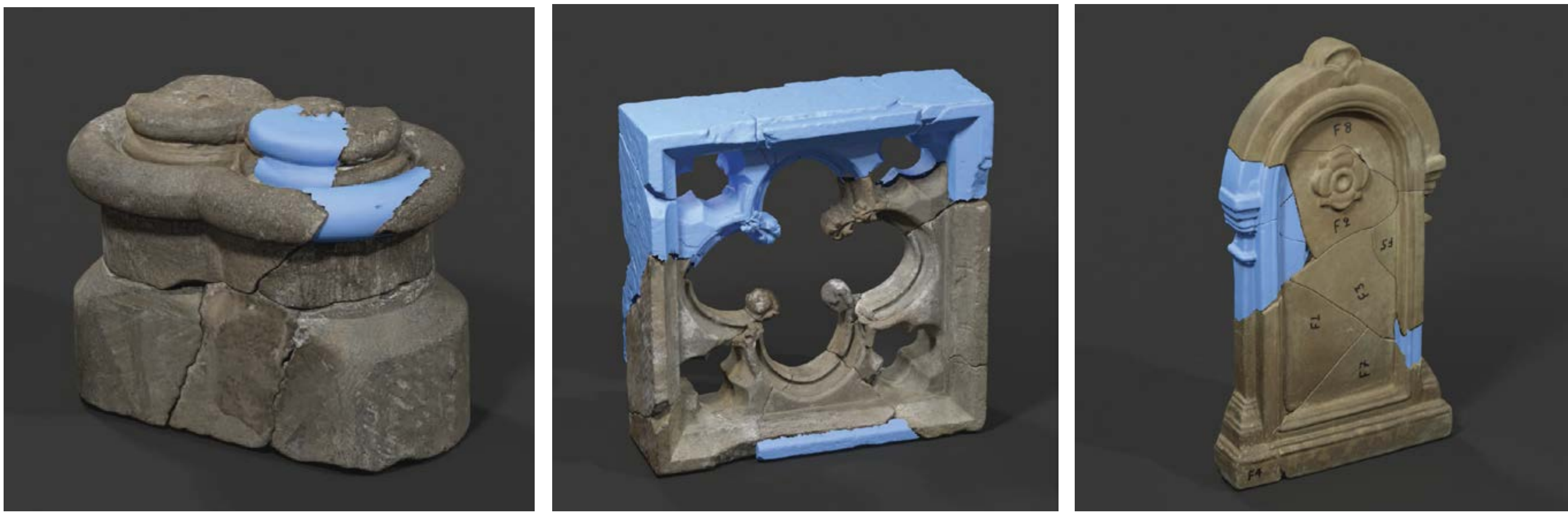
- Many methods address the (digital) reassembly problem for fractured objects

[Huang et al. 2006, Gregor et al. 2014, Mavridis et al. 2015, …]

- The output consists of multiple disjoint objects with visible gaps between them ***(fracture lines)***

Problem Statement

- Similar problem with symmetry-based completion

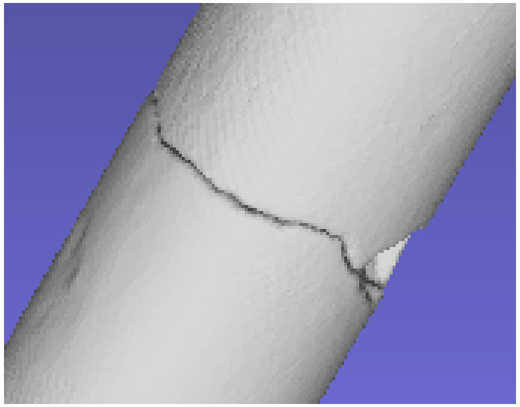


[Sipiran et al. 2014, Mavridis et al. 2015, Andreadis et al. 2015, ...]

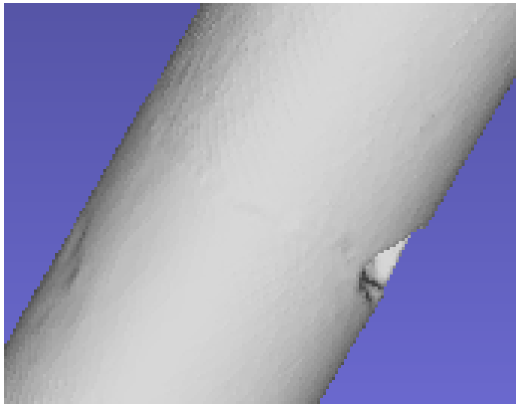
- The original and complementary shape need to be ***seamlessly merged***.

Goal

- Given the output of a digital reassembly or completion algorithm:
  - Create a **single watertight object**
  - *With concealed* fracture lines (if desired)
- The last step in a digital restoration pipeline



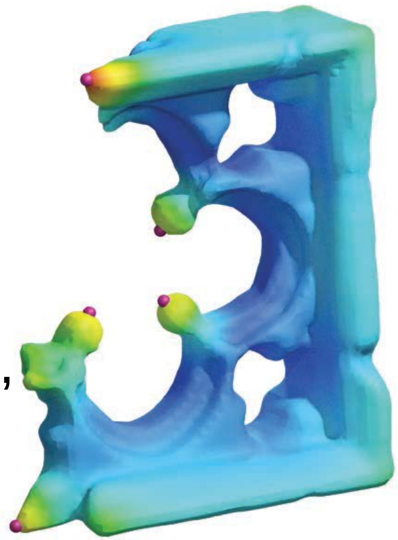
Input



Watertight output

Motivation

- Provide a complete digital restoration which allows to:
  - **Restore the appearance** of a fractured CH object.
    Shape Analysis
    Compute *descriptors* for symmetry detection, retrieval,...
  - **Finite Element Analysis**
    Study the stability, reaction to physical forces, vibrations or heat.
  - **3D Printing**
    A direct way to show the results to a wider audience.



For many of these tasks, the reassembled object should be **watertight**.

Potential Approaches

- **Option 1:** Solve the problem using a re-meshing approach
  - **Step 1:** Filter-out the fracture facet points. (based on proximity and orientation)
  - **Step 2:** Perform a re-meshing of the remaining points.
    - (Screened) Poisson reconstruction
    - Or similar surfel-based algorithm [Amenta and Kil 2004]

(Huang et al. 2006 mentions this approach, although the paper does not include any related results)

Re-meshing problems

- Does not preserve the intact regions
  - They are affected by the re-meshing process.
- Computationally intensive for dense point clouds
- Filtering parameters could require a lot of tuning.

More details on Poisson-based the results section

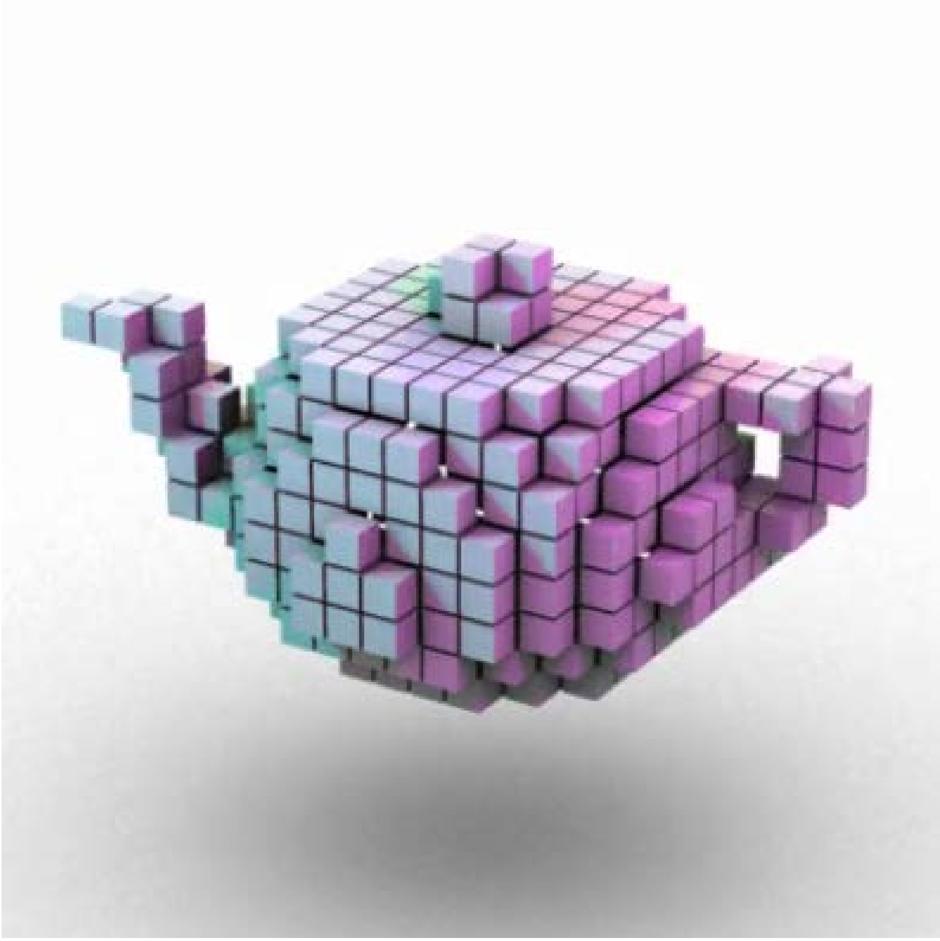
Potential Approaches

- **Option 2:** Solve the problem using a volumetric approach
  - **Step 1:** Convert input fragments to volumes
  - **Step 2:** Merge the volumes and fill the gaps
  - **Step 3:** Convert back to triangles

Which ***volumetric representation*** to use?

Binary Volumes

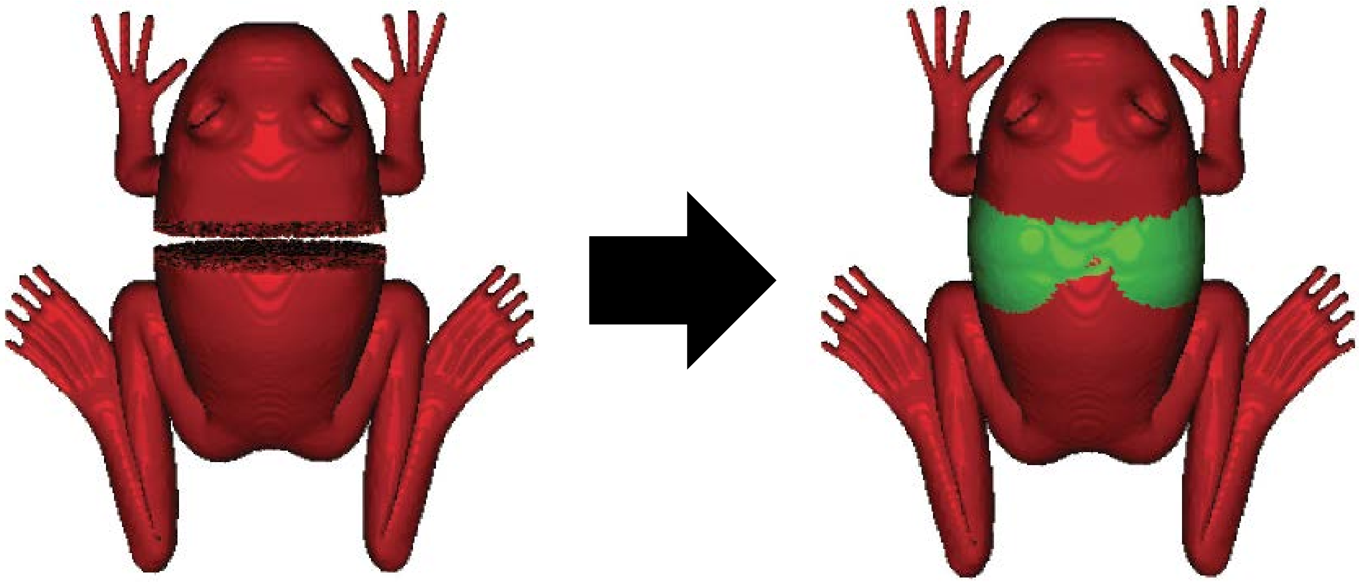
- Binary voxels:
  - 0 -> empty space
  - 1 -> occupied



Voxelized teapot
(Source: Maya voxelization script)

Binary Volumes

- *Robust Gap Removal from Binary Volumes* [Sobiecki et al. 2016]



(Concurrently developed with our approach)

Binary Volumes

- Very high resolutions are required to avoid sampling artifacts & aliasing
  - Increased memory consumption
  - Increased processing time

For our application **we need a more *rich* volumetric representation**

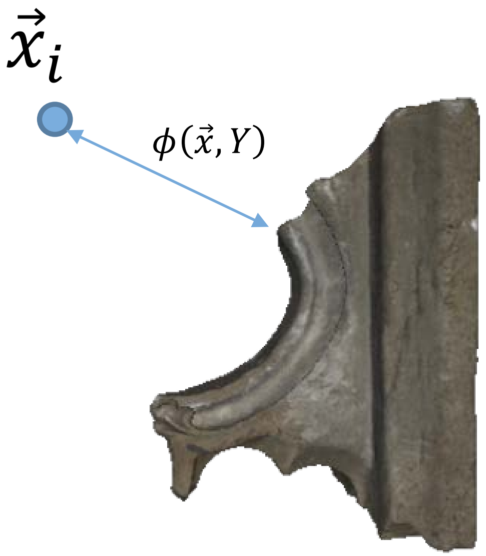
Distance Function

- For every point $\vec{x}$ in space,

Distance Function

- For every point $\vec{x}$ in space, $\phi(\vec{x}, Y)$ measures the closest distance to surface Y.

$$\phi(\vec{x}, Y) = \min_{y \in Y} \|x - y\|_2$$

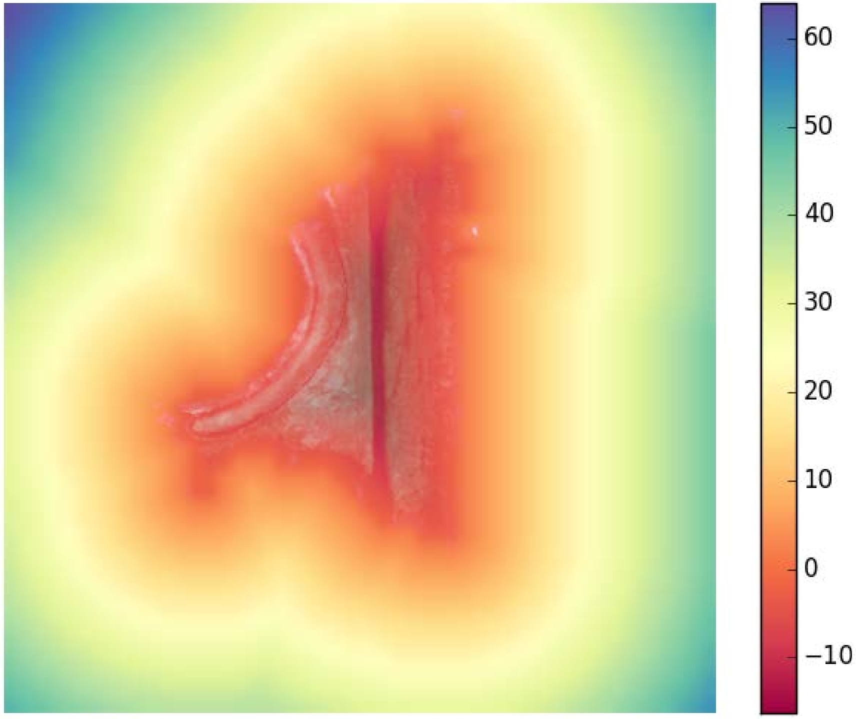


Distance Function

- For every point $\vec{x}$ in space, $\phi(\vec{x}, Y)$ measures the closest distance to surface Y.

$$\phi(\vec{x}, Y) = \min_{y \in Y} \|x - y\|_2$$

- This function defines a field in space (***distance field***)



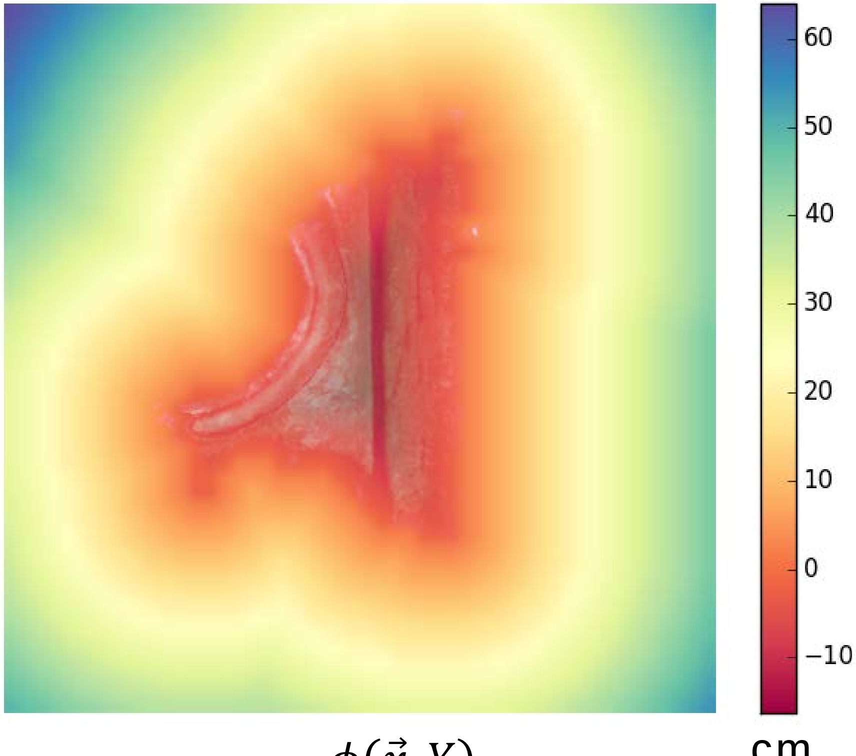
$\phi(\vec{x}, Y)$

Distance Function

- For every point $\vec{x}$ in space, $\phi(\vec{x}, Y)$ measures the closest distance to surface Y.

$$\phi(\vec{x}, Y) = \min_{y \in Y} \|x - y\|_2$$

- This function defines a field in space (***distance field***)
- ***Signed*** distance field: Negative values for points inside the surface.



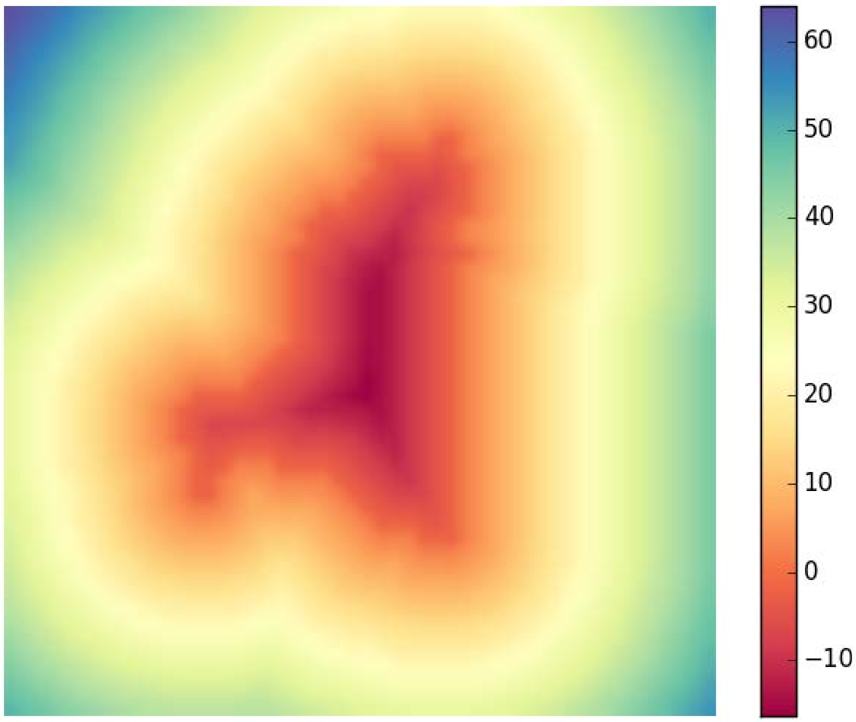
$\phi(\vec{x}, Y)$

Distance Function

- For every point $\vec{x}$ in space, $\phi(\vec{x}, Y)$ measures the closest distance to surface Y.

$$\phi(\vec{x}, Y) = \min_{y \in Y} \|x - y\|_2$$

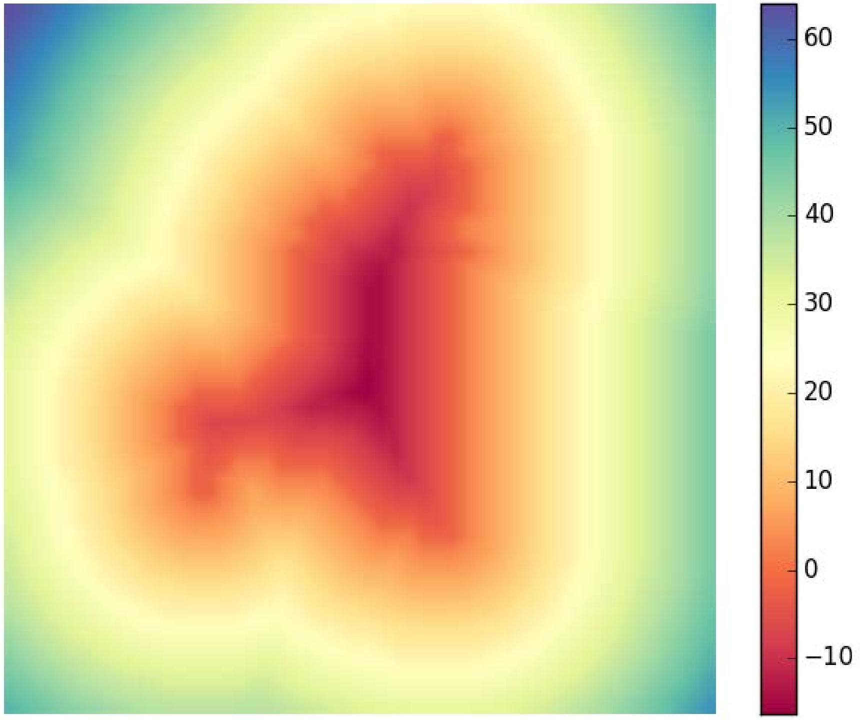
- This function defines a field in space (***distance field***)
- ***Signed*** distance field: Negative values for points inside the surface.



$\phi(\vec{x}, Y)$

Data Structure

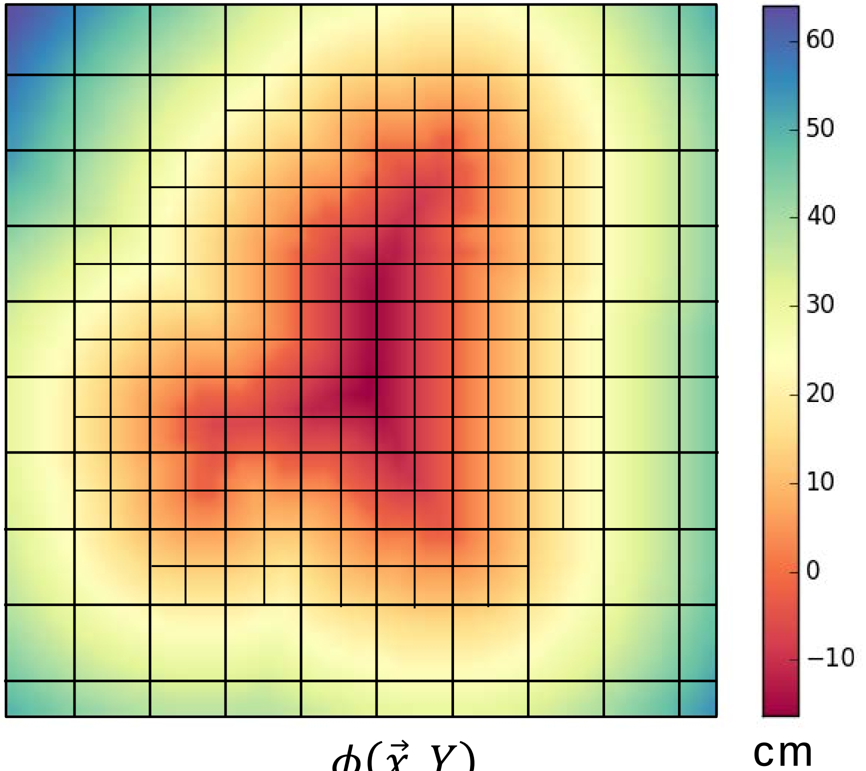
- $\phi(\vec{x}, Y)$ is a continuous function in space ($\mathcal{R}^3 \rightarrow \mathcal{R}$)



$\phi(\vec{x}, Y)$

Data Structure

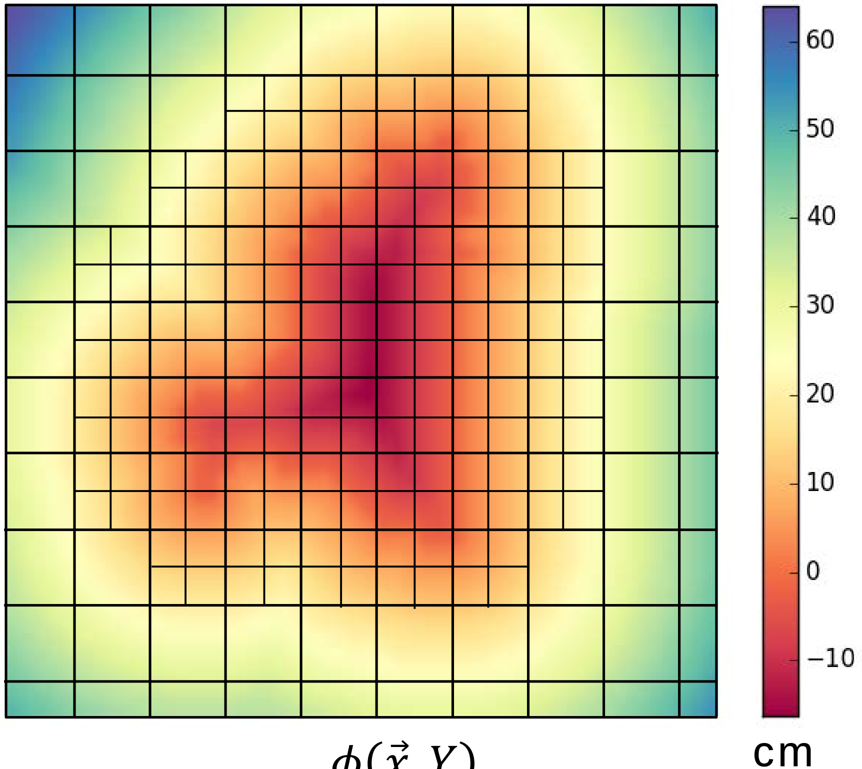
- $\phi(\vec{x}, Y)$ is a continuous function in space ($\mathcal{R}^3 \rightarrow \mathcal{R}$)
- We ***discretize*** it over an hierarchical grid.



$\phi(\vec{x}, Y)$

Data Structure

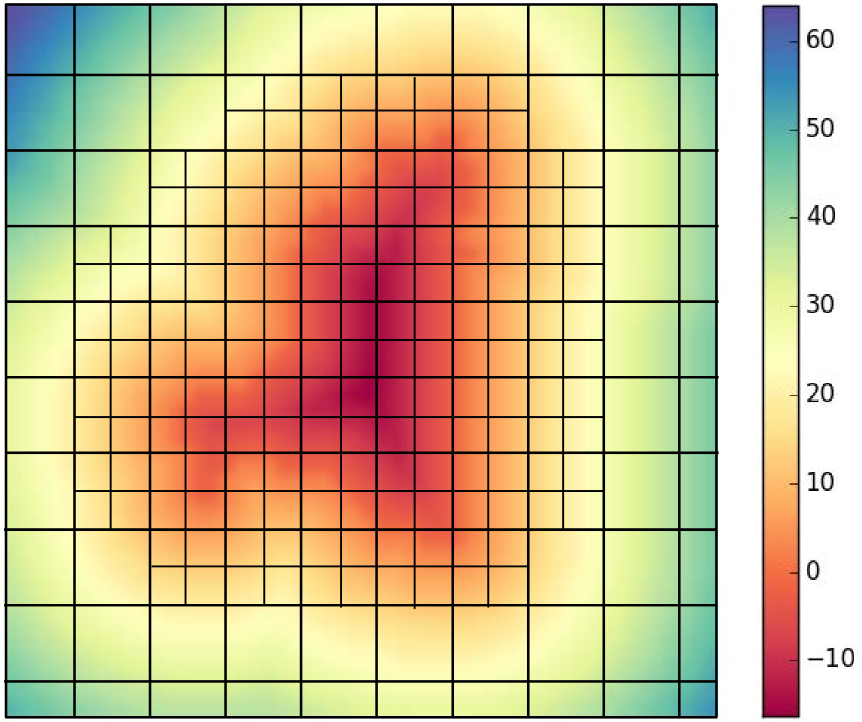
- $\phi(\vec{x}, Y)$ is a continuous function in space ($\mathcal{R}^3 \rightarrow \mathcal{R}$)
- We ***discretize*** it over an hierarchical grid.
- Which is stored in **OpenVDB** format



$\phi(\vec{x}, Y)$

Data Structure

- $\phi(\vec{x}, Y)$ is a continuous function in space ($\mathcal{R}^3 \rightarrow \mathcal{R}$)
- We ***discretize*** it over an hierarchical grid.
- Which is stored in **OpenVDB** format

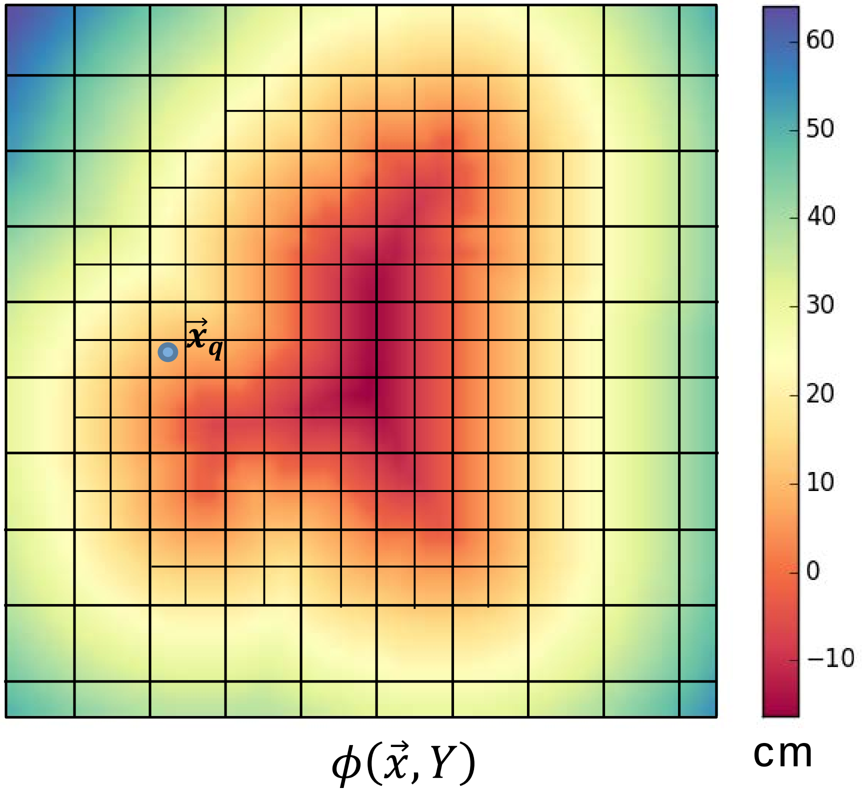


$\phi(\vec{x}, Y)$

(For clarity we show only two hierarchy levels, while OpenVDB actually uses three)

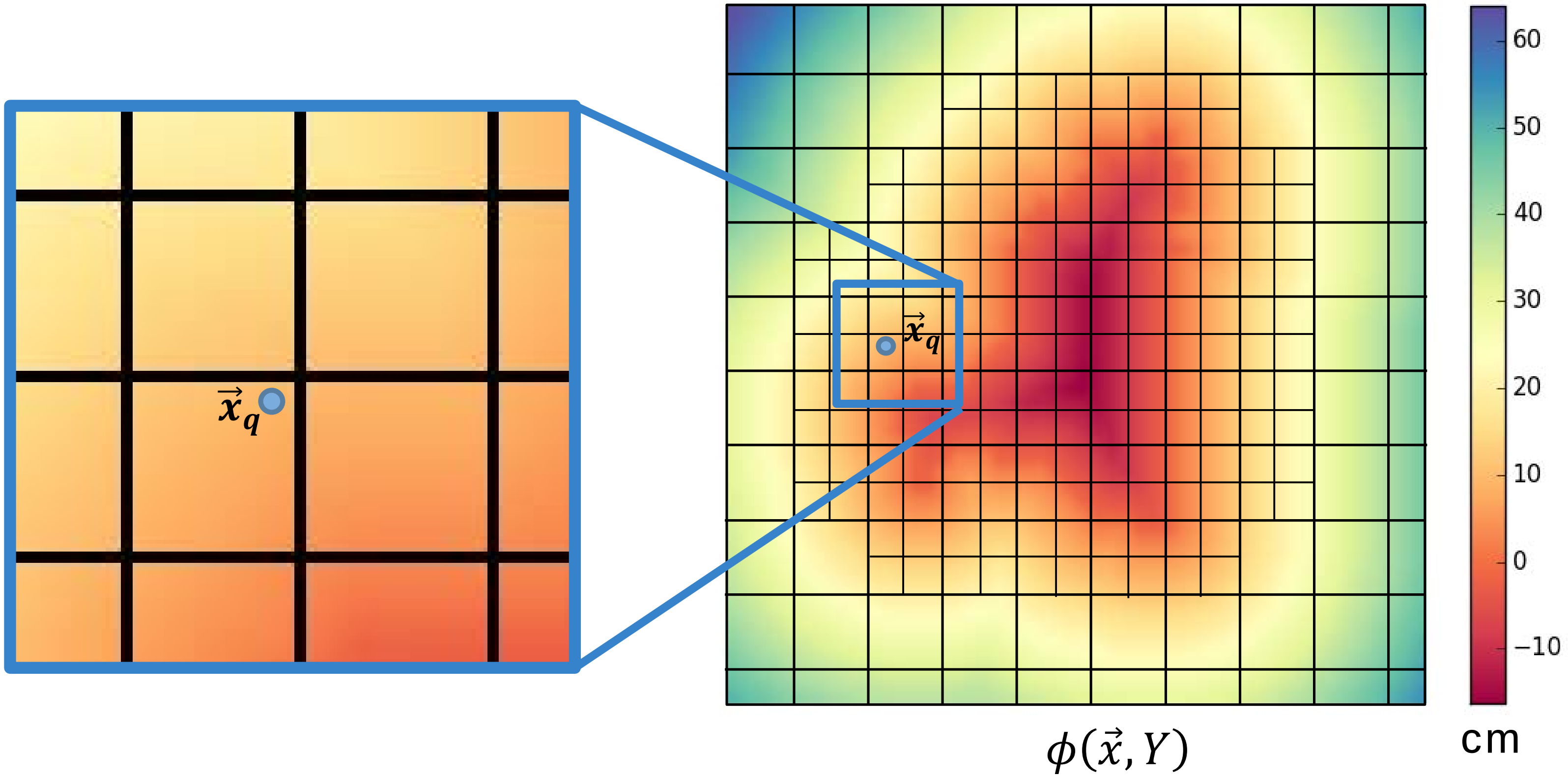
Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$



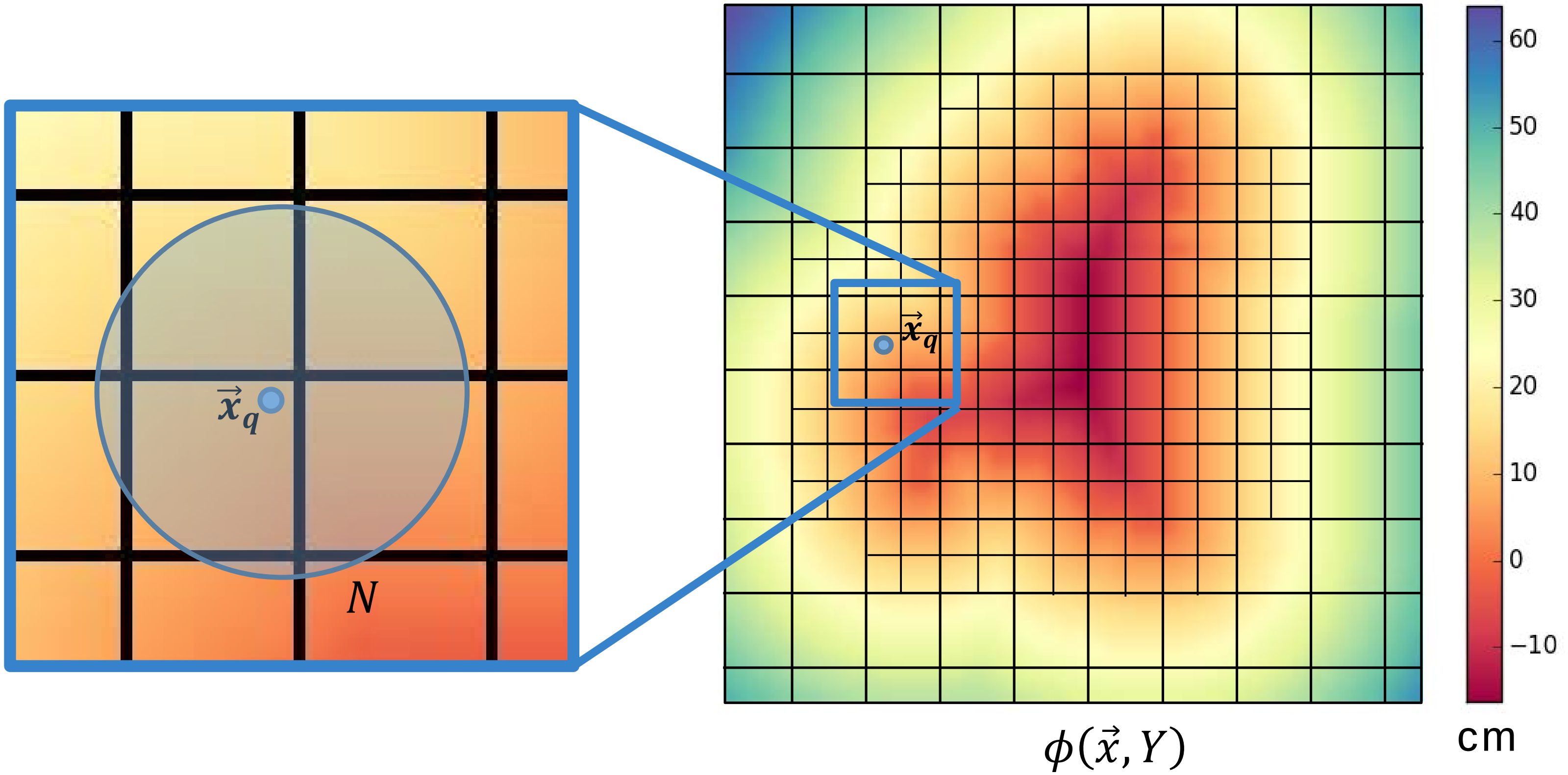
Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$



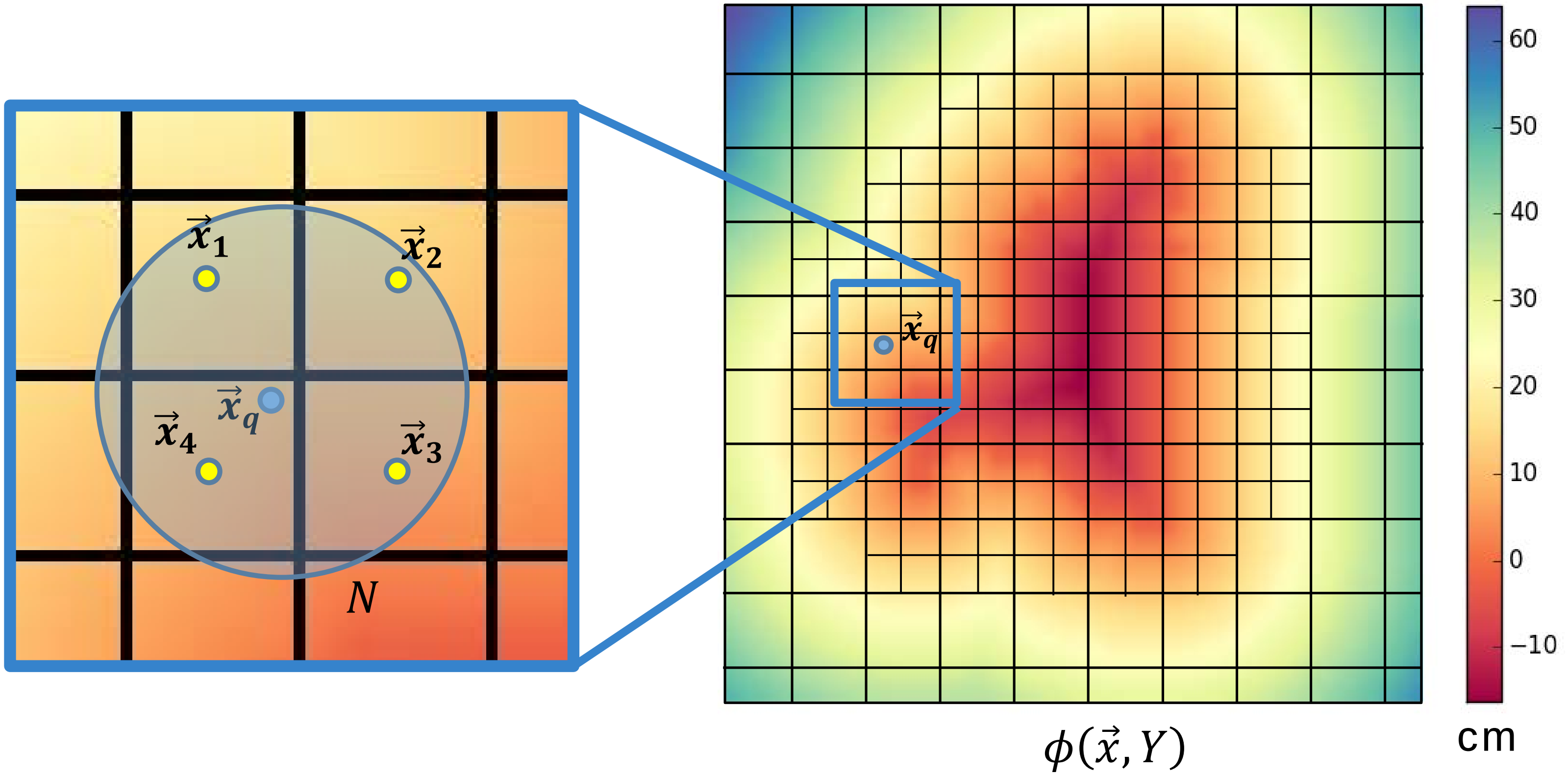
Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$



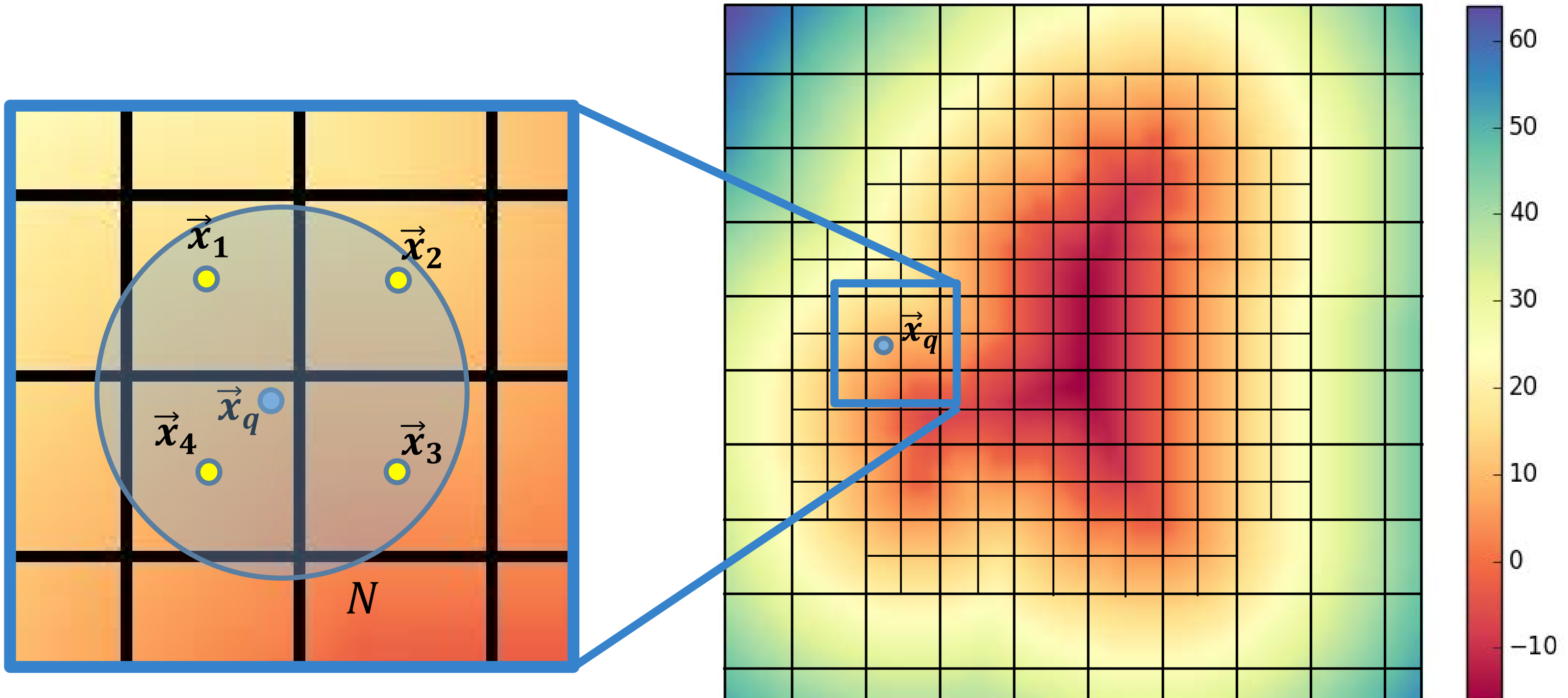
Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$



Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$

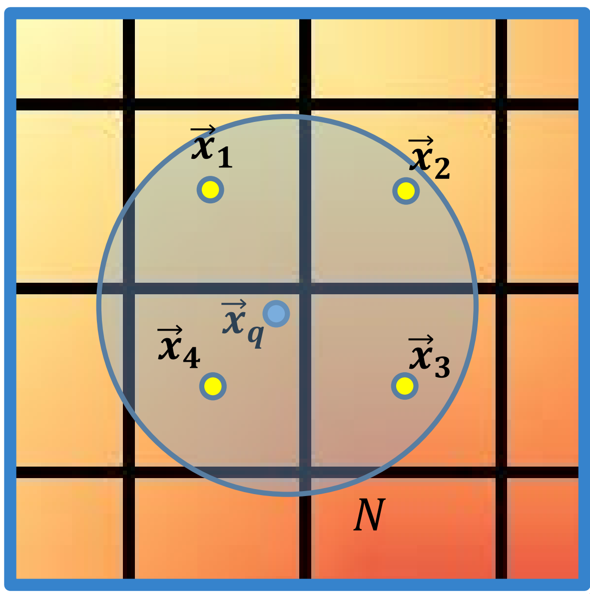


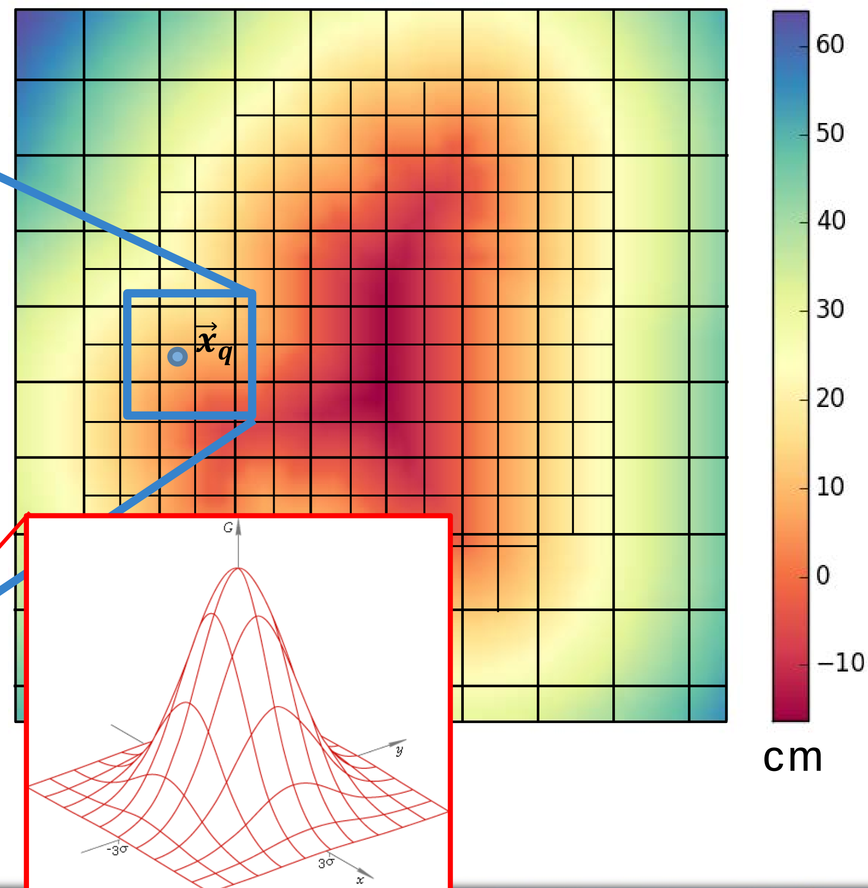
$\phi(\vec{x}, Y)$ cm

$$\phi(\vec{x}_q, Y) = \frac{\sum_{x_i \epsilon N} w_i \phi(\vec{x}_i, Y)}{\sum_{x_i \epsilon N} w_i}$$

Sampling

- Given the discretized representation, we can **reconstruct** the distance function at an **arbitrary point** $\vec{x}_q$



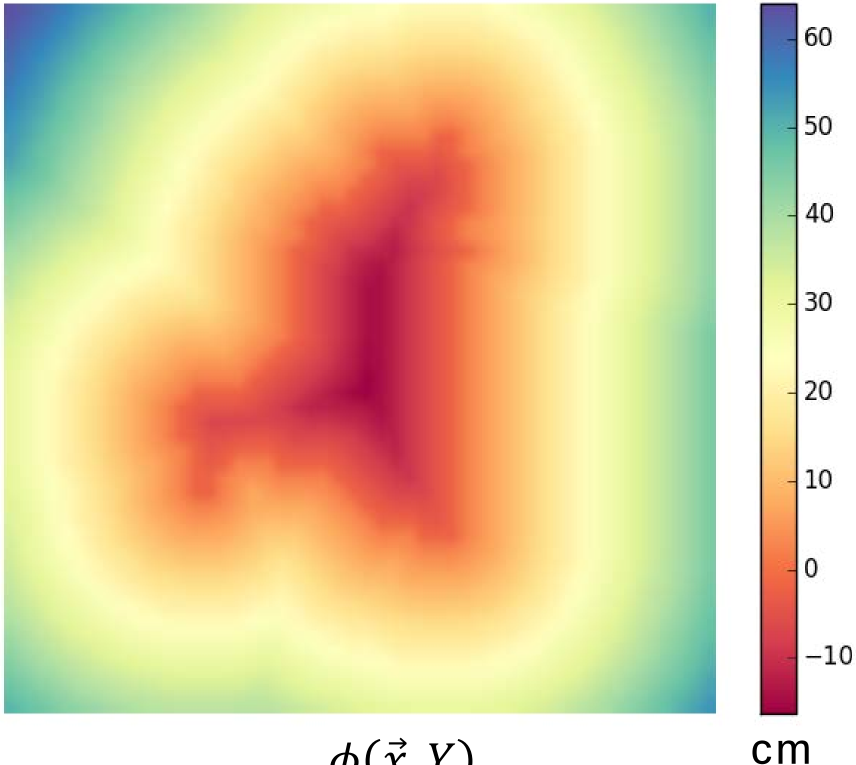


$$\phi(\vec{x}_q, Y) = \frac{\sum_{x_i \epsilon N} w_i \phi(\vec{x}_i, Y)}{\sum_{x_i \epsilon N} w_i}$$

Gaussian or similar reconstruction filter

Sampling

- With sampling, we can *forget* about discretization and "*pretend*" that we have a continuous function...

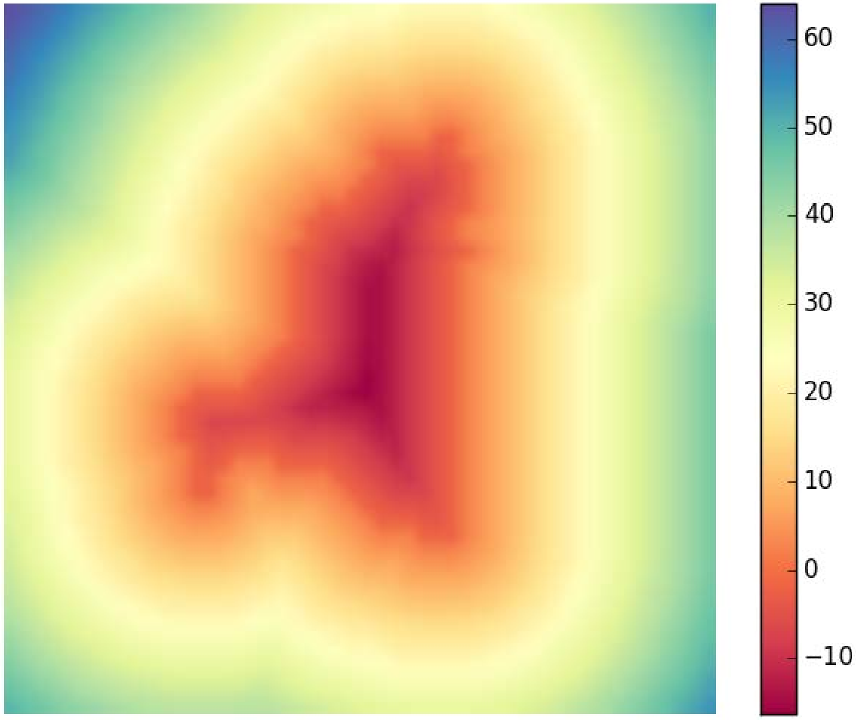


$\phi(\vec{x}, Y)$

Implicit Surfaces (IS)

- Given $\phi(\vec{x}, Y)$, we can define the original surface Y ***implicitly*** as

$$\phi(\vec{x}, Y) = 0$$



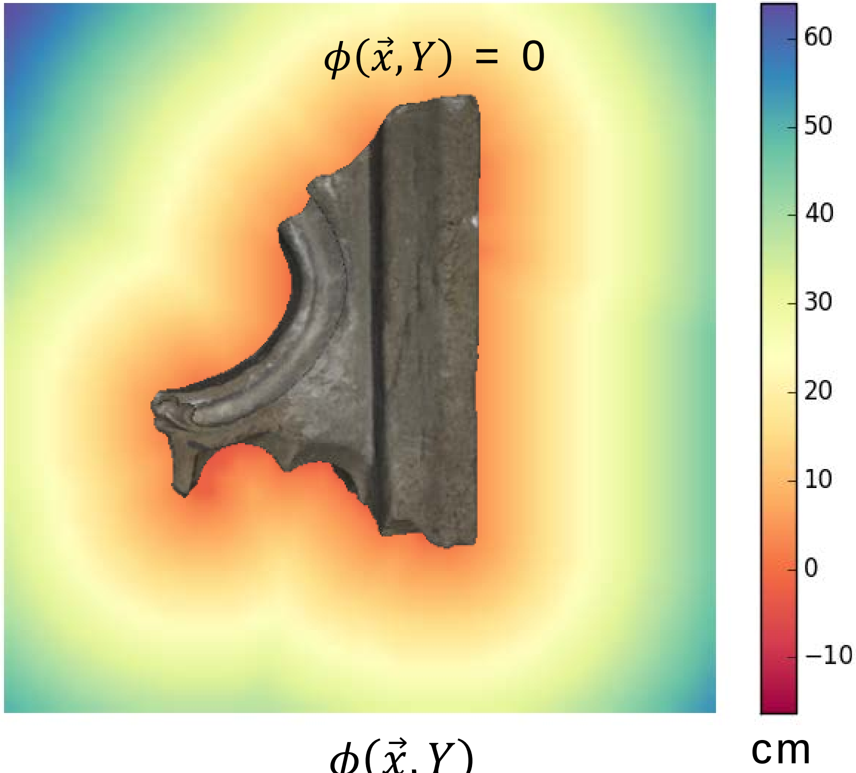
$\phi(\vec{x}, Y)$

Implicit Surfaces (IS)

- Given $\phi(\vec{x}, Y)$, we can define the original surface Y ***implicitly*** as

$$\phi(\vec{x}, Y) = 0$$

(Zero-level *isosurface*)



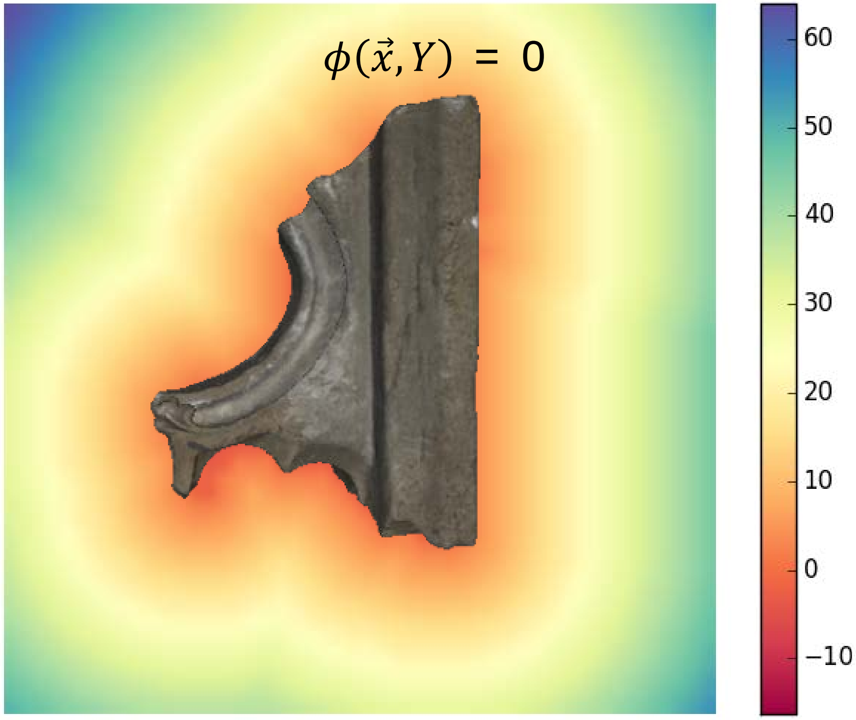
$\phi(\vec{x}, Y)$

Implicit Surfaces (IS)

- Given $\phi(\vec{x}, Y)$, we can define the original surface Y ***implicitly*** as

$$\phi(\vec{x}, Y) = 0$$

(Zero-level *isosurface*)



$\phi(\vec{x}, Y)$

The terms ***distance function***, ***distance field***, ***level set*** are used interchangeably

Geometry Processing with IS

- Geometric surfaces are represented using distance fields
- Geometry processing is made easy:
  - Step 1: define a function that takes as input 1D distance values and outputs new 1D distance values ($\mathcal{R} \rightarrow \mathcal{R}$).
  - Step 2: Apply this function on the entire distance field (or locally).

- CSG union operation:

$$\phi(\vec{x}, U) = \min(\phi(\vec{x}, Y), \phi(\vec{x}, Z))$$

Where $\phi(\vec{x}, U)$ is the distance function that encodes the union of surface Y and surface Z

Soft Union Operation

- Traditional CSG union is not suited for our application, because we want to close potential gaps between the two surfaces
- For this reason we define a pairwise *soft union* operator:

$$\phi(\vec{x}, U) = \min\big(\phi(\vec{x}, Y), \phi(\vec{x}, Z)\big)$$

Same as the hard union

Soft Union Operation

- Traditional CSG union is not suited for our application, because we want to close potential gaps between the two surfaces
- For this reason we define a pairwise *soft union* operator:

$$\phi(\vec{x}, U) = \min\big(\phi(\vec{x}, Y), \phi(\vec{x}, Z)\big) - \frac{g(x)^2}{4r}$$

Extra distance attenuation term

Same as the hard union

where

$$g(\vec{x}) = \max(r - |\phi(\vec{x}, Y) - \phi(\vec{x}, Z)|, 0)$$

The parameter *r* controls the *smoothness* of the union operation

Soft Union Operation

- Traditional CSG union is not suited for our application, because we want to close potential gaps between the two surfaces
- For this reason we define a pairwise *soft union* operator:

$$\phi(\vec{x}, U) = \min\big(\phi(\vec{x}, Y), \phi(\vec{x}, Z)\big) - \frac{g(x)^2}{4r}$$

Extra distance attenuation term

Same as the hard union

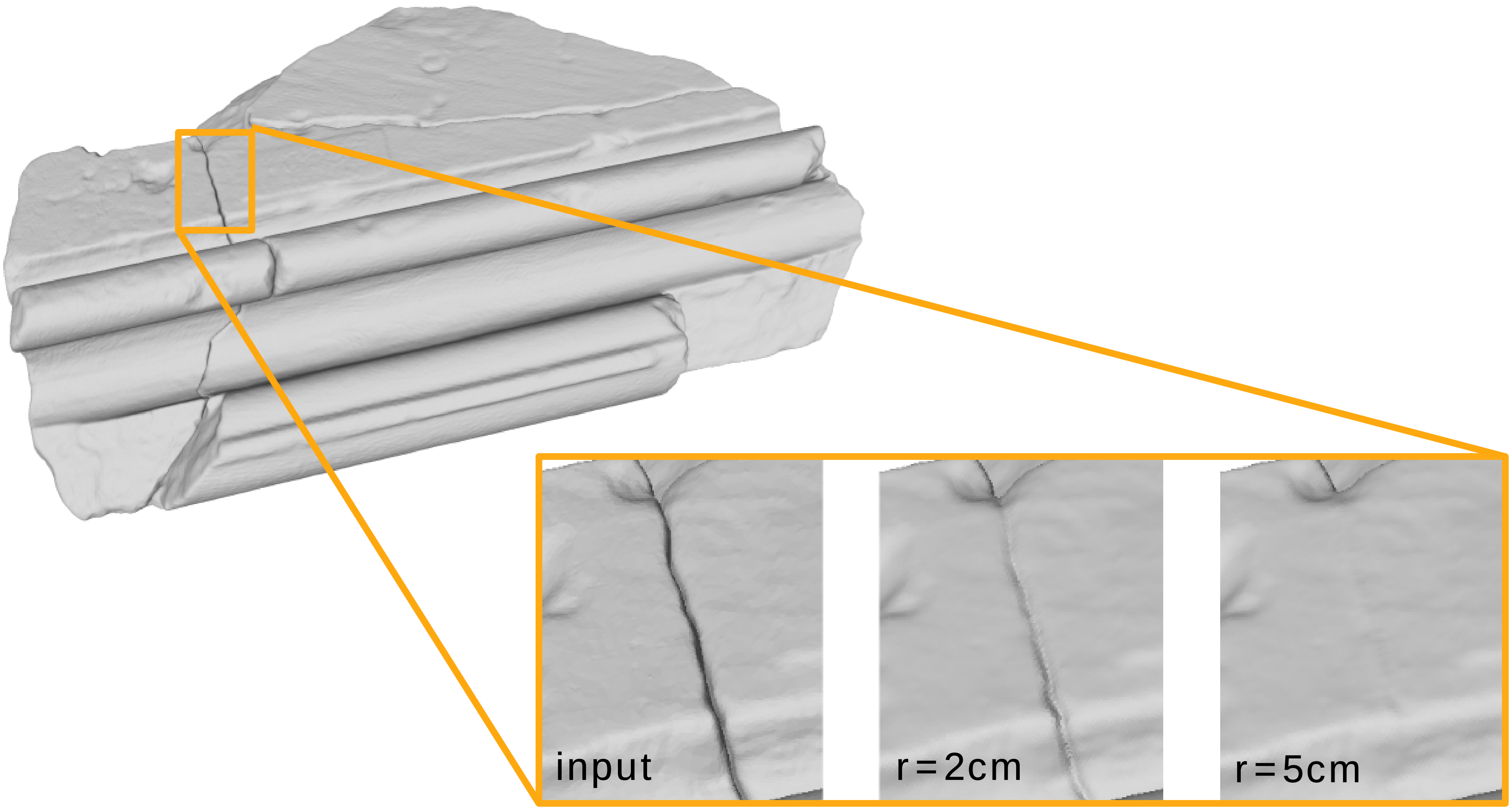
where

$$g(\vec{x}) = \max(r - |\phi(\vec{x}, Y) - \phi(\vec{x}, Z)|, 0)$$

The parameter *r* controls the *smoothness* of the union operation

For points $\vec{x}$ that are:
- close to only one of the two surfaces or far away from both, $g(\vec{x})$ will be zero.
- close to both surfaces, $g(\vec{x})$ will make $\phi(\vec{x}, U)$ go to zero more quickly.

Example



Iterative Accumulation

- For N objects $\phi(x, Y_i)$, we perform successive pairwise unions and iteratively accumulate the results:

$$\phi(\vec{x}, U)_i = su(\phi(\vec{x}, U)_{i-1}, \phi(x, Y_i)), \qquad 1 \leq i \leq N$$

where

$\phi(\vec{x}, U)_0$ is the empty volume

$su(.,.)$ is the smooth union operation in the previous slide

Iso-surface Polygonization

- In the end, we convert the zero-level iso-surface to a polygon surface
  - Marching cubes
  - **Dual contouring** (implemented in OpenVDB)

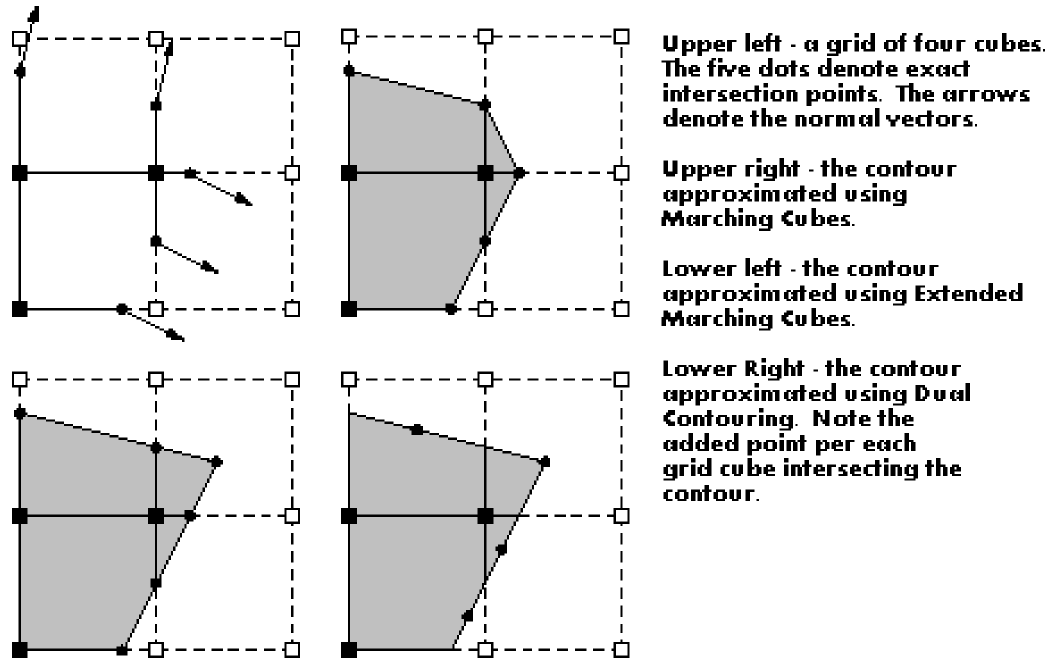
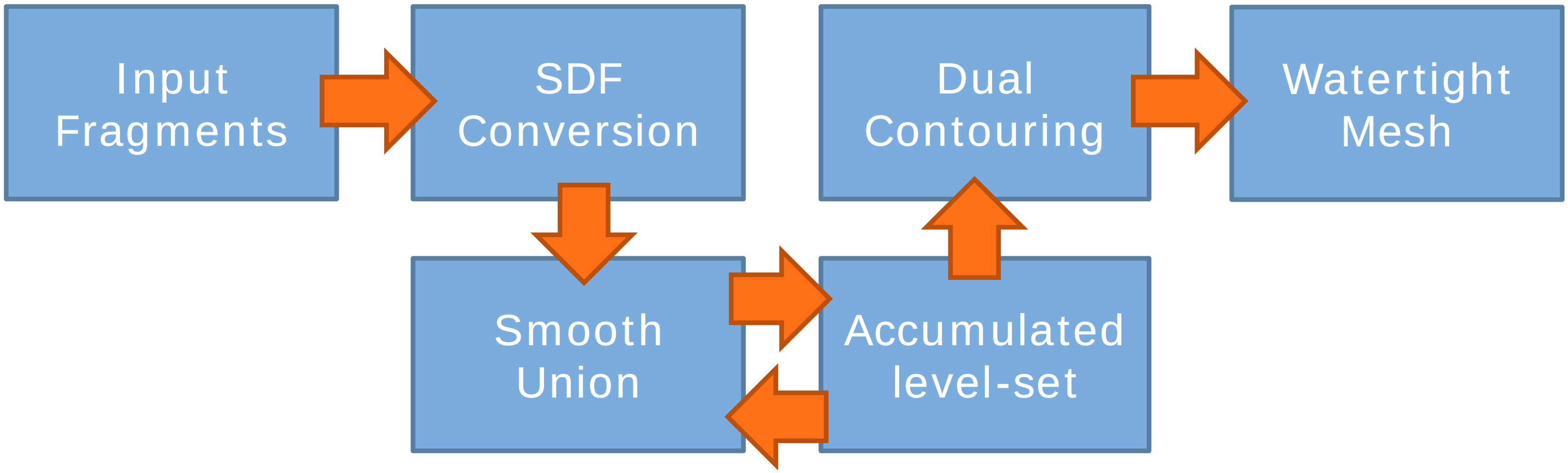
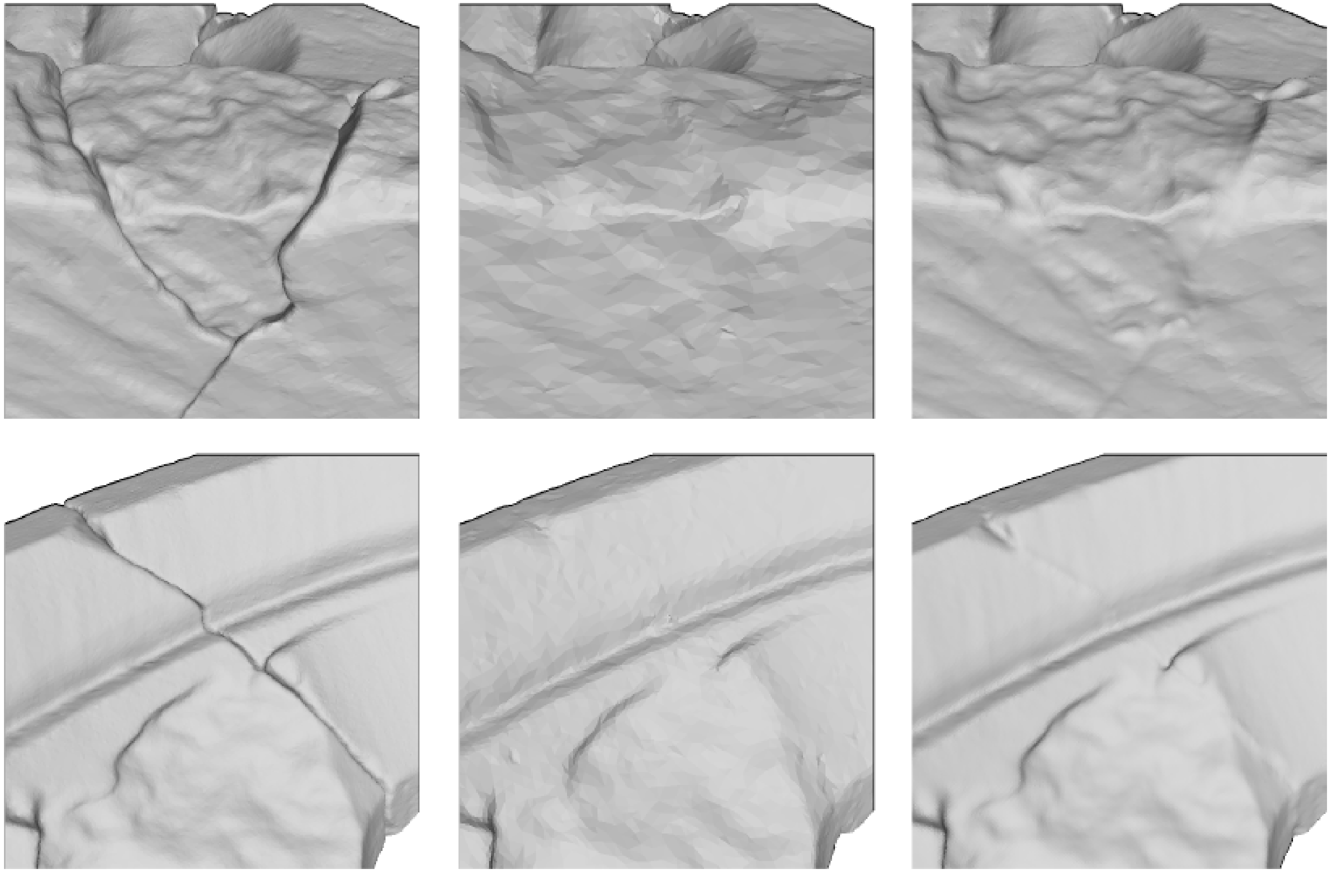


Image source: Ronen Tzur

Complete Pipeline



Results

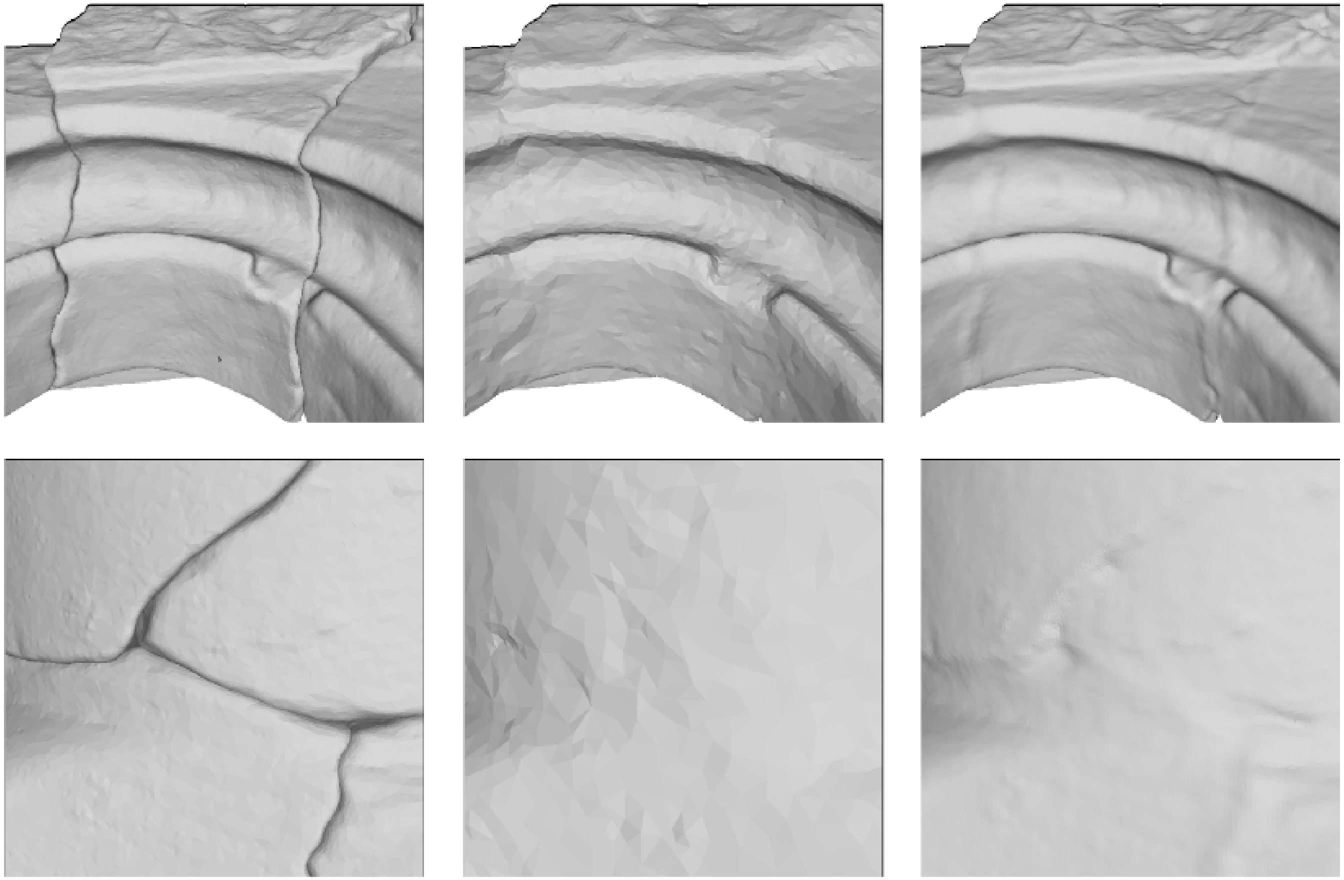


Input

Poisson
re-meshing

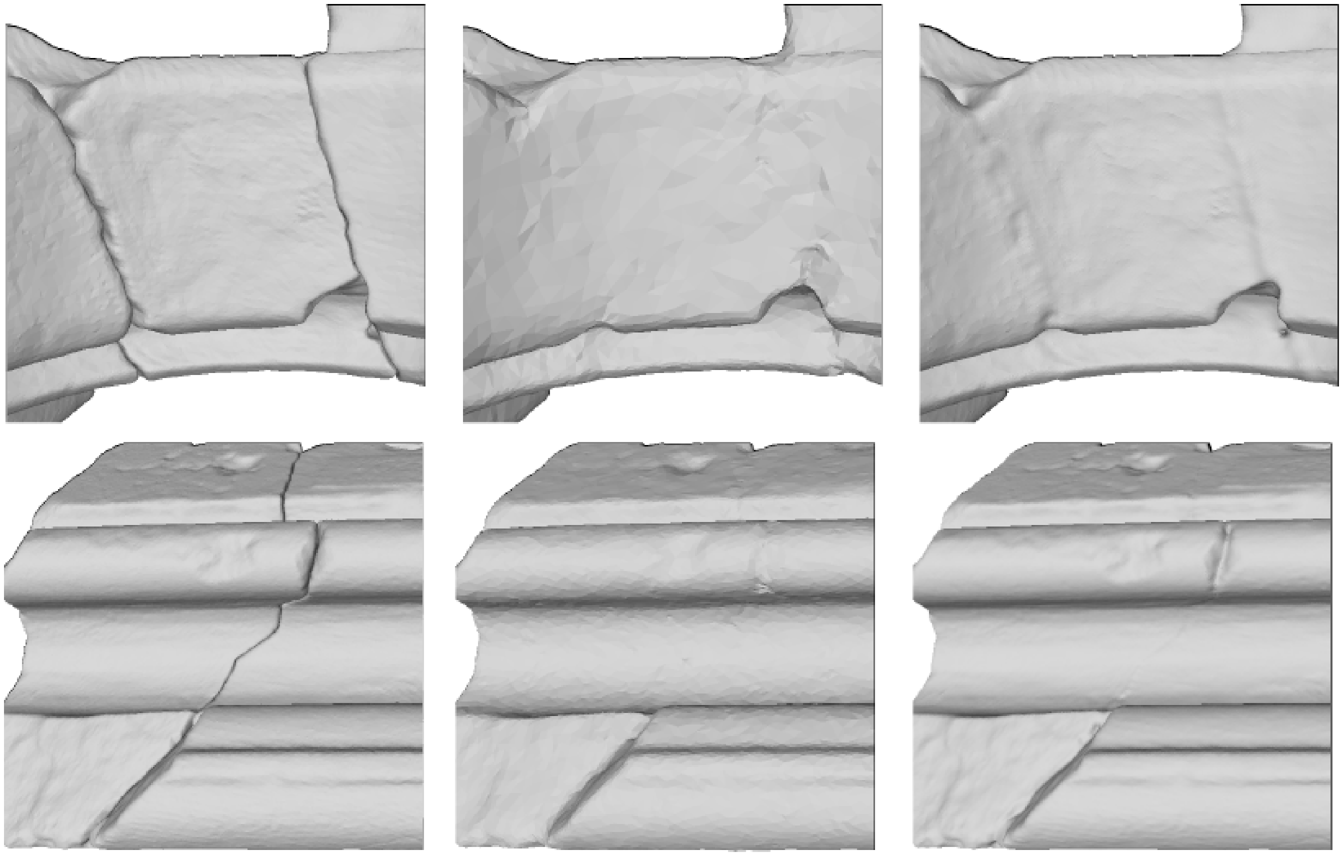
Soft Union

Results



Input | Poisson re-meshing | Soft Union

Results



Input

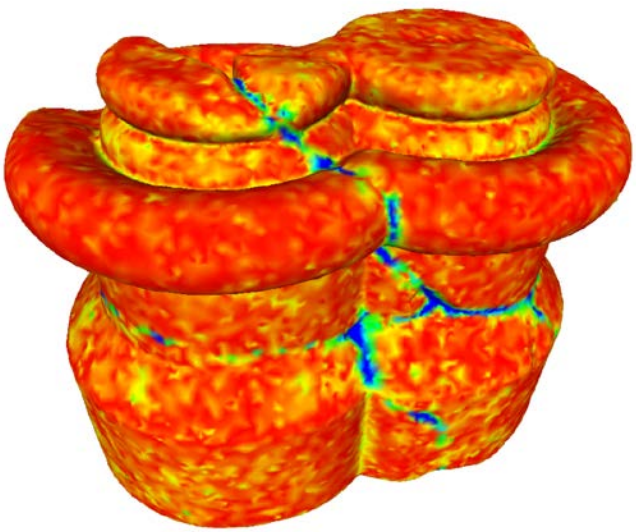
Poisson
re-meshing

Soft Union

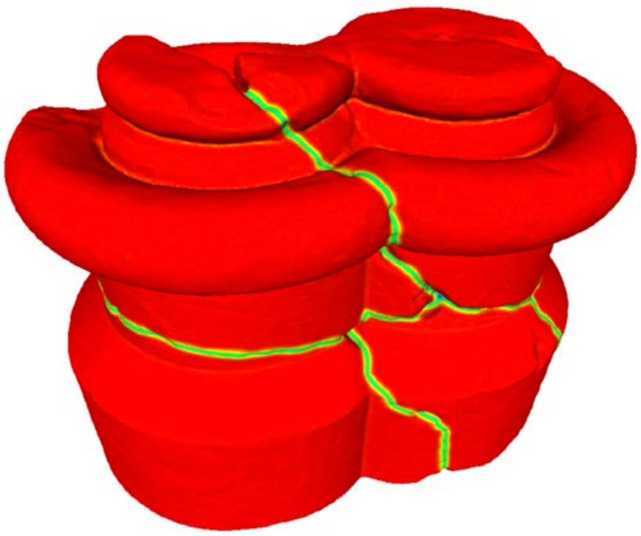
Precision

Distance from the original fractured mesh:

(red denotes zero distance)



Poisson
re-meshing



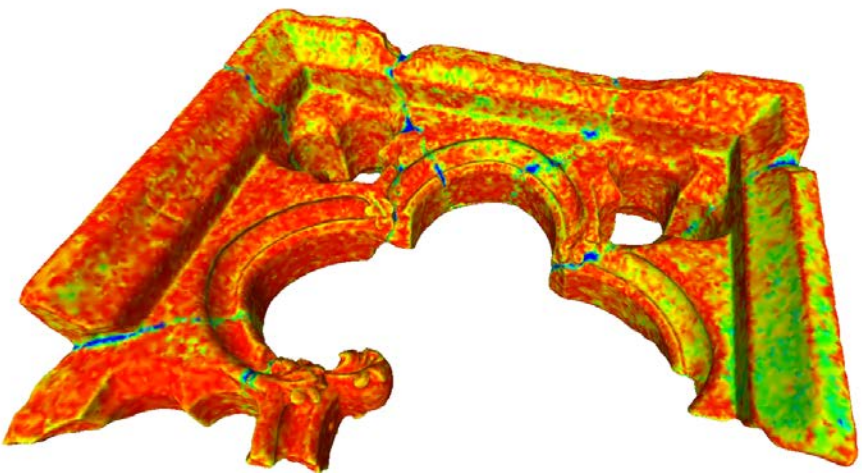
Soft Union

Re-meshing introduces large errors on intact regions.
Soft union preserves the intact regions and deviates only in the fracture lines

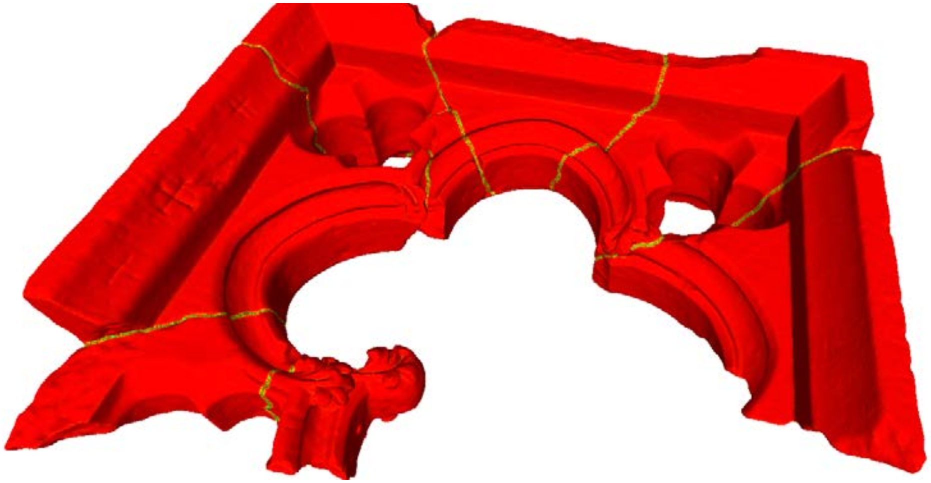
Precision

Distance from the original fractured mesh:

(red denotes zero distance)



Poisson
re-meshing



Soft Union

Re-meshing introduces large errors on intact regions.
Soft union preserves the intact regions and deviates only in the fracture lines

In-depth comparison

- Both approaches work with distance functions.
- Both approaches extract the iso-surface

In-depth comparison

- Both approaches work with distance functions.
- Both approaches extract the iso-surface
- **Re-meshing** approach:
  (based on Poisson or similar reconstruction)
  - Transform oriented points to a continuous vector field.
  - Find a scalar distance function whose gradient match the vector field.
  - Extract the appropriate iso-surface

In-depth comparison

- Both approaches work with distance functions.
- Both approaches extract the iso-surface
- **Re-meshing** approach:
  (based on Poisson or similar reconstruction)
  - Transform oriented points to a continuous vector field.
  - Find a scalar distance function whose gradient match the vector field.
  - Extract the appropriate iso-surface
- **Soft-Union** approach:
  - Transform input surface to distance function
  - Merge and **locally** change the distance function at the fracture lines.
  - Extract the appropriate iso-surface.

In-depth comparison

- Both approaches work with distance functions.
- Both approaches extract the iso-surface
- **Re-meshing** approach:
  (based on Poisson or similar reconstruction)
  - Transform oriented points to a continuous vector field.
  - Find a scalar distance function whose gradient match the vector field.
  - Extract the appropriate iso-surface
- **Soft-Union** approach:
  - Transform input surface to distance function
  - Merge and **locally** change the distance function at the fracture lines.
  - Extract the appropriate iso-surface.

Soft union preserves the distance function of the existing triangulation

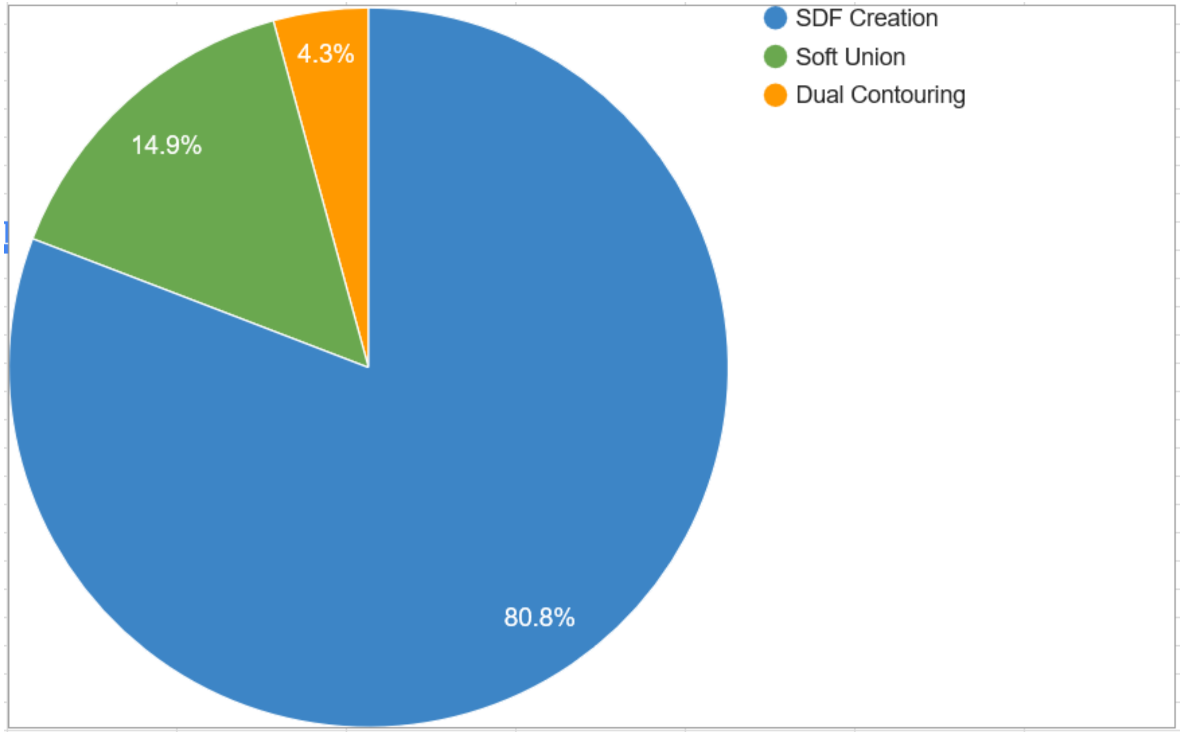
Performance

Input Set	# vertices	Re-meshing	Soft union
Dora Arch	130K	20.9 sec	0.9 sec
Dora Block	280K	43.7 sec	6.7 sec
Column Base	1849K	504 sec	11.9 sec
Embrasure (full)	12533K	> 8 Hours	75.0 sec

Total processing time (excluding disk I/O)

The re-meshing approach did not finish after 8 hours for our largest dataset.

Soft Union Performance Analysis



Percentage of time spend
(measured on the *Embrasure* data set)

The soft-union time is dominated by the conversion to SDF.
The actual soft-union operation is very fast / can be adjusted interactively

Memory Consumption

	Resolution	MBytes
Dora Block	400x397x560	117.4
Dora Arch	525x344x454	133.2
Dora Column	395x294x520	36.9

The voxel dimension was set to 1 cm.
The memory consumption is reasonable, even for high grid resolutions

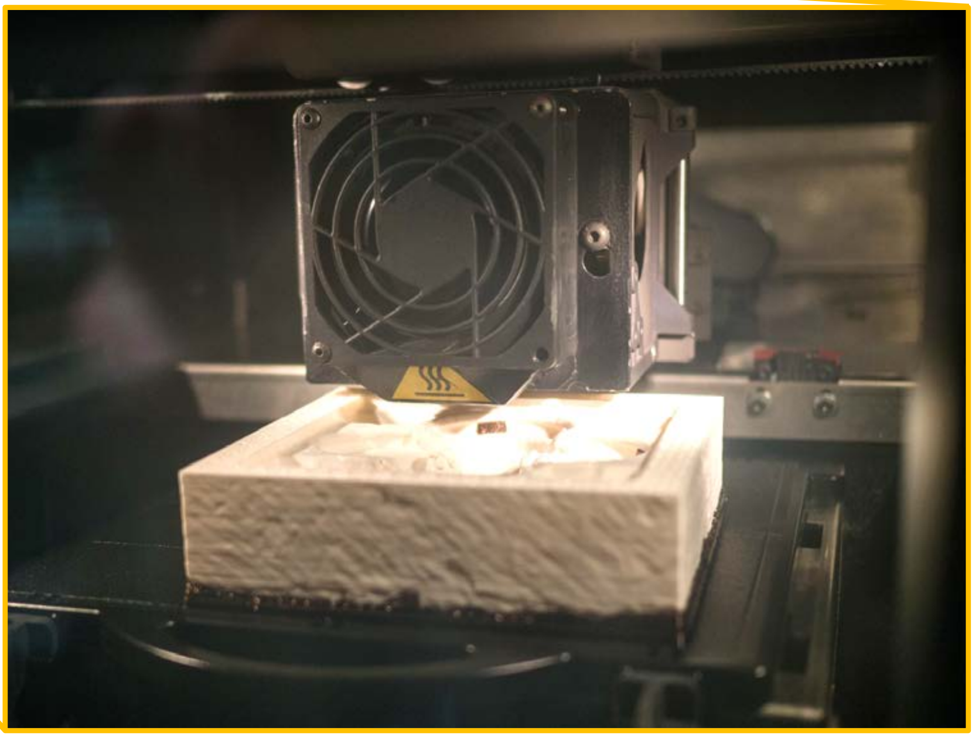
When implementing the algorithm, only one fragment and the accumulated volume has to be in memory.

3D Printing

- **3D printer:** *Stratasys dimension elite*
- **Material:** ABS

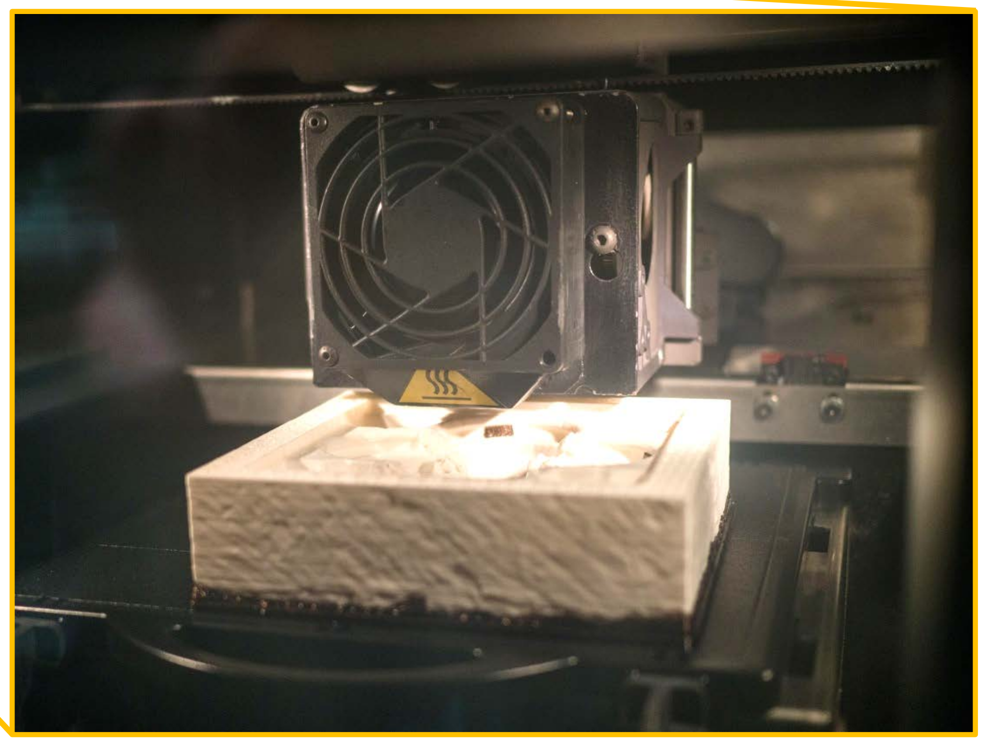
3D Printing

- **3D printer:** *Stratasys dimension elite*
- **Material:** ABS



3D Printing

- **3D printer:** *Stratasys dimension elite*
- **Material:** ABS



After ~20 hours for each shape…

3D Printing

3D Printing

Limitations

- Single smoothing parameter throughout the fracture line
- **Future work:**
  - **Adaptively** adjust the smoothness parameter
  - Do the restoration **interactively**, with a user controlled “brush” with varying radius

Conclusions

- We have presented a novel approach to:
  - Create **watertight meshes** from the output of digital reassembly and completion algorithms.
  - Conceal **fracture lines** (to the desired extend)
- Based on a volumetric **implicit surface** representation and a **soft union** operation.
- We have evaluated the results on **3D printing** of CH artifacts.

- **Final conclusion:** One more valuable tool in the overall ***digital restoration*** toolbox.

Thank you for your attention!

- More information:
  - http://www.presious.eu
  - http://www.cgv.tugraz.at/

Acknowledgements:
This research was partially supported by the EU FP7 programme for research, technological development and demonstration grand agreement no 600533.