# Meshing

Umberto Castellani
Robotics, Vision and control

# 3D modelling from reality pipeline

**Acquisition**

↓

**Registration**

↓

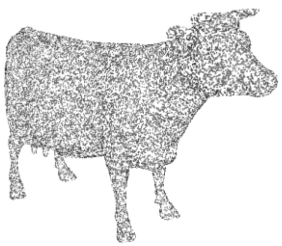
**Meshing**

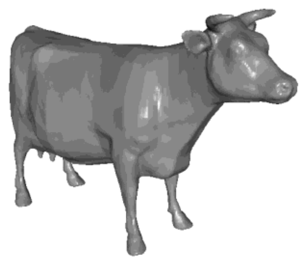
↓

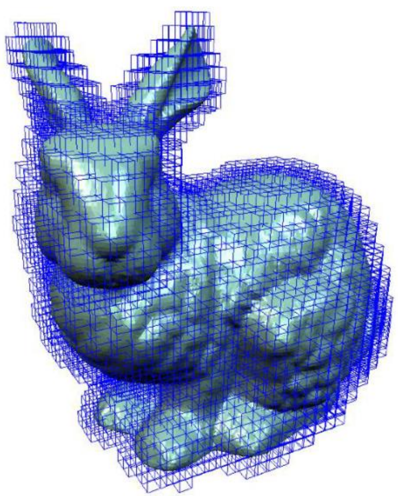
**Advances**

# Overall aim

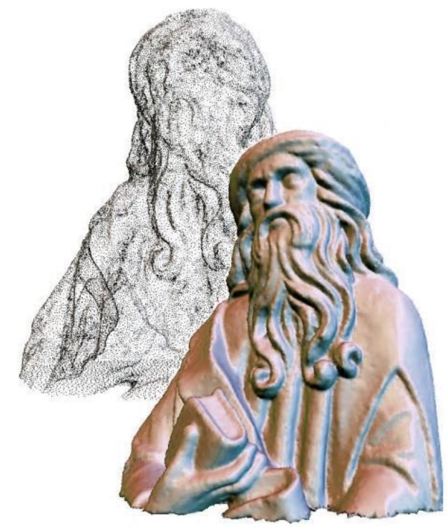
- Once views are aligned a merging procedure is required to obtain a single mesh of the entire object



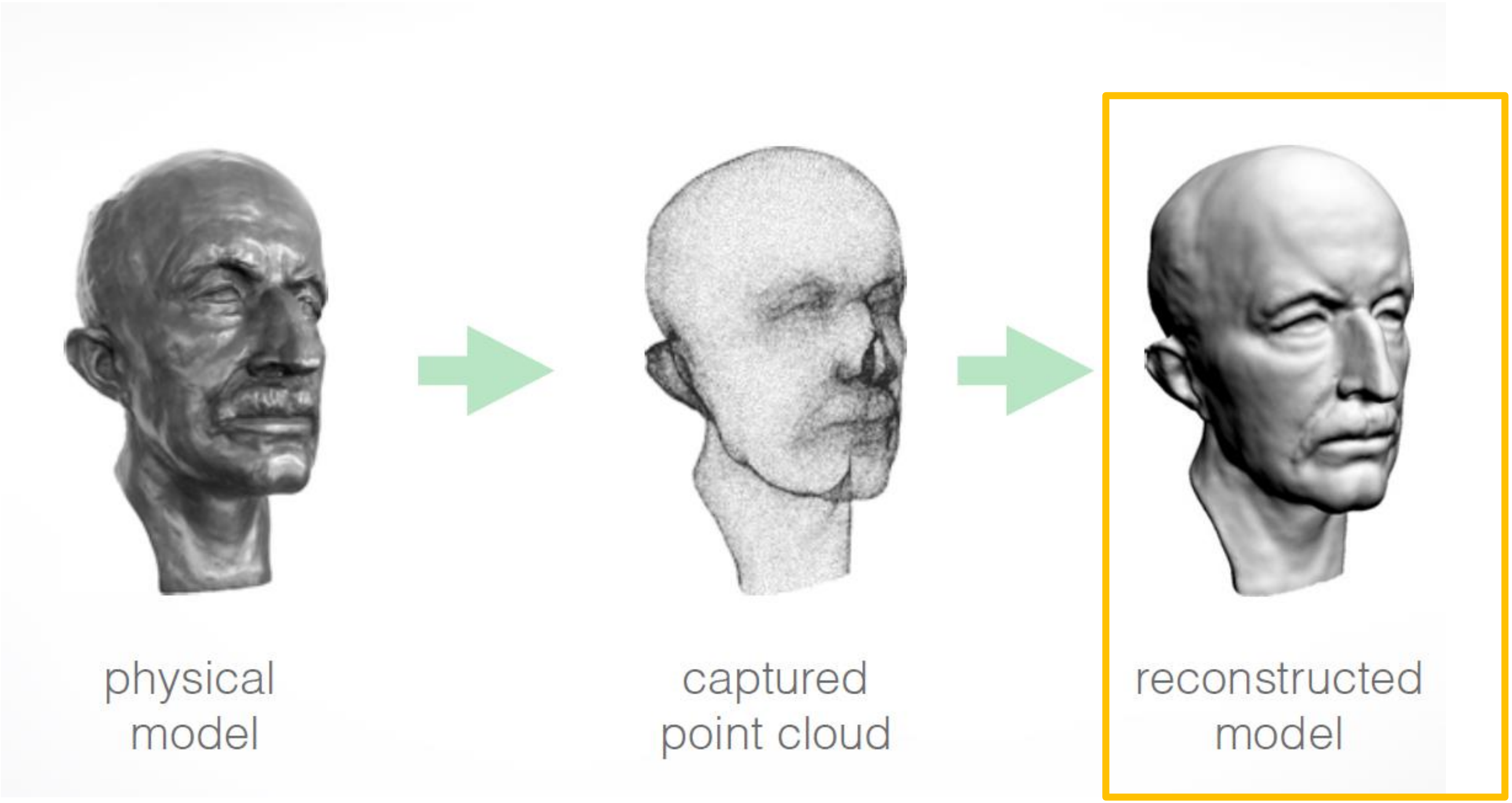




**Marching cube**



# Surface reconstruction




**Polygonal mesh**

# Input

## Set of irregular sample points

- with or without normals
- examples: multi-view stereo, union of range scan vertices

## Set of range scans

- each scan is a regular quad or tri-mesh
- normal vectors can be obtained through local connectivity

# Surface reconstruction

- **Two approaches:**

**Explicit**

Local surface connectivity estimation

Point interpolation

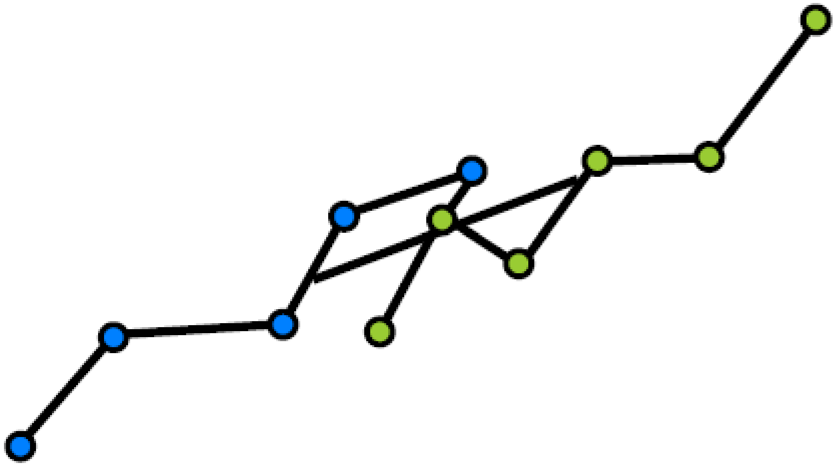
**Implicit**

Signed distance function estimation

Mesh approximation

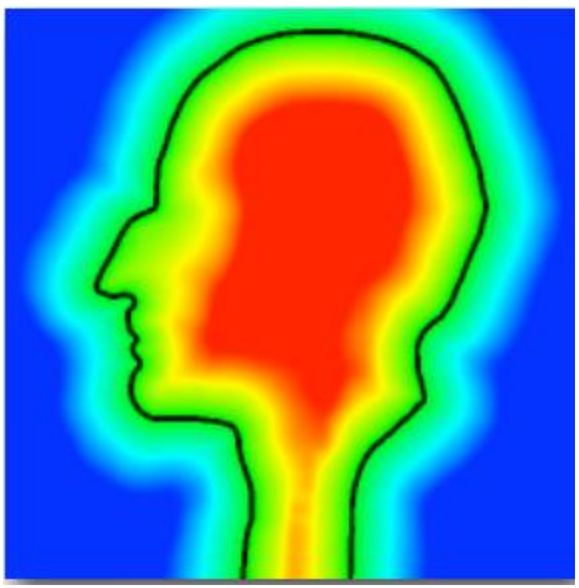
# Explicit methods

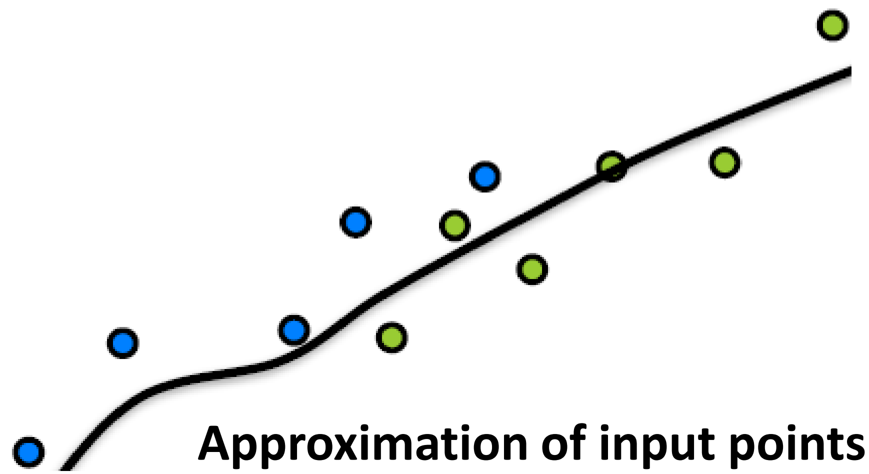
- Connect sample points by triangles
- Exact interpolation of sample points
- Bad for noisy or misaligned data
- Can lead to holes or non-manifold situations



# Implicit methods

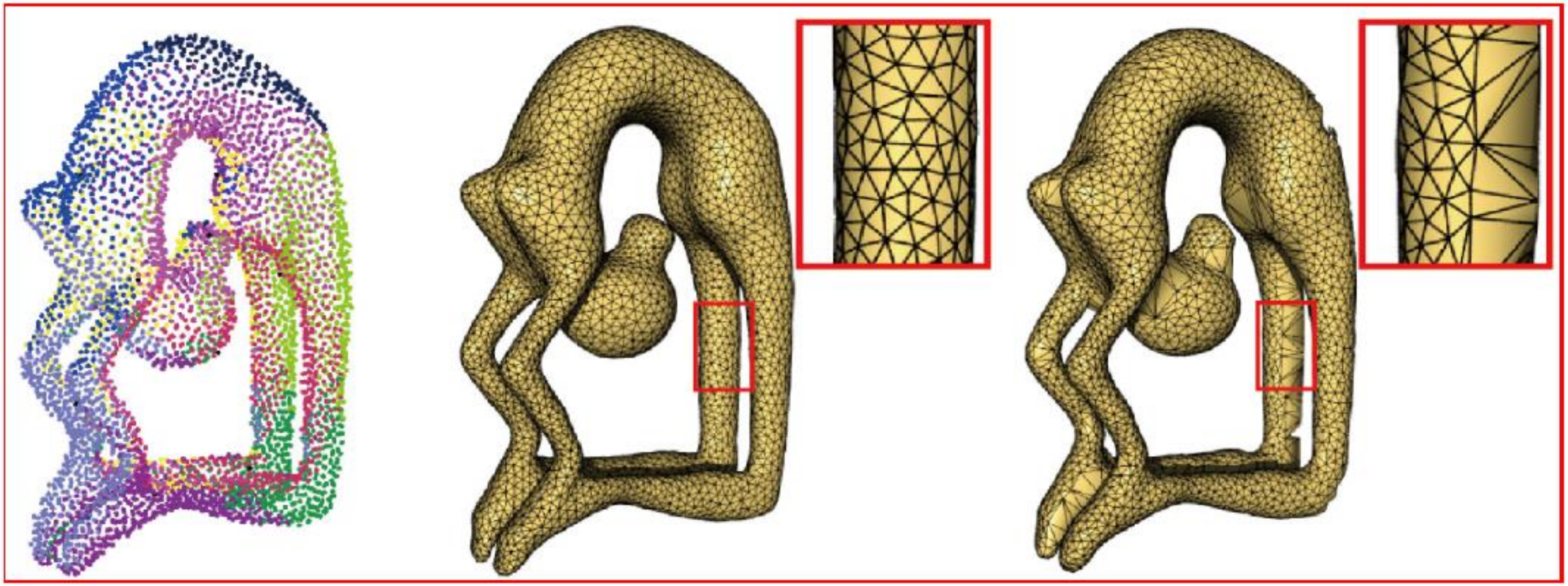
1. estimate a signed distance function(SDF);
2. extract 0-level set mesh using Marching Cubes






Ouput is a **Watertight manifold** by construction !

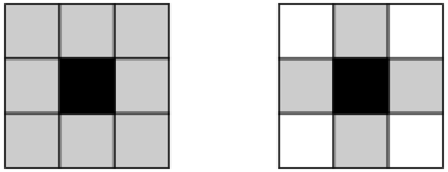
# Explicit vs Implicit



Input Implicit Explicit

# Esplicit method

- Mesh reconstruction from **range image**,
  - **Idea**: points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood



**8-neigh** **4-neigh**

- Zippering range scans,
  - **Idea**: "Zipper" several scans to one single model

# Mesh from range image

- Points are on a regular grid where the connectivity can be inherited from the pixel neighbourhood,

**BUT…**

1) Not all the pixels on the range image are the projection of a point on the 3D space!

   ➡ a binary mask can be used to define valid points,

2) Nearby pixels should not correspond to nearby points on the 3D space!

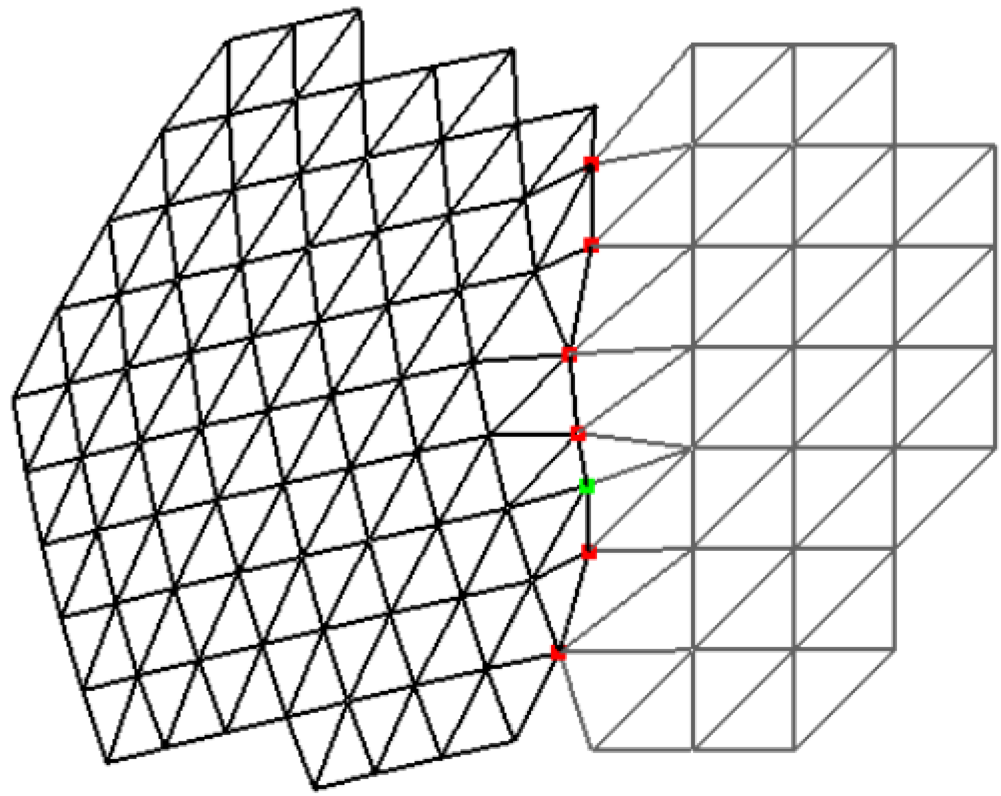
   ➡ a robust strategy can be used to remove long edges.

**See Matlab script available from lab section!**

# Zippering range scans

- “Zipper” several scans to one single model

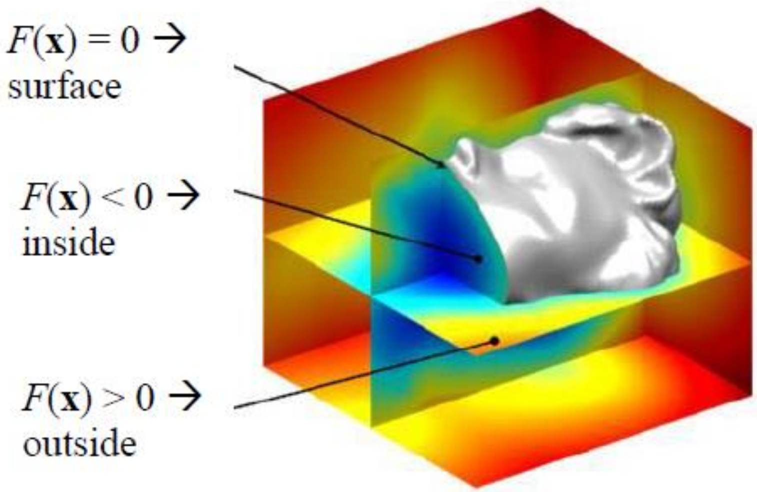
1. Project & insert boundary vertices
2. Intersect boundary edges
3. Discard overlap region
4. Locally optimize triangulation

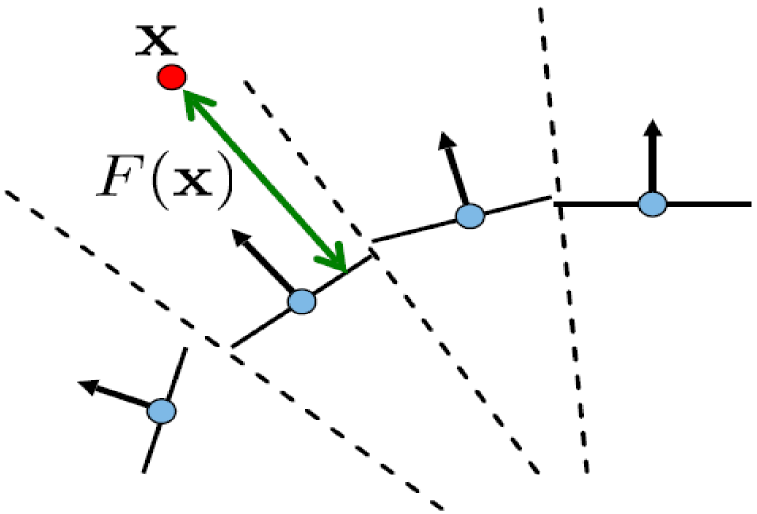


**Not much used in practice!**

# Implicit methods

- Several methods for signed distance computation (SDF)
  - **Marching Cube**: classical method
  - **Poisson method**: the currently most used method

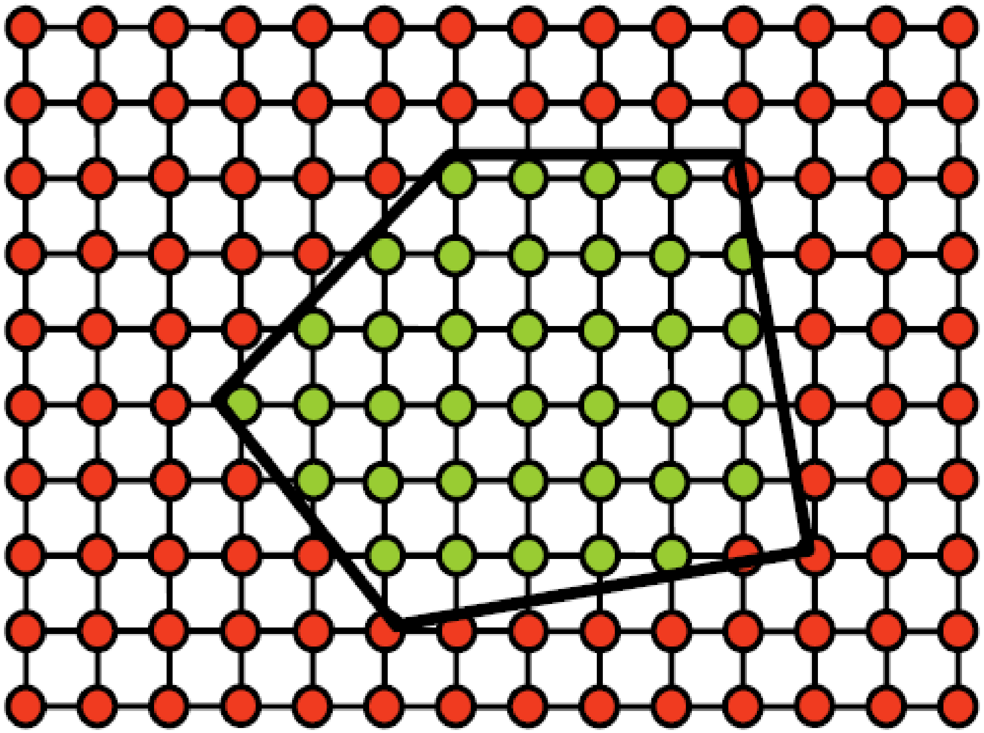



**E.g.: signed distance to the tangent plane of the closest point**

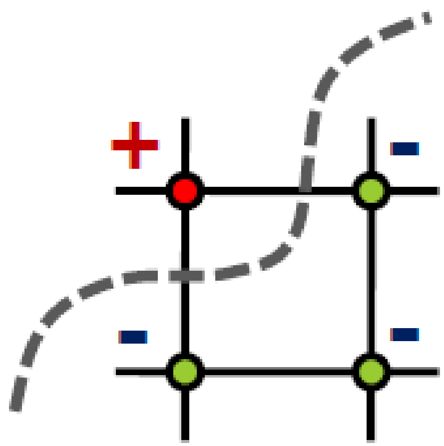
# Marching cube

- **Idea**: sample the SDF



# Marching cube

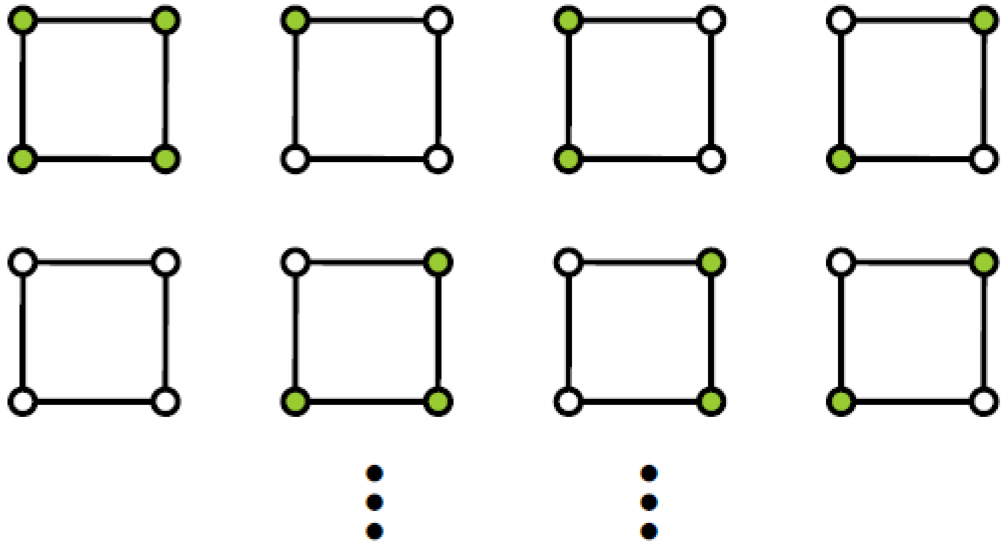
- **Idea**: sample the SDF



$F(\mathbf{x}) < 0$

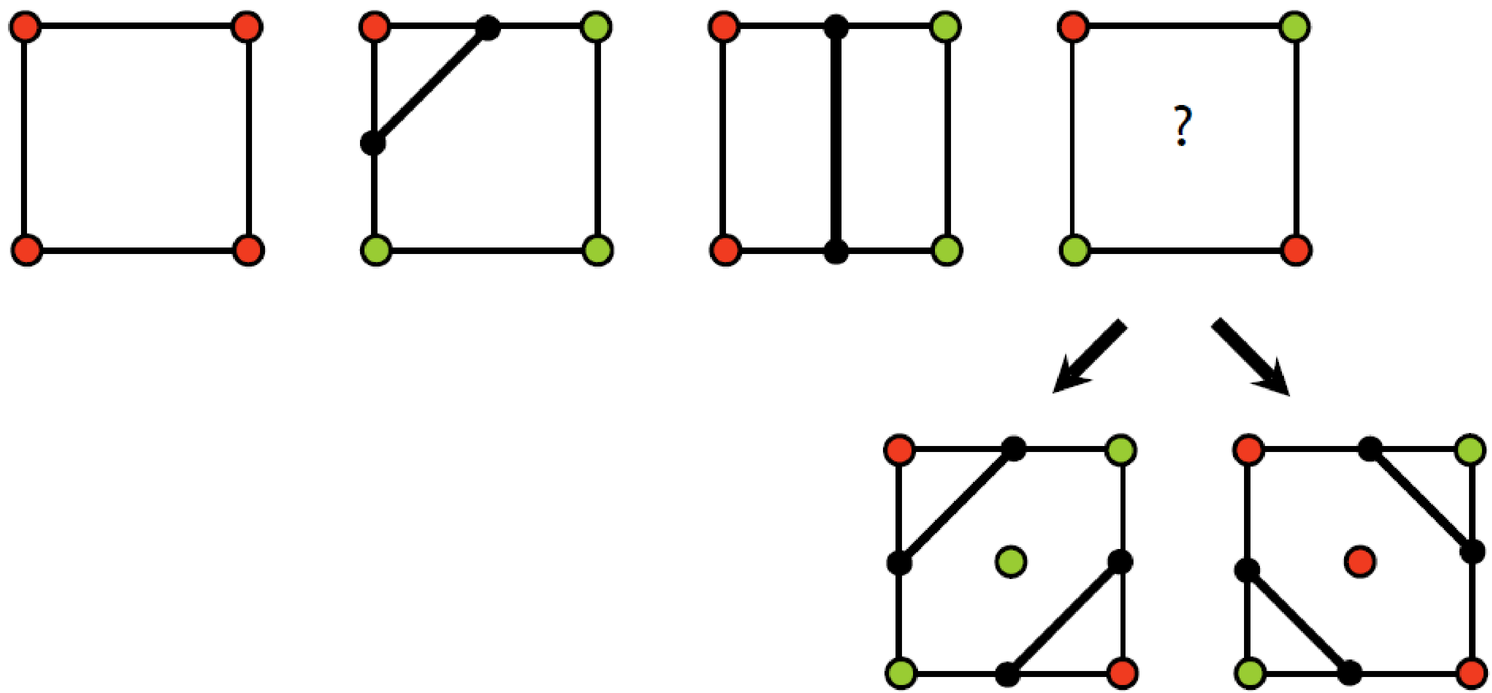
# Marching square

- 16 different configurations in 2D
- 4 equivalence classes (up to rotational and reflection symmetry + complement)

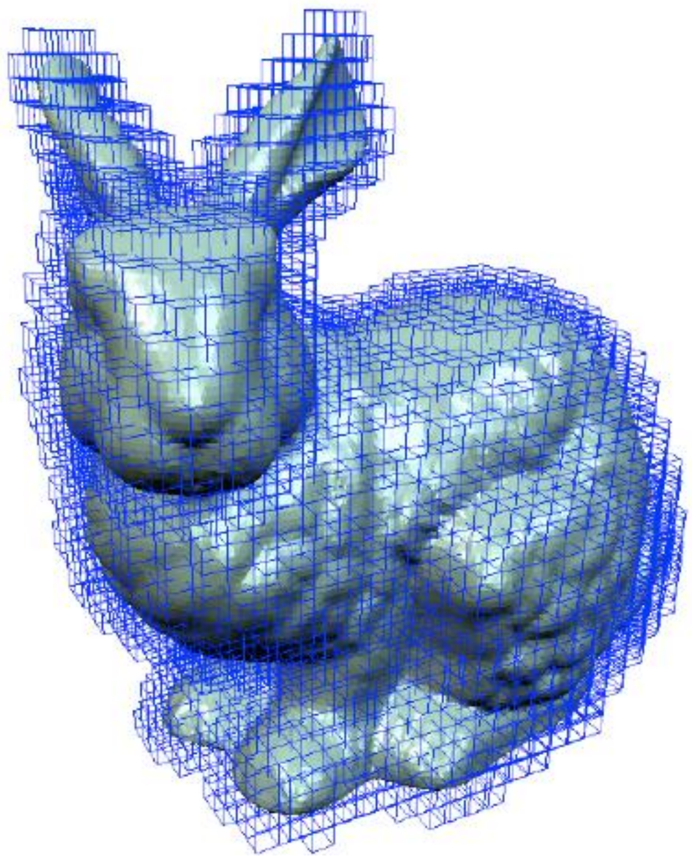


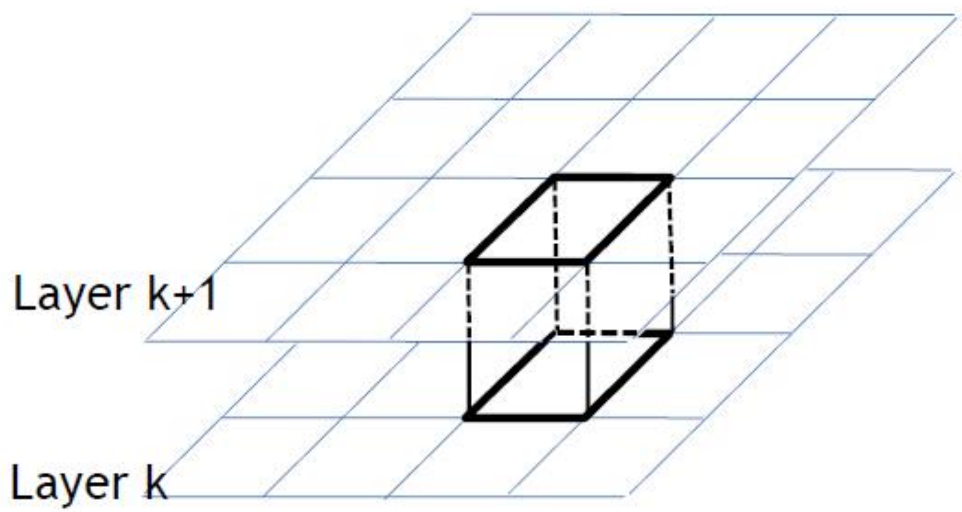
# Marching square

- 4 equivalence classes (up to rotational and reflection symmetry + complement)



# Marching cube



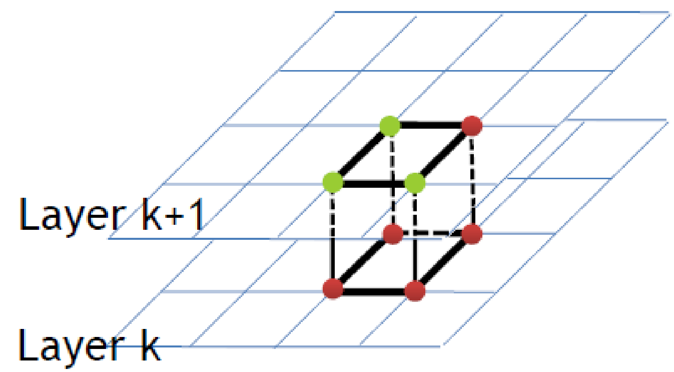



**3D case!**

# Marching cube

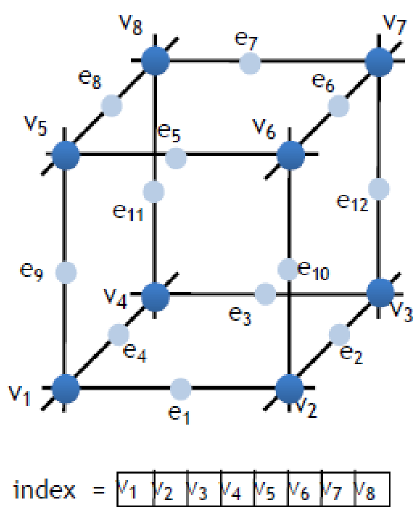
Marching Cubes (Lorensen and Cline 1987)

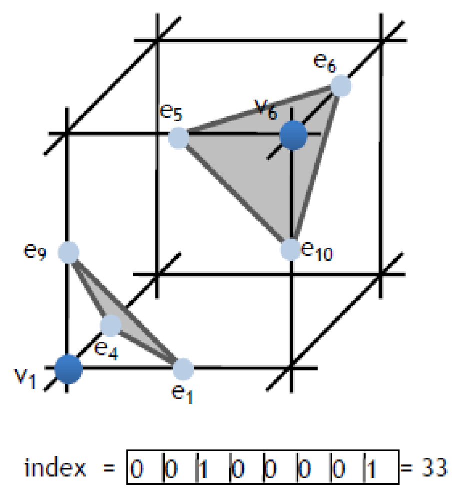
1. Load 4 layers of the grid into memory
2. Create a cube whose vertices lie on the two middle layers
3. Classify the vertices of the cube according to the implicit function (inside, outside or on the surface)



# Marching cube

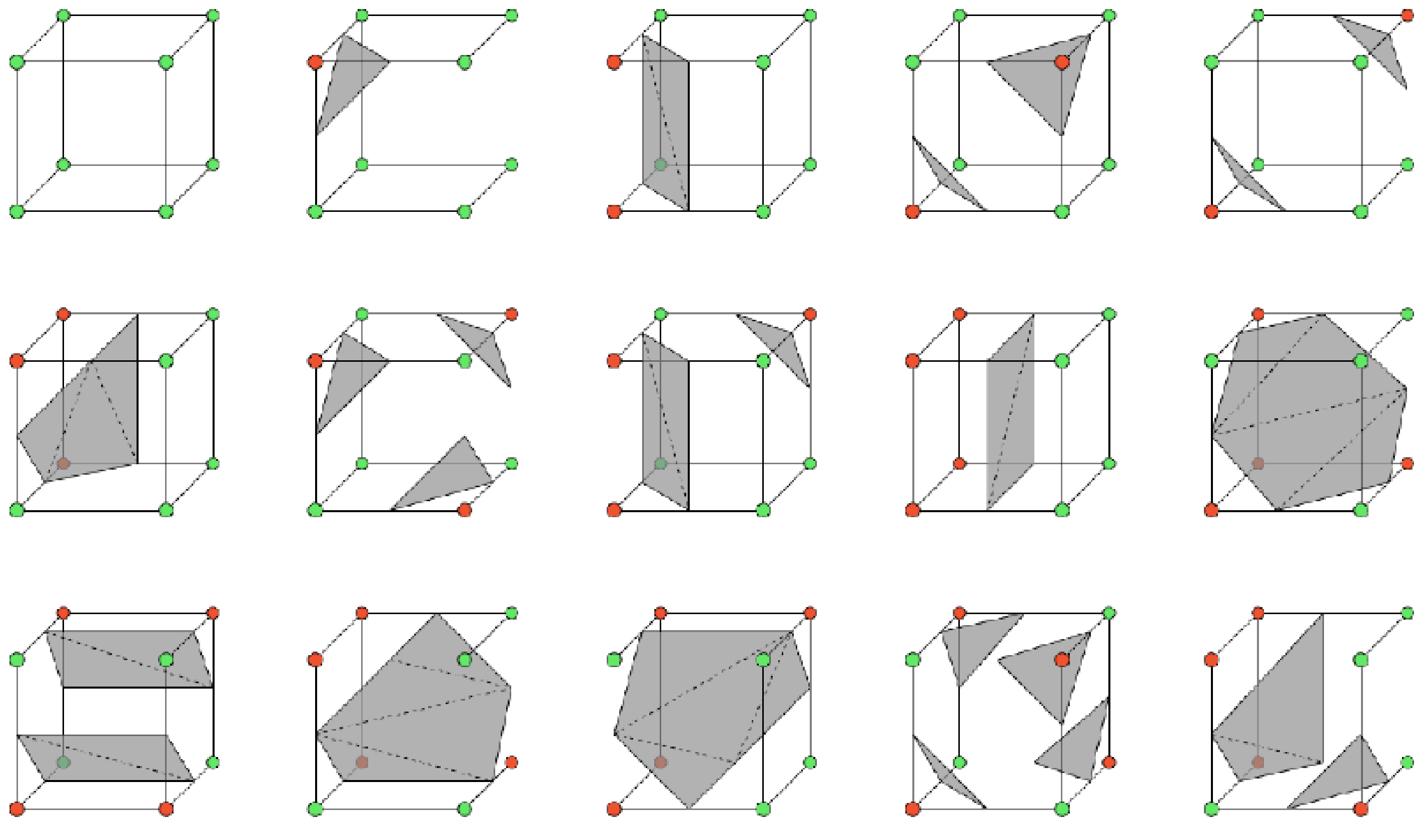
Compute case index. We have $2^8$= 256 cases (0/1 for each of the eight vertices) – can store as 8 bit (1 byte) index.



# Marching cube

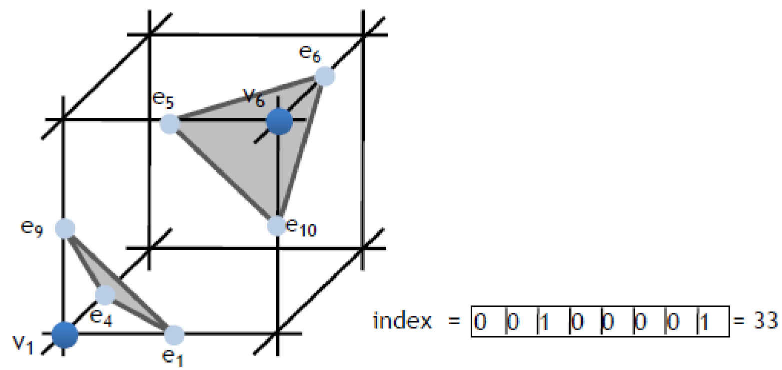
Unique cases (by rotation, reflection and complement)



# Marching cube

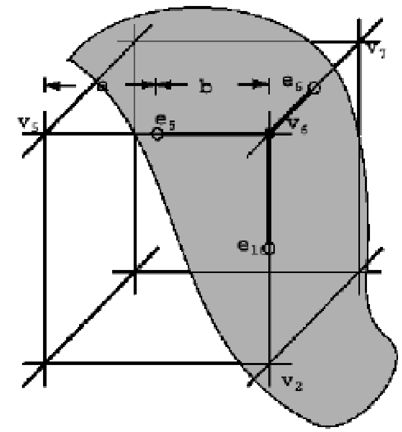
Using the case index, retrieve the connectivity in the look-up table

- Example: the entry for index 33 in the look-up table indicates that the cut edges are $e_1$; $e_4$; $e_5$; $e_6$; $e_9$ and $e_{10}$ ; the output triangles are ($e_1$; $e_9$; $e_4$) and ($e_5$; $e_{10}$; $e_6$).



# Marching cube

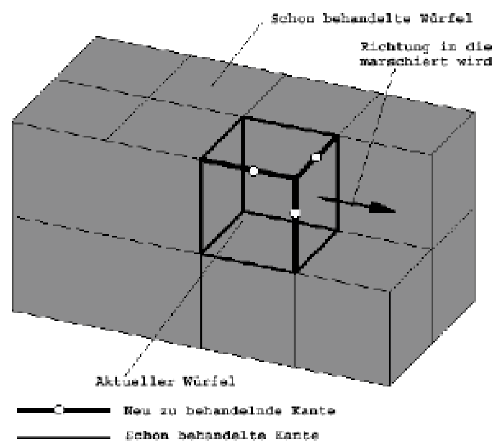
Compute the position of the cut vertices by linear interpolation:




$$\mathbf{v}_s = t\mathbf{v}_a + (1-t)\mathbf{v}_b$$

$$t = \frac{F(\mathbf{v}_b)}{F(\mathbf{v}_b) - F(\mathbf{v}_a)}$$

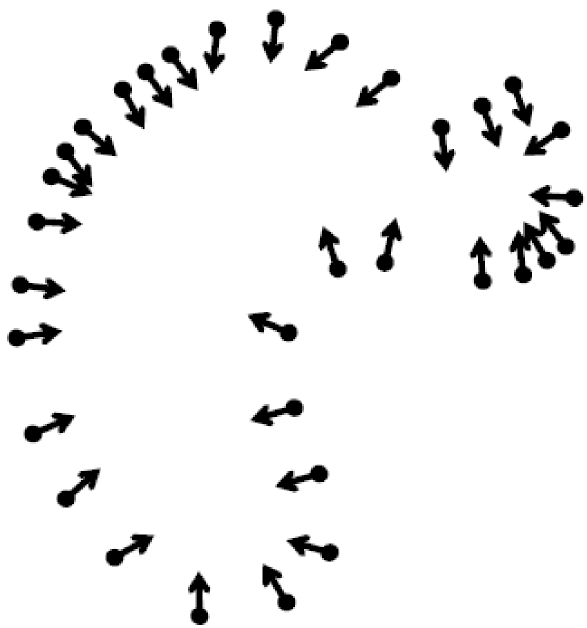
Move to the next cube



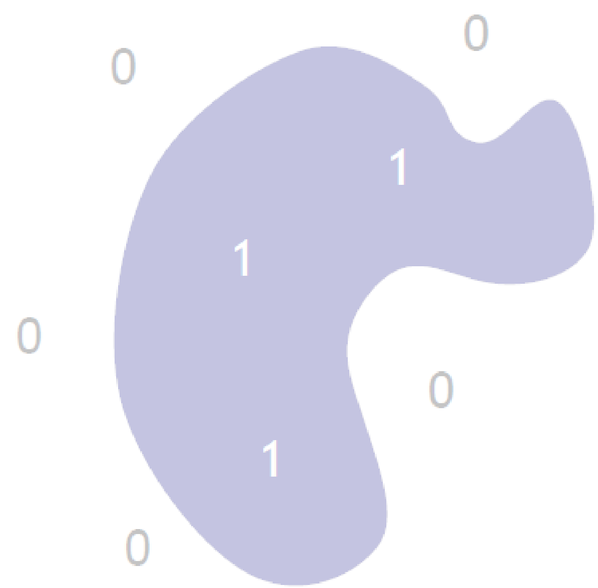
# Poisson method

- Global fitting of an **indicator function** using Partial Differential Equation (PDE),

# Poisson method



Oriented points

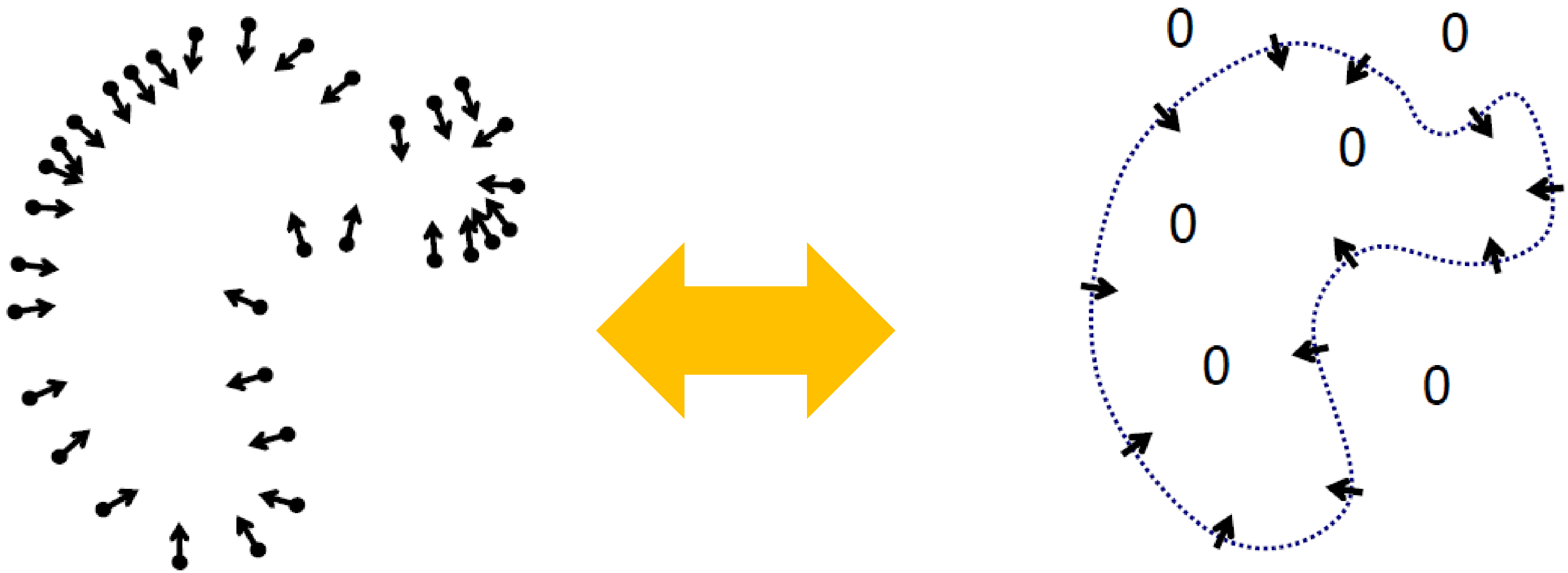



Indicator function

$\chi_{\mathcal{M}}$

We don't know the indicator function ☹

# Poisson method




Oriented points

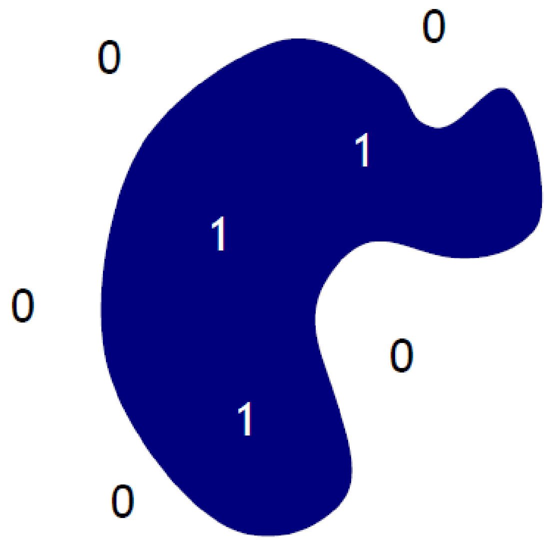
Indicator gradient

$$\nabla \chi_{\mathcal{M}}$$

But we can estimate its gradient! ☺

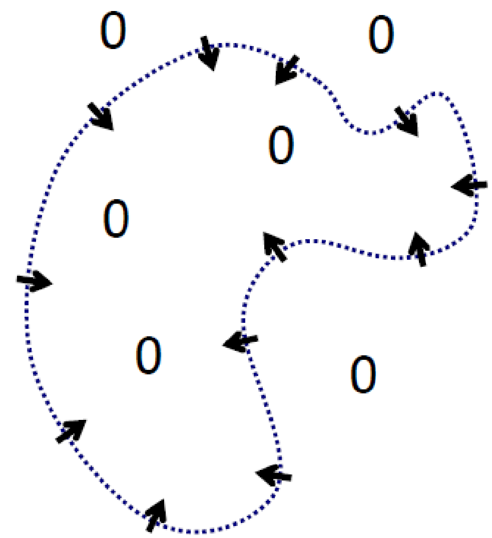
# Poisson method

Oriented points




Indicator function

$\chi_{\mathcal{M}}$




Indicator gradient

$\nabla \chi_{\mathcal{M}}$

Reconstruct $\chi$ by solving the Poisson equation

$$\Delta X_M = \nabla \cdot (\nabla X_M)$$

# Poisson method

**See Meshlab excercise in Lab!**