Representations

Xiao-Ming Fu

Outline

- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

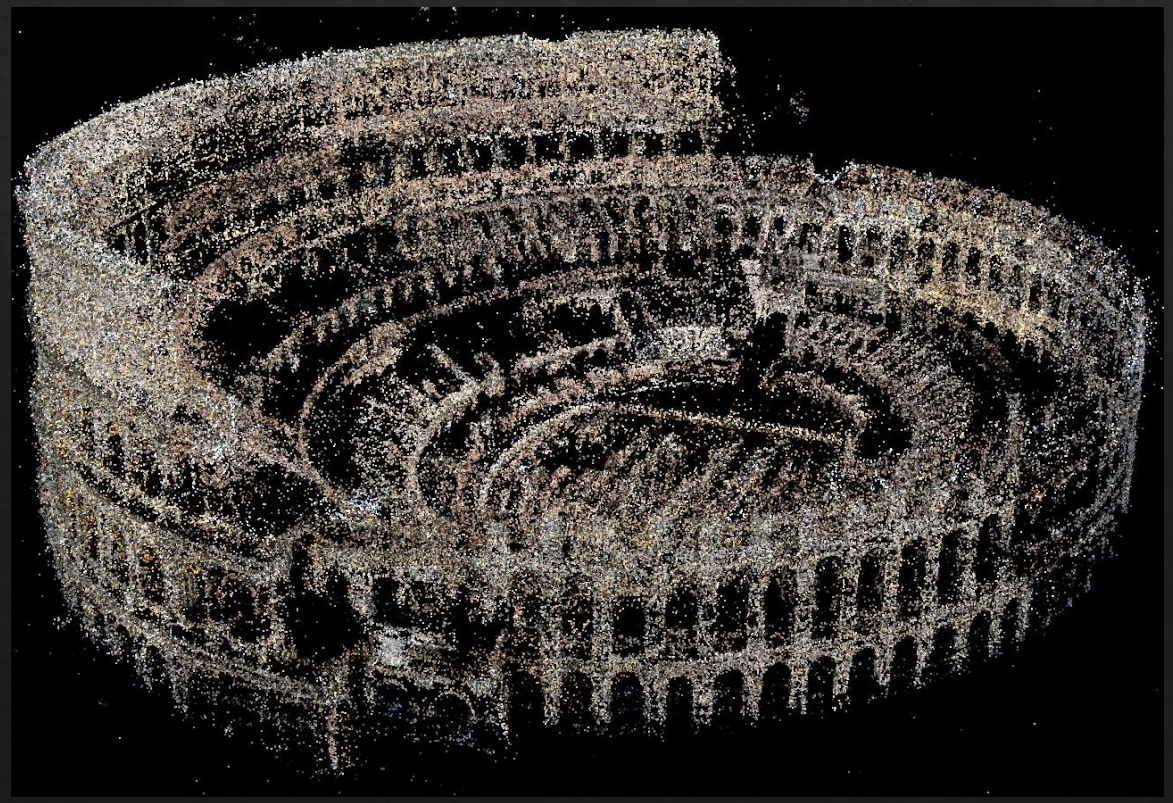
Outline

- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

Point cloud

$$\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_{N_v}\}, \boldsymbol{p}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \mathcal{R}^3$$

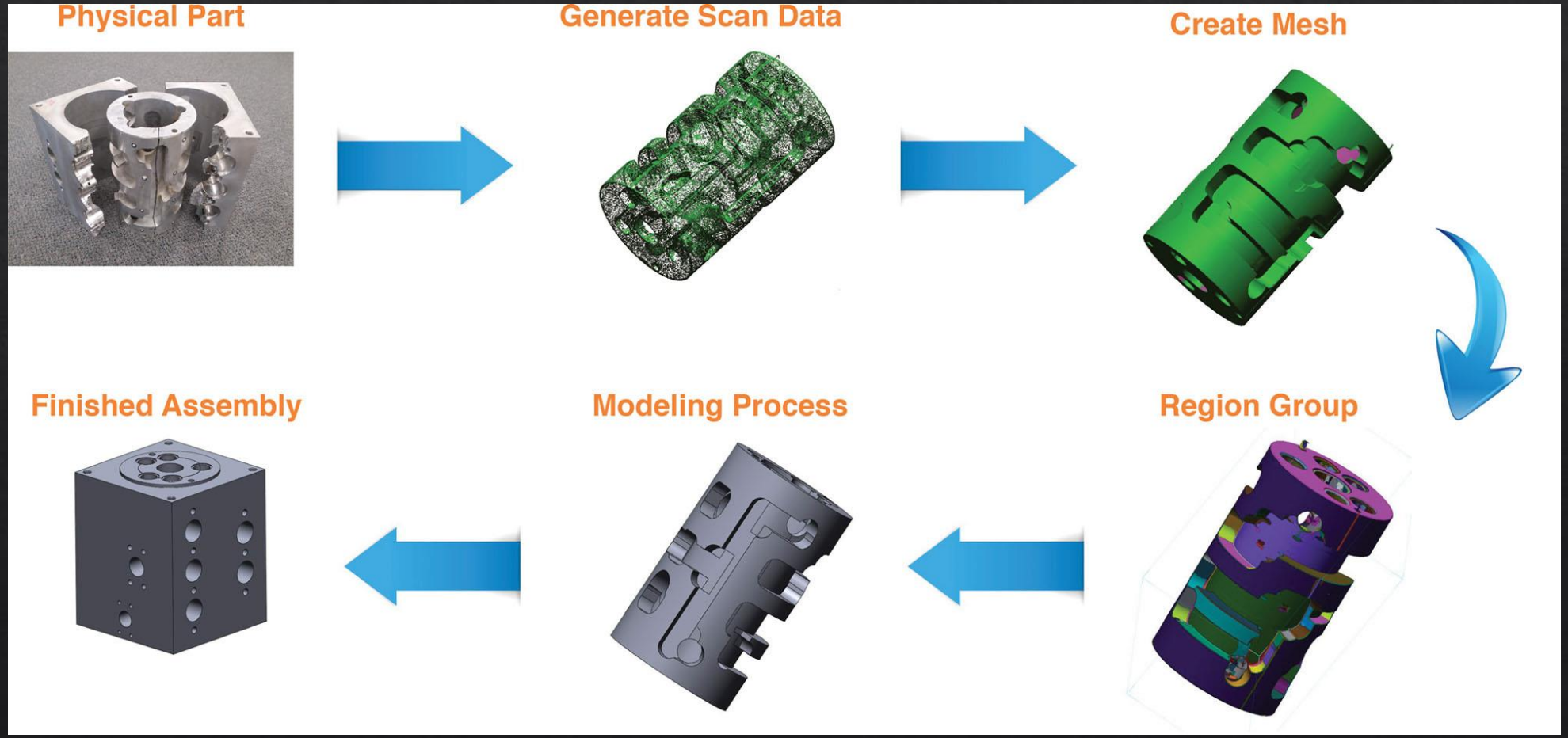
A set of data points in some coordinate system



3D scanners

Applications - scanners

- Reverse engineering



Applications - scanners

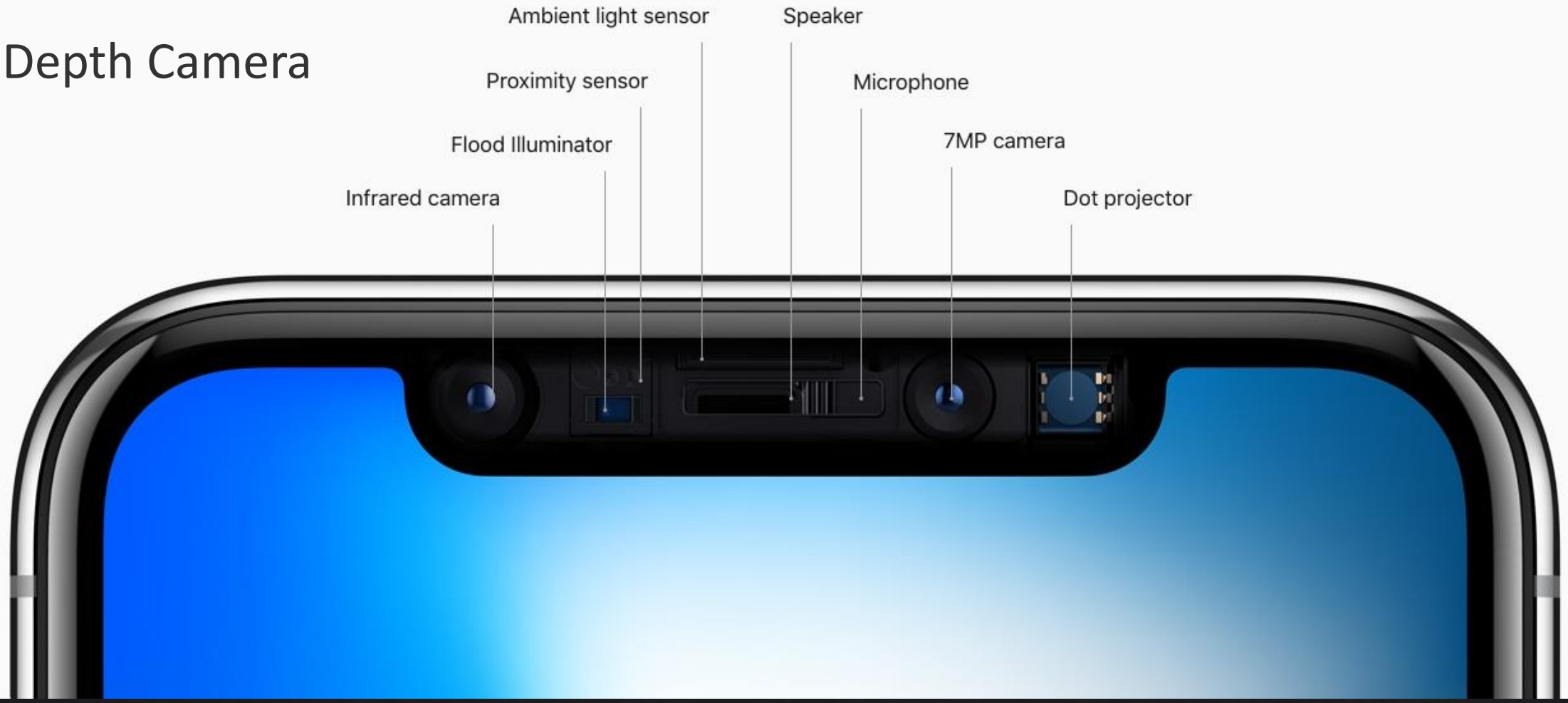
- Digital preservation of cultural heritage sites and objects

Depth camera

- Depth camera: Kinect, RealSense, IPhone X, ……

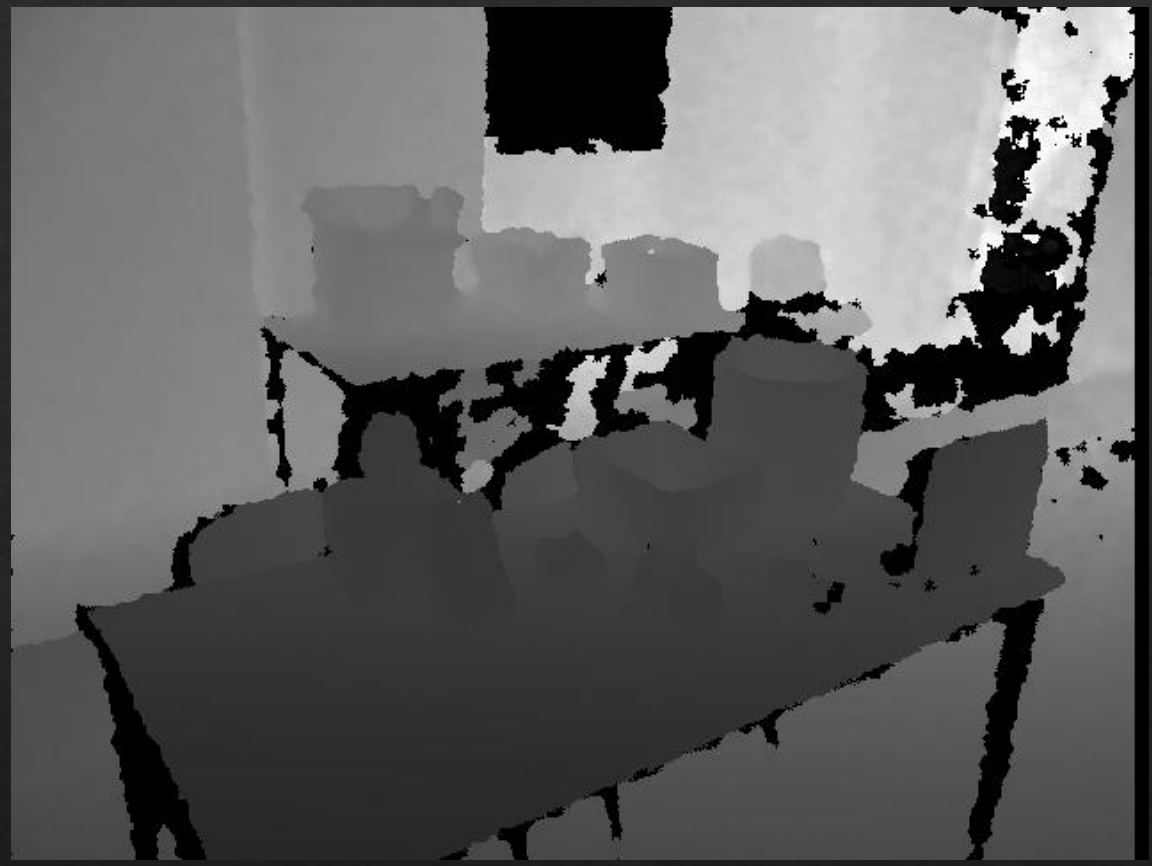
Depth camera

True Depth Camera



Depth Image

Gray image: pixel represents distance from camera.
Nearer is brighter.



Real capture by Kinect.
Nearer is darker.

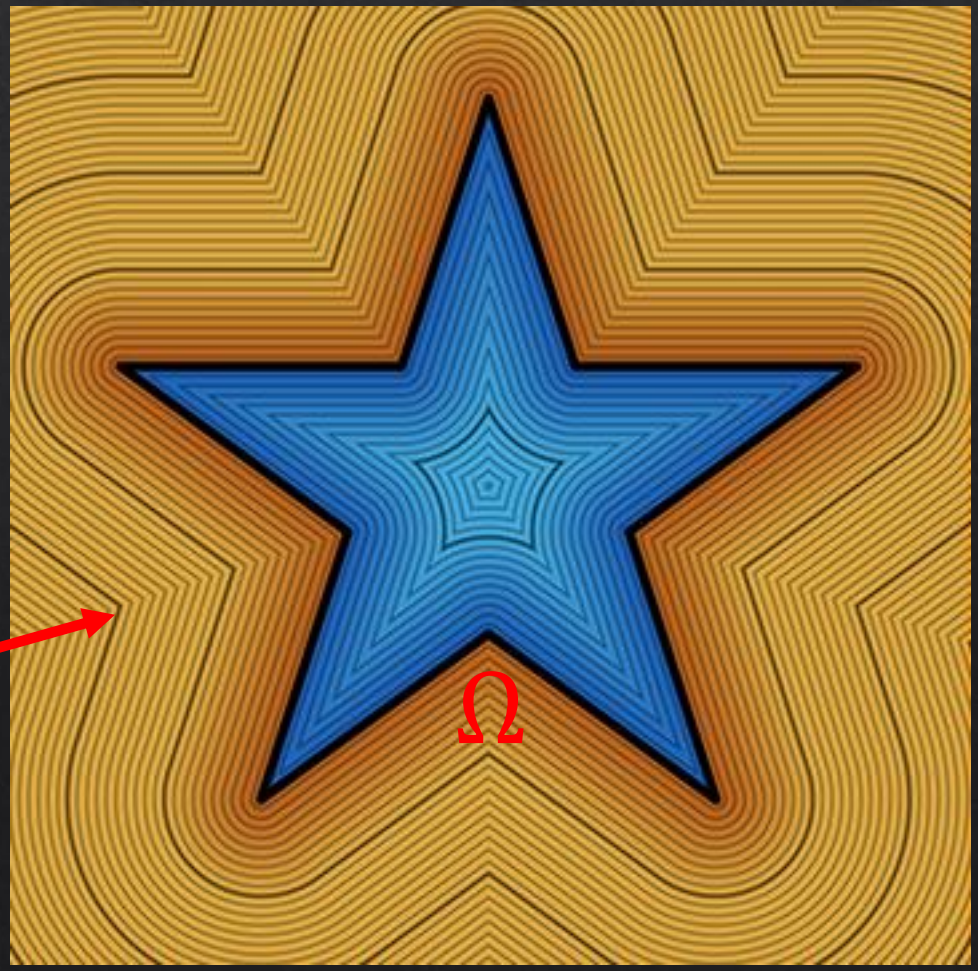
Outline

- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

Signed distance field

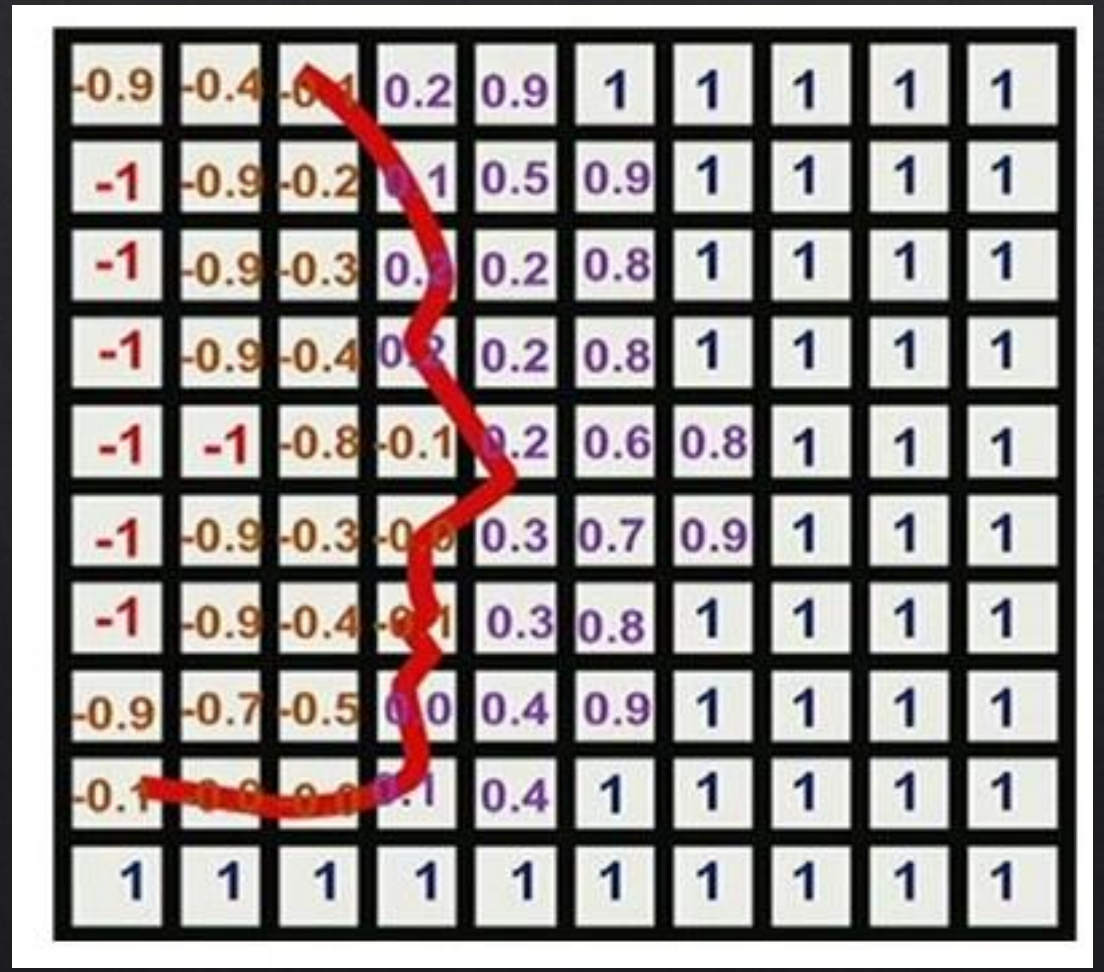
- The distance of a given point x from the boundary of Ω:

$$f(x) = \begin{cases} -d(x, \partial\Omega), & \text{if } x \in \Omega \\ d(x, \partial\Omega), & \text{if } x \in \Omega^c \end{cases}$$



Truncated signed distance field (TSDF)

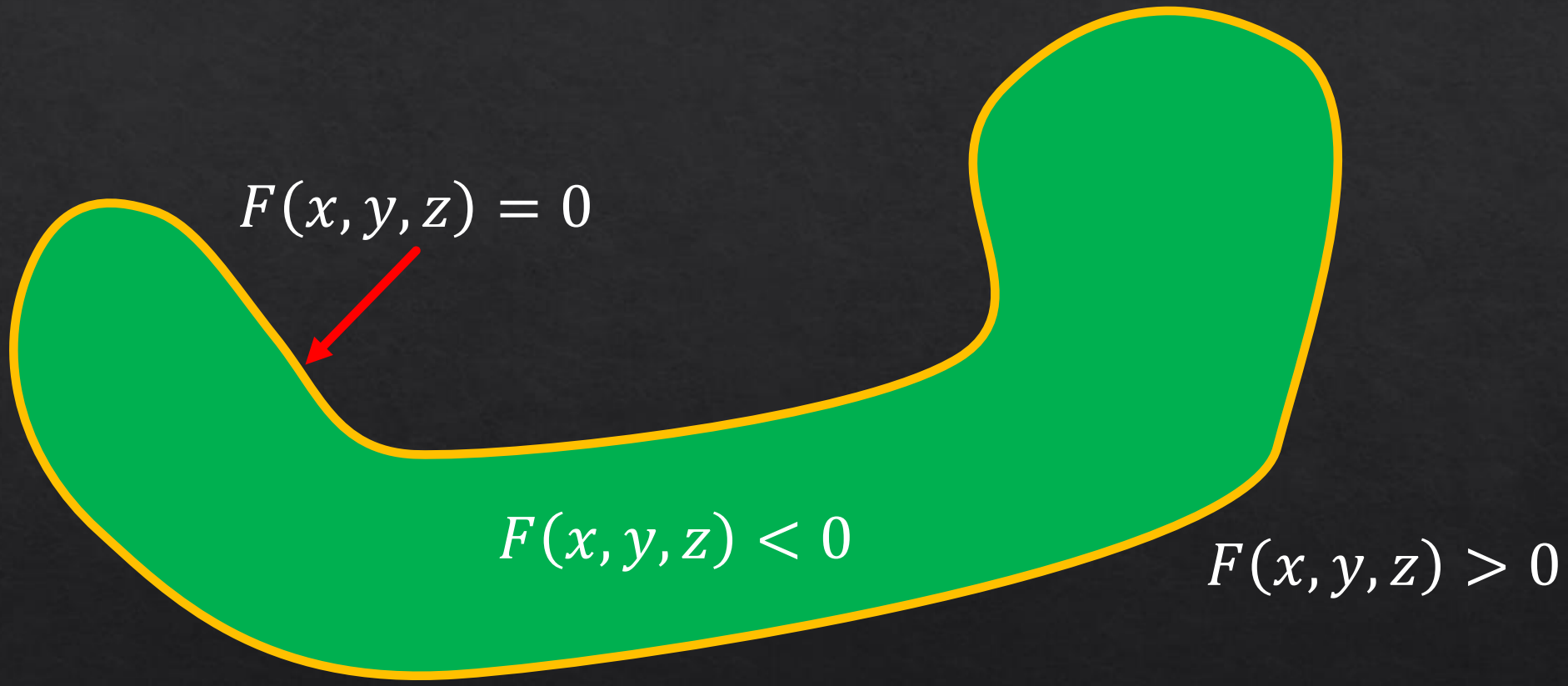
- Less memory than SDF



Outline

- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

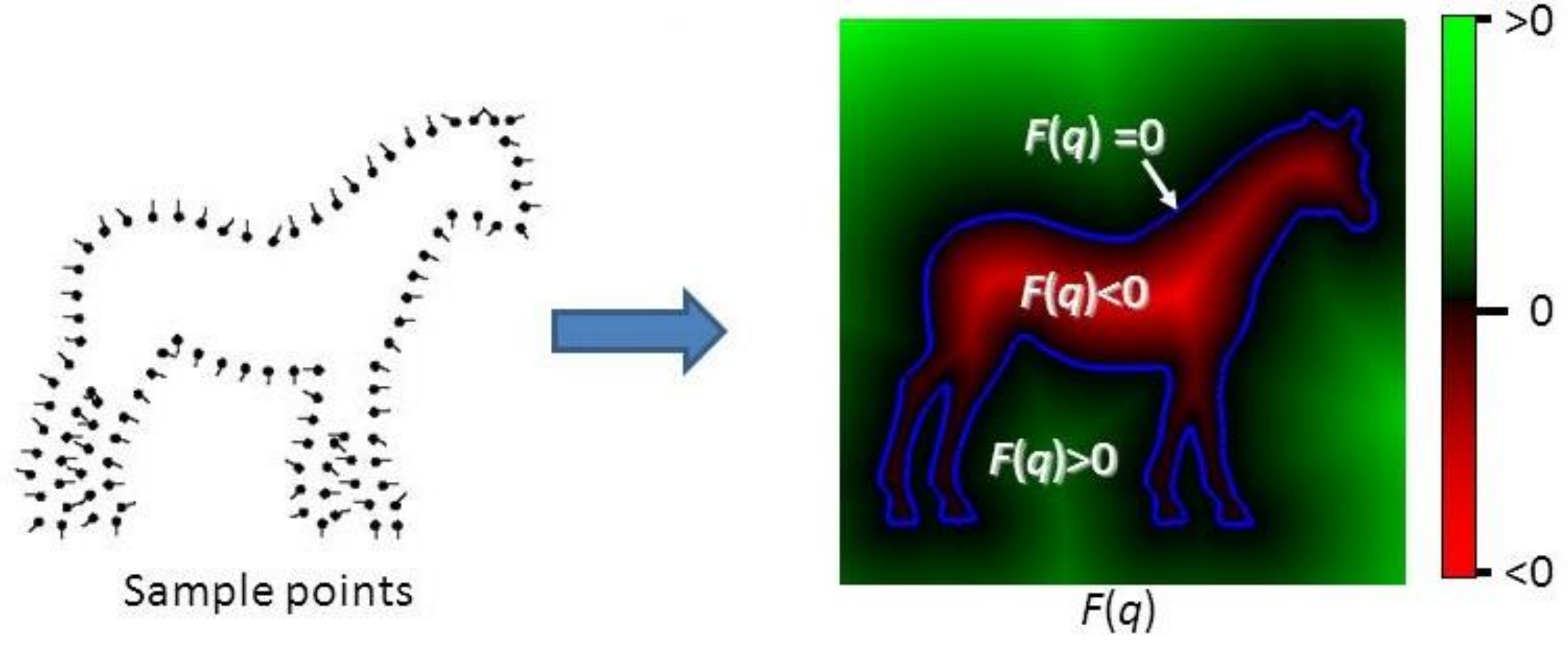
Implicit function



- Singed distance field is also an implicit function.

Implicit function

- Surface reconstruction

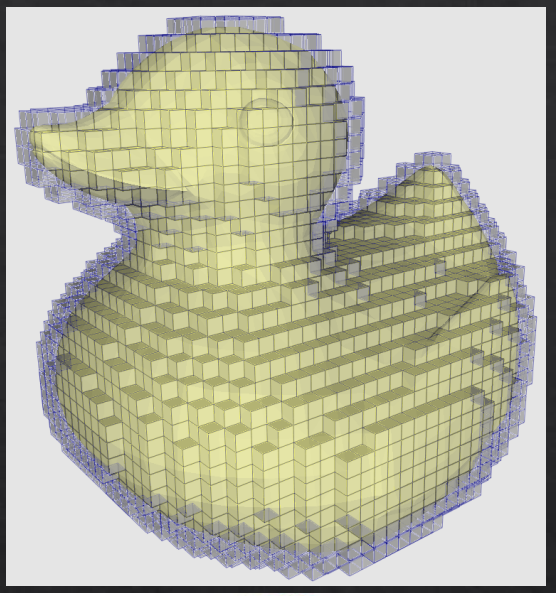


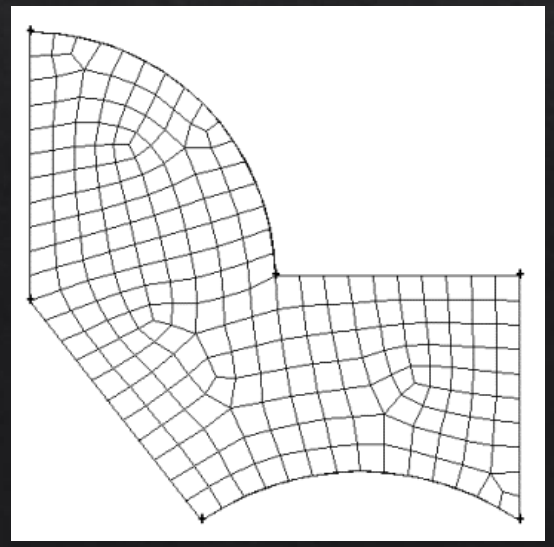
Outline

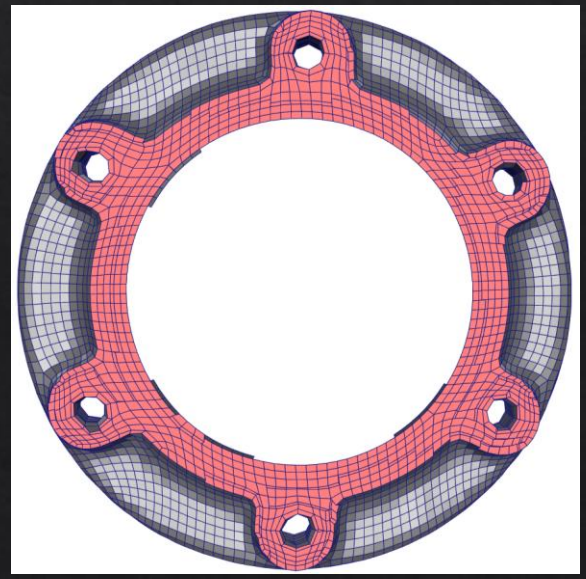
- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

Grid

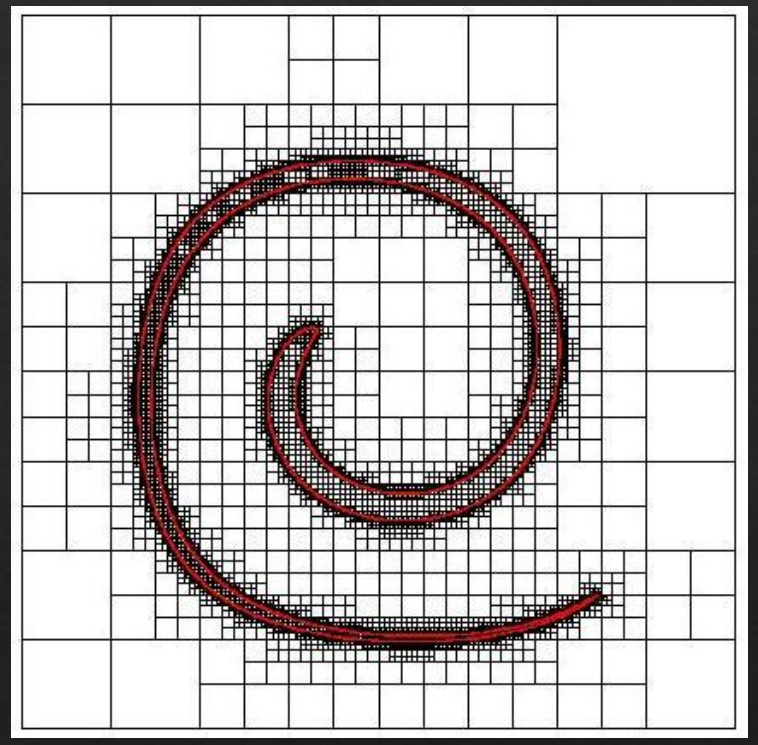
- Image
- Voxel
- Quad mesh
- All-Hex mesh



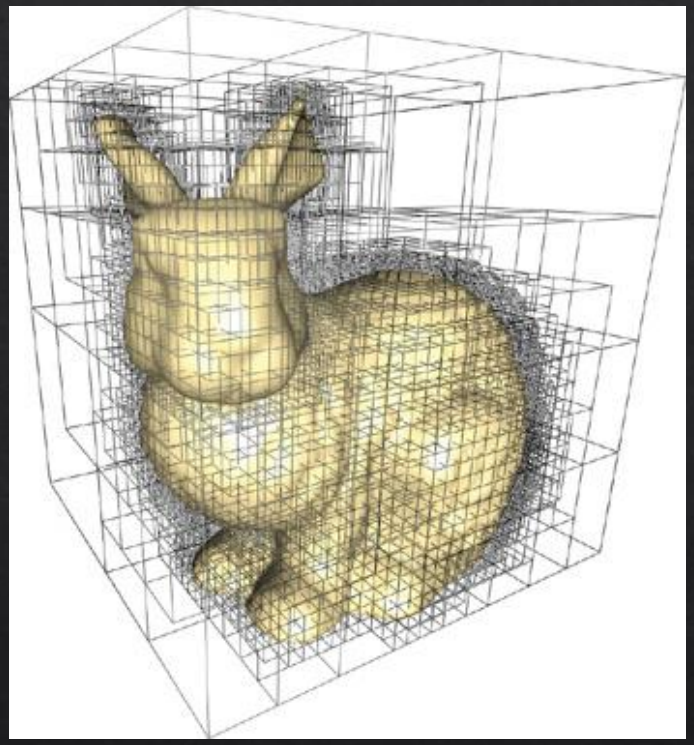




Adaptive grid - hierarchical octree



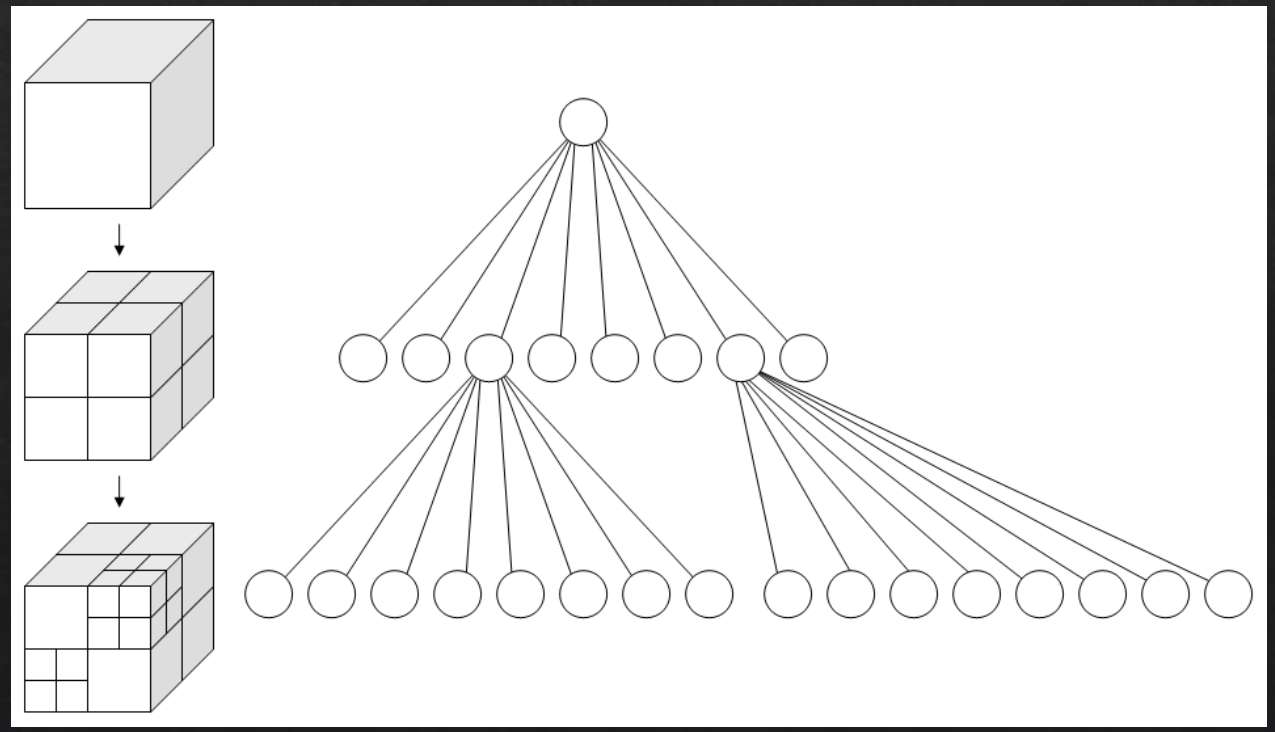
2D case



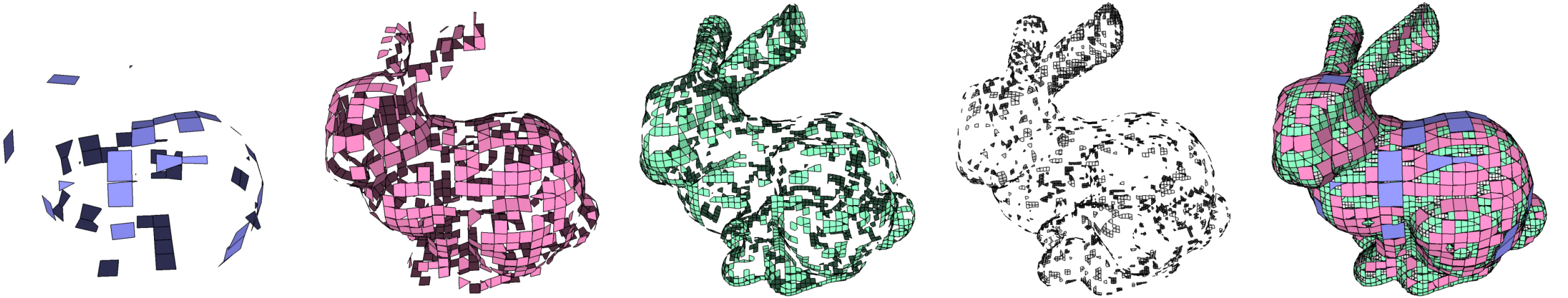
3D case

Partitioning rule

- The commonly-used octant partitioning depends on:
  - (1) the existence of the shape inside the octant
  - (2) the partitioning is performed until the max tree depth is reached.



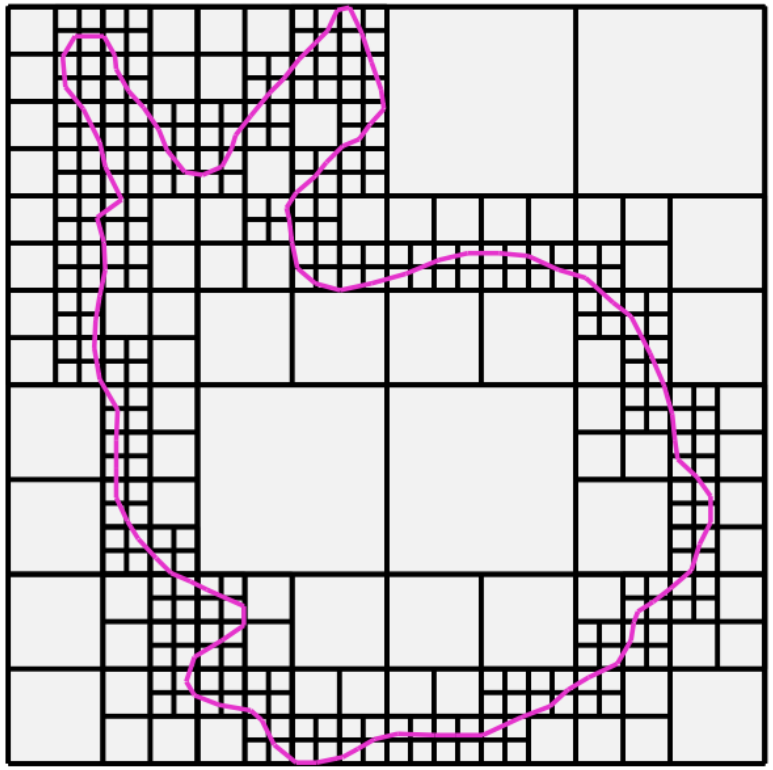
Patch-based octree

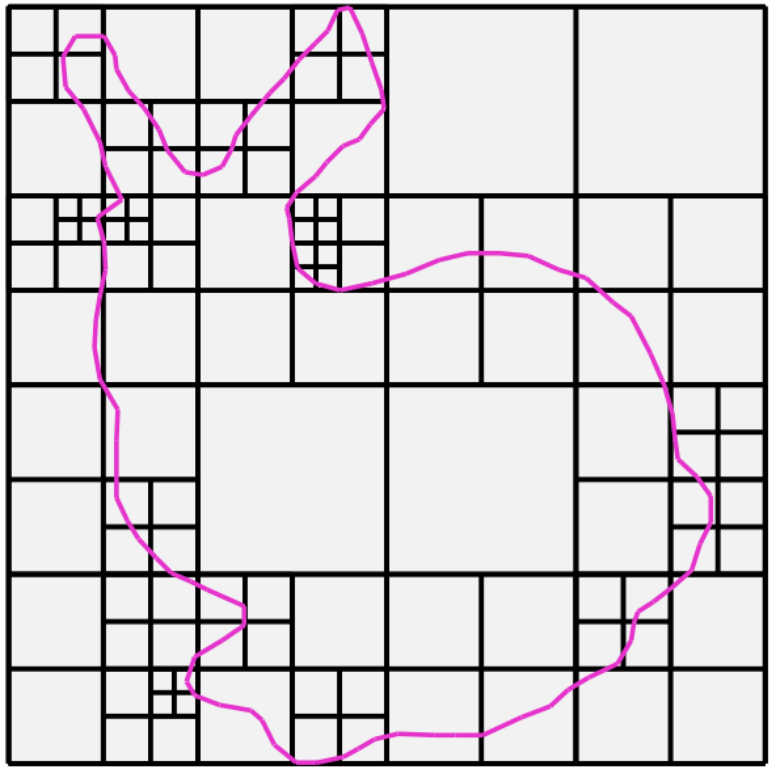


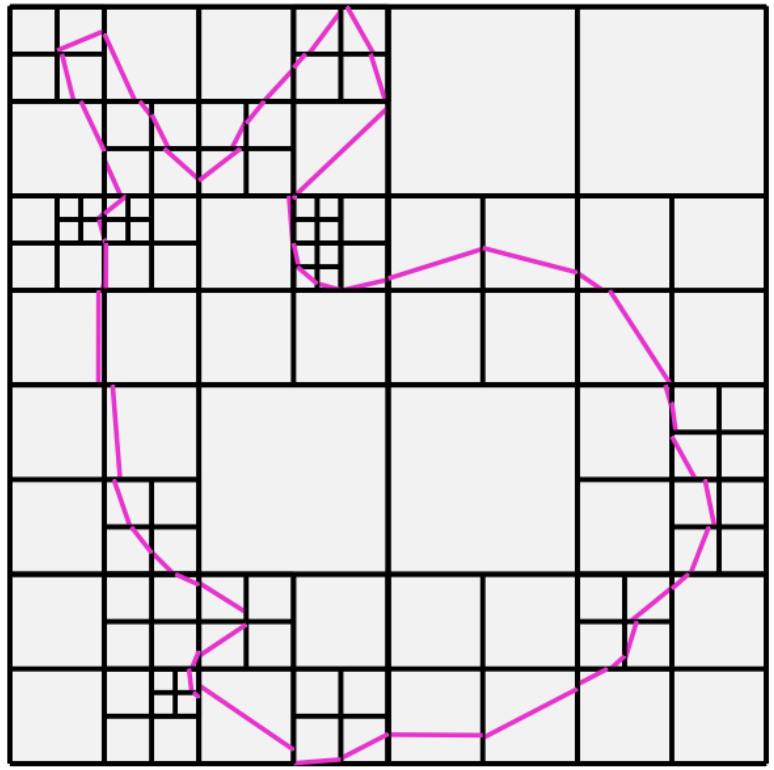
(a) 4th-level (b) 5th-level (c) 6th-level (d) 7th-level (e) all non-empty leaf nodes

The partitioning rule of the octree is:
For any octant O which is not at the max depth level, subdivide it if the local surface S_O restricted by it is not empty and the Hausdorff distance $d_H(S_O, P_O)$ larger than a predefined threshold.

Patch-based quadtree







Paper: Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes

Outline

- Point cloud
- Signed distance field
- Implicit function
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Mesh
  - Triangle, Tetrahedron
  - Data structure
    - Halfedge
  - File format
    - Obj, Off

Triangle Mesh

- A collection of triangles
  - without any particular mathematical structure
- Each triangle: a segment of a piecewise linear surface representation
- Geometric component
  - Discrete vertices: $V = \{v_1, \dots, v_{N_V}\}$
- Topological component
  - Triangle: $F = \{f_1, \dots, f_{N_F}\}$
  - Edge: $E = \{e_1, \dots, e_{N_E}\}$

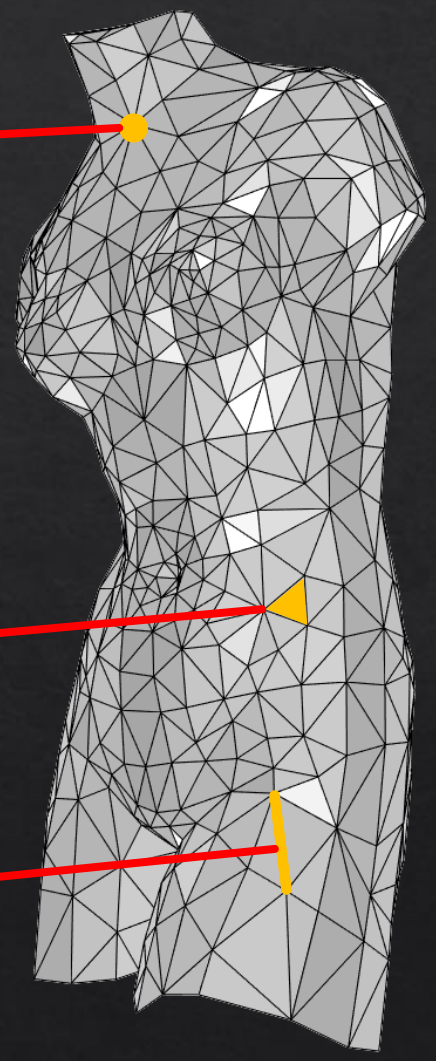
Triangle Mesh

$$\boldsymbol{p}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \in \mathcal{R}^3$$

$$\mathcal{P} = \{\boldsymbol{p}_1, \dots, \boldsymbol{p}_{N_v}\}$$

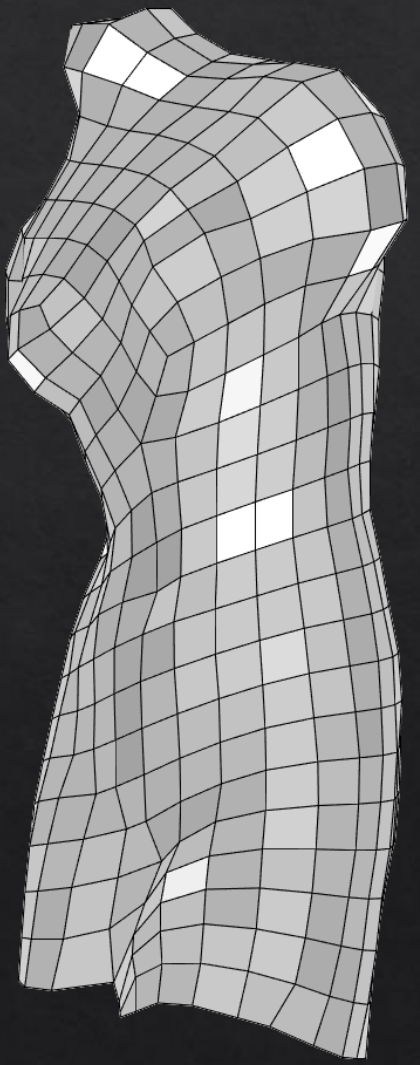
f_i: $v_{i,1}, v_{i,2}, v_{i,3}$

e_j: $v_{j,1}, v_{j,2}$



triangle mesh

Graph



quad mesh

Homework 1

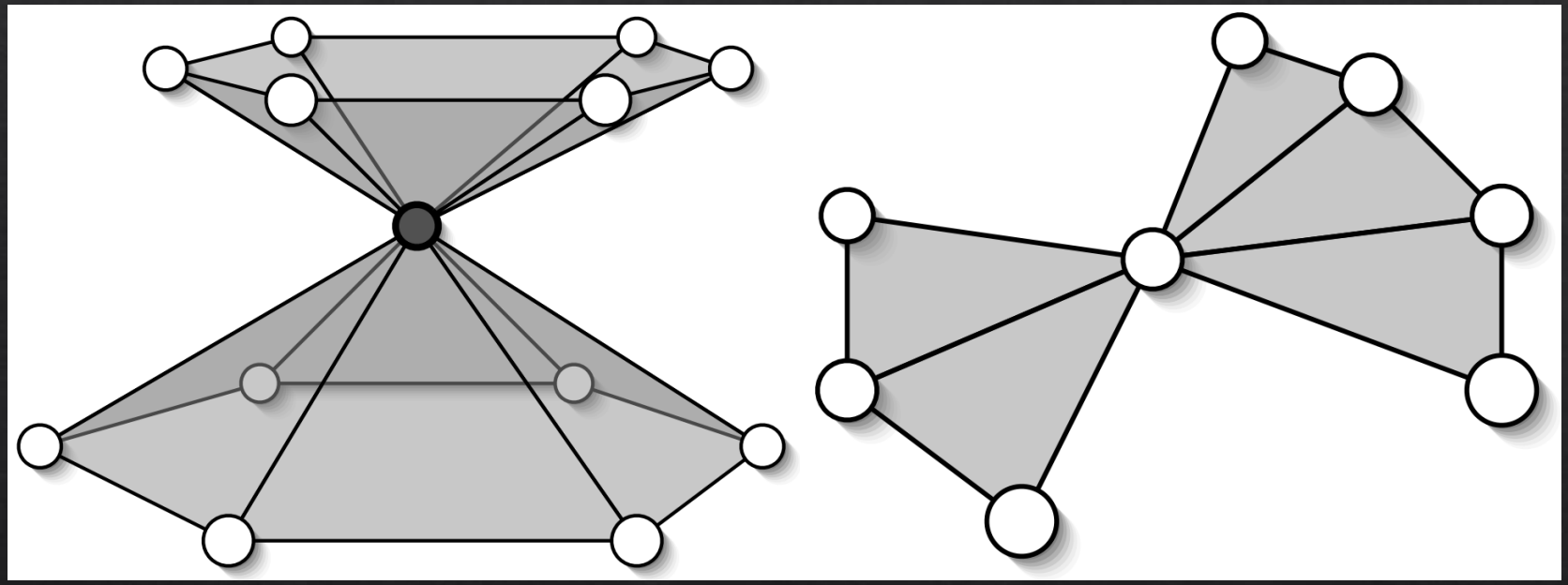
- Shortest path
  - Input: two vertices
  - Output: a edge path connecting the input two vertices with shortest length
  - How about the geodesic path?

- Minimal spanning tree
  - Input: some (>2) vertices
  - Output: a tree passing through all input vertices with minimum length

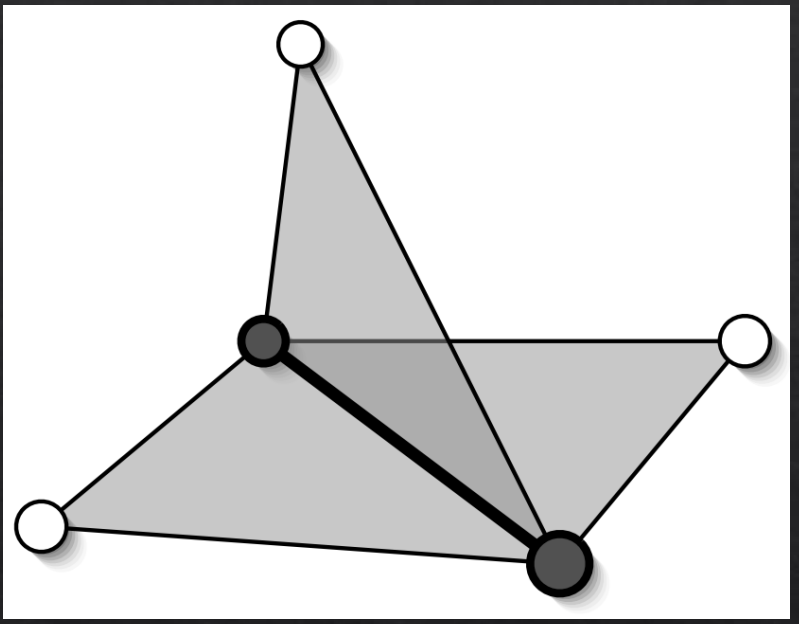
P: a set of vertices

1. For each pair of terminal points $\mathbf{v}_a \in \mathcal{P}$ and $\mathbf{v}_b \in \mathcal{P}$, compute the shortest path $\mathcal{C}_{a,b}$ between $\mathbf{v}_a$ and $\mathbf{v}_b$ on $\mathcal{G}$.
2. Construct a complete graph with the node set $\mathcal{P}$. The weight of each edge $\overline{\mathbf{v}_a\mathbf{v}_b}$ is equal to the length of $\mathcal{C}_{a,b}$.
3. Construct an MST for this shortest distance graph.
4. All mesh edges in the shortest paths that correspond to edges in the MST form an approximate Steiner tree for $\mathcal{G}$.

2-manifold



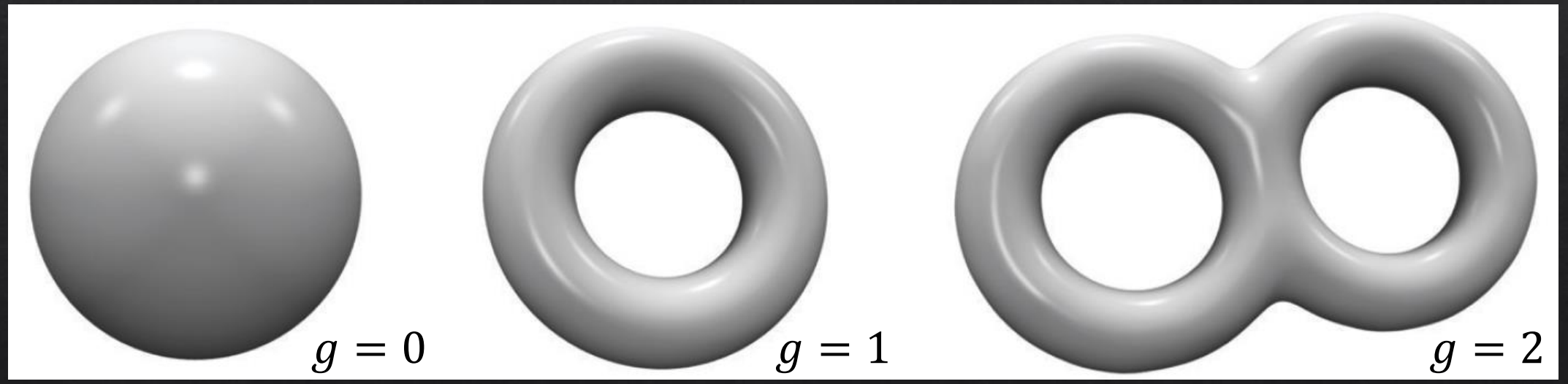
Non-manifold vertex is generated by pinching two surface sheets together at that vertex such that the vertex is incident to more than one fan of triangles.



Non-manifold edge has more than two incident triangles.

Euler formula

- $N_V - N_E + N_F = 2(1 - g)$
  - The numbers of vertices N_V, edges N_E, and faces N_F in a closed and mesh.



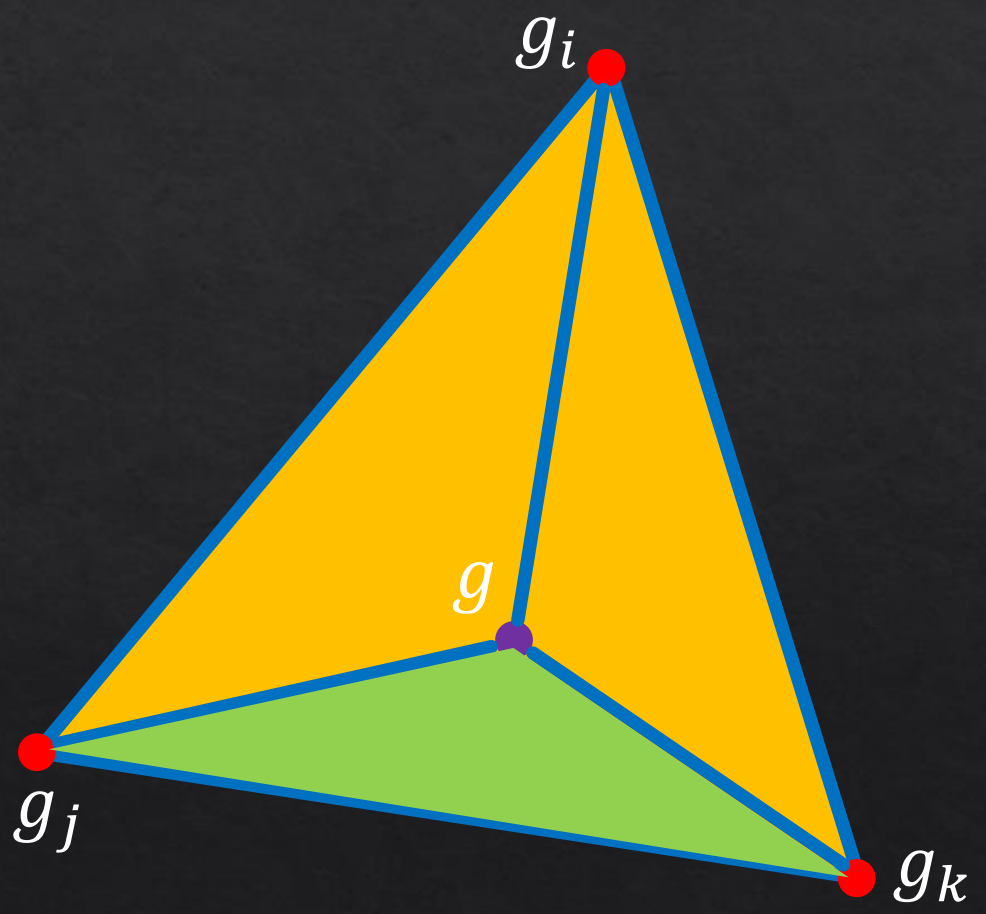
- $N_F \approx 2N_V, N_E \approx 3N_V$
  - 1 face, 1.5 edge → $N_E = 1.5N_F$

Barycentric coordinate

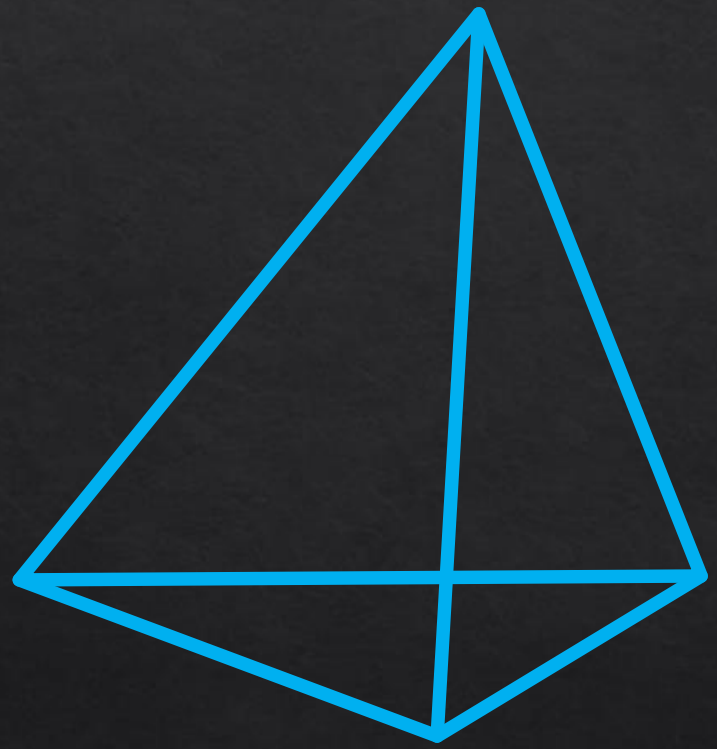
$$g = \alpha g_i + \beta g_j + \gamma g_k$$
$$\alpha + \beta + \gamma = 1,$$
$$\alpha, \beta, \gamma \geq 0\ .$$

$$\alpha = \frac{s_i}{s_i + s_j + s_k}$$

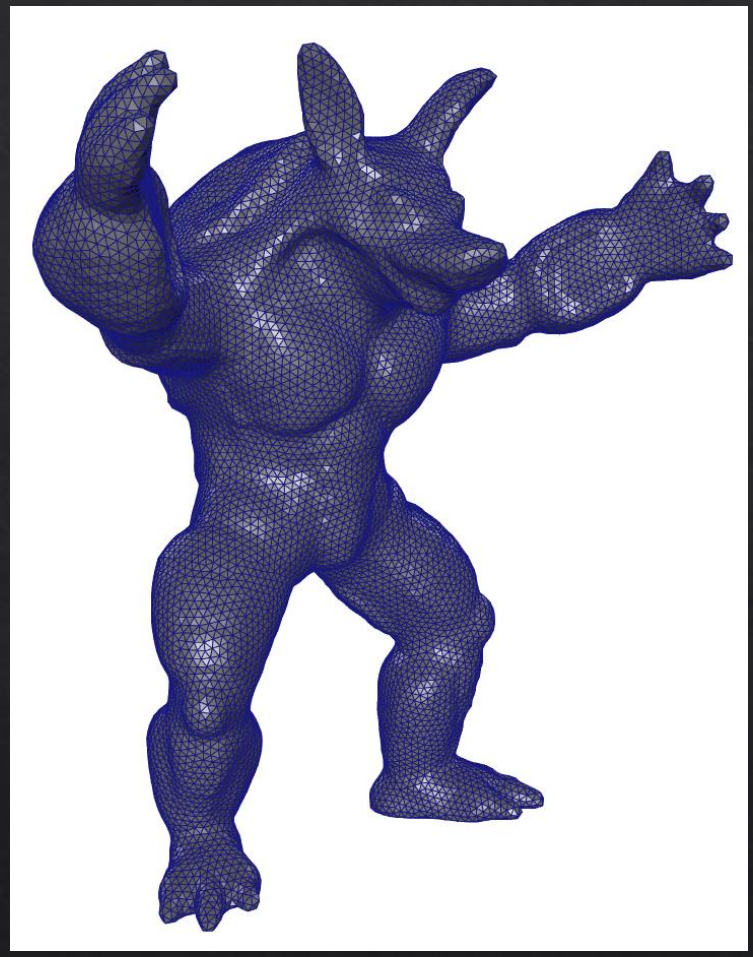
s_i: area of the green triangle

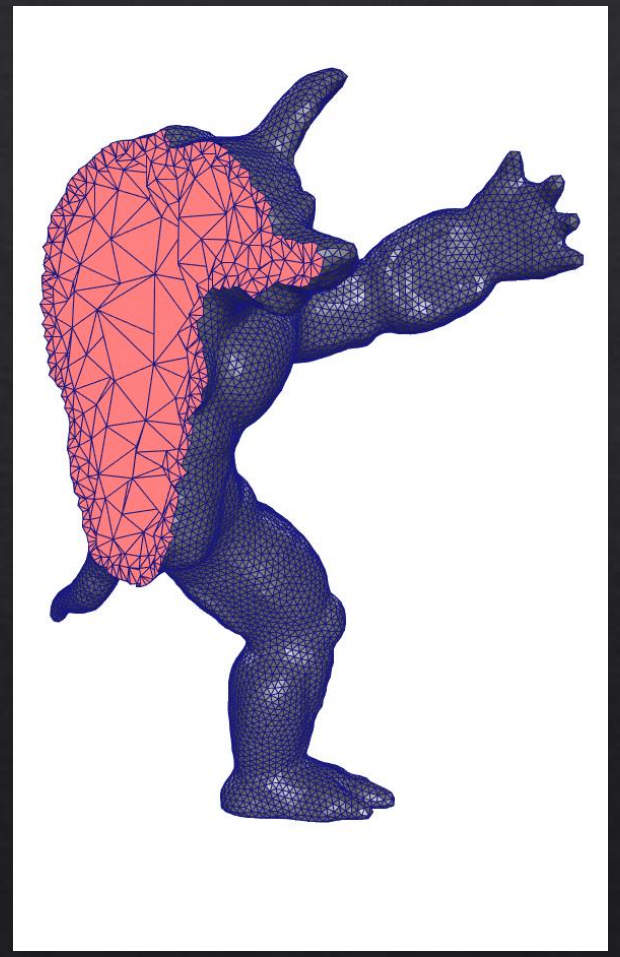


Tetrahedral Mesh



One tetrahedron

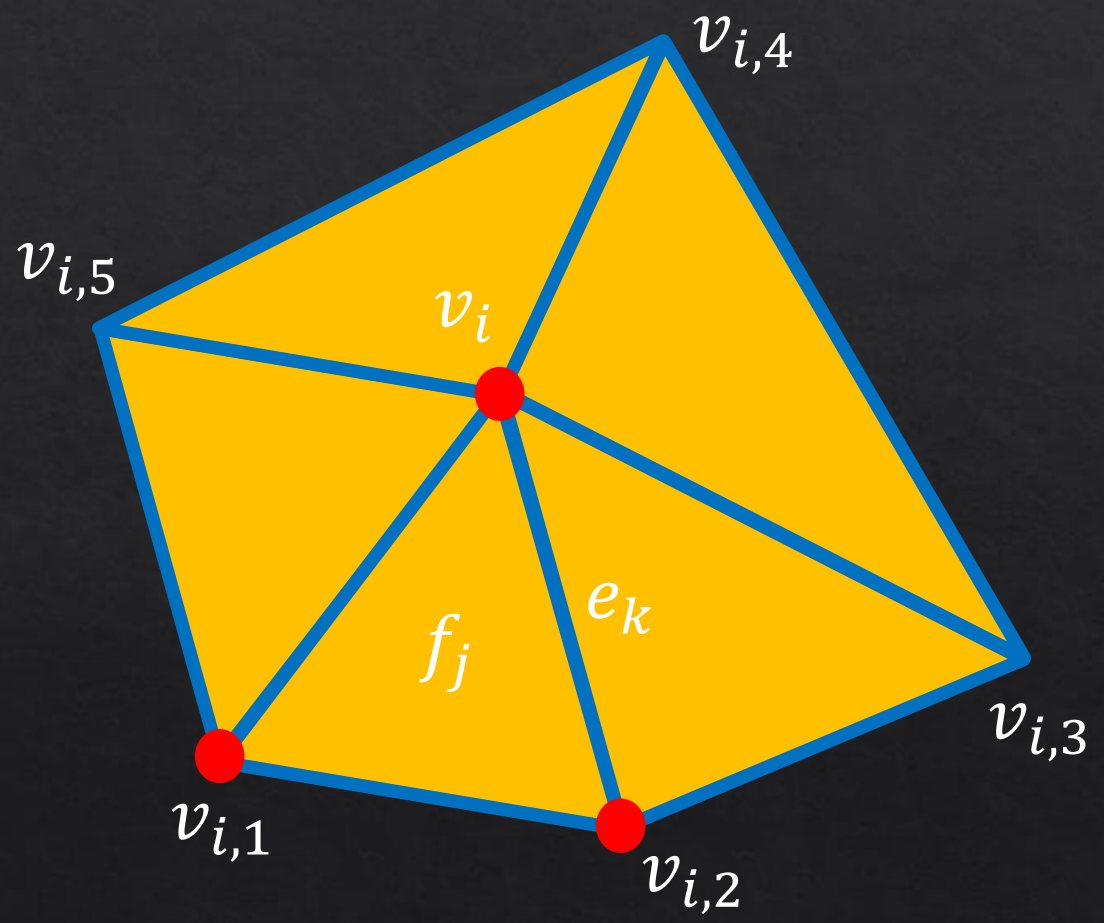




A collection of tetrahedrons

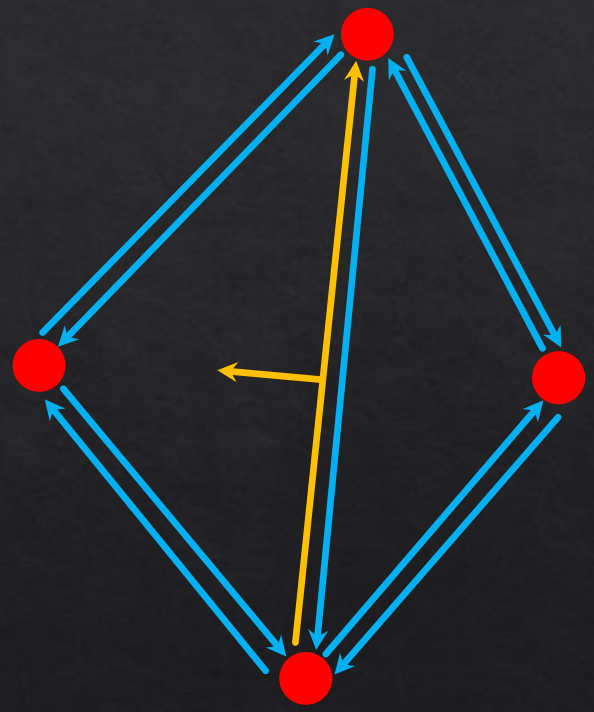
Data structure – requirements

- Given f_j, find its containing vertices in order.
- Given v_i, find its one-ring facets in order.
- Given v_i, find its outgoing edges.
- Given v_i, find its adjacent vertices.
- Given e_k, find its connected two facets.
- Given f_j and e_k, find another facet which connects e_k.
- … …



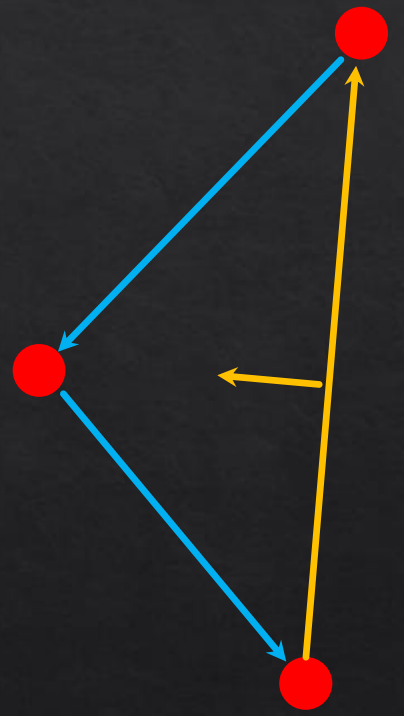
Halfedge

- Split each edge into two oriented halfedges.
- Halfedges are oriented consistently in counterclockwise order around each face and along each boundary.
- One halfedge corresponds one face.
- Boundary edge: empty face.

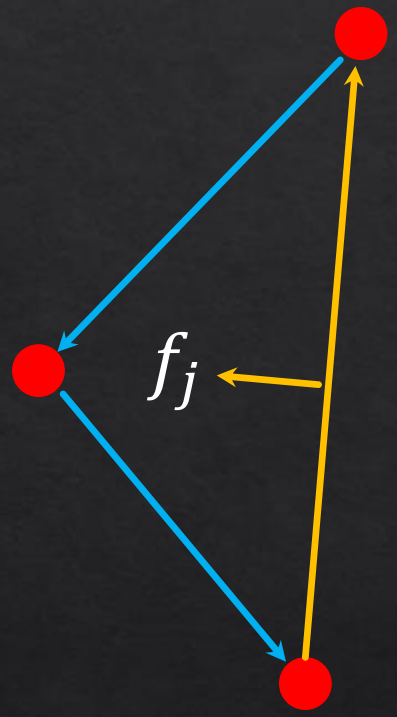


Halfedge

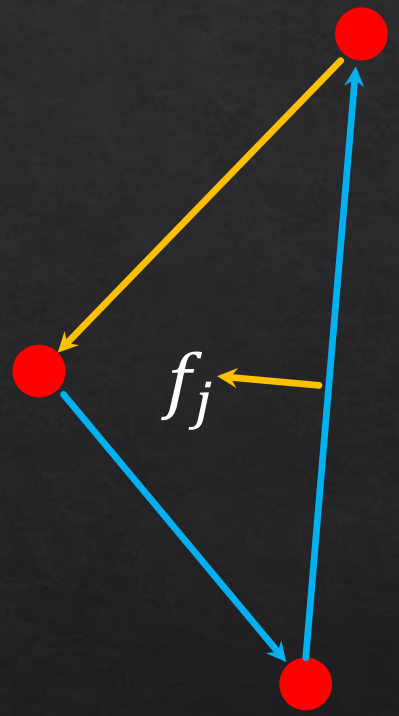
- Each halfedge:
  - the vertex it points to,
  - its adjacent face,
  - the next halfedge of the face or boundary,
  - the previous halfedge in the face,
  - its opposite (or inverse) halfedge.
- Each vertex:
  - Position
  - One outgoing halfedge
- Each face:
  - One referenced halfedge



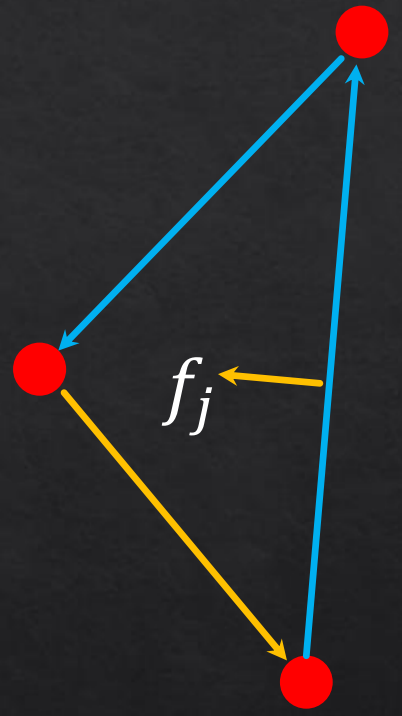
Vertices of one facet



One halfedge of f_j

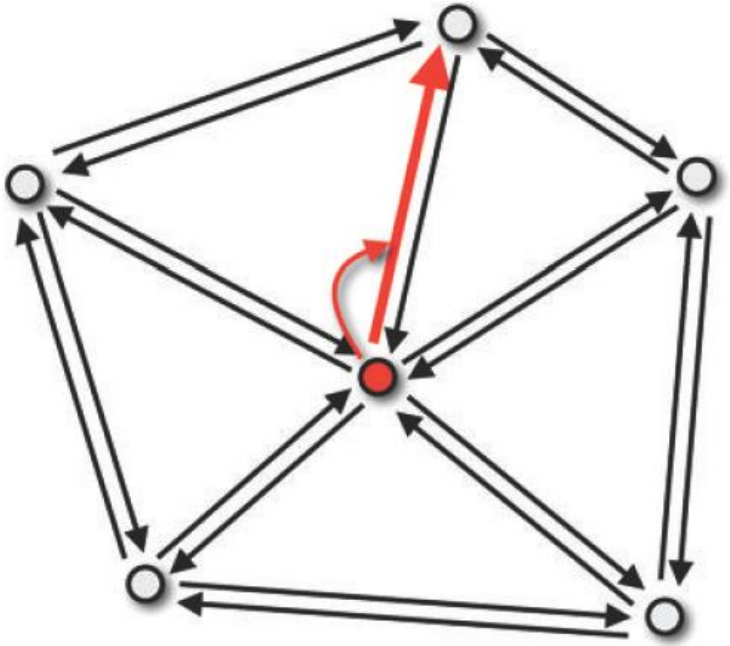


Next halfedge

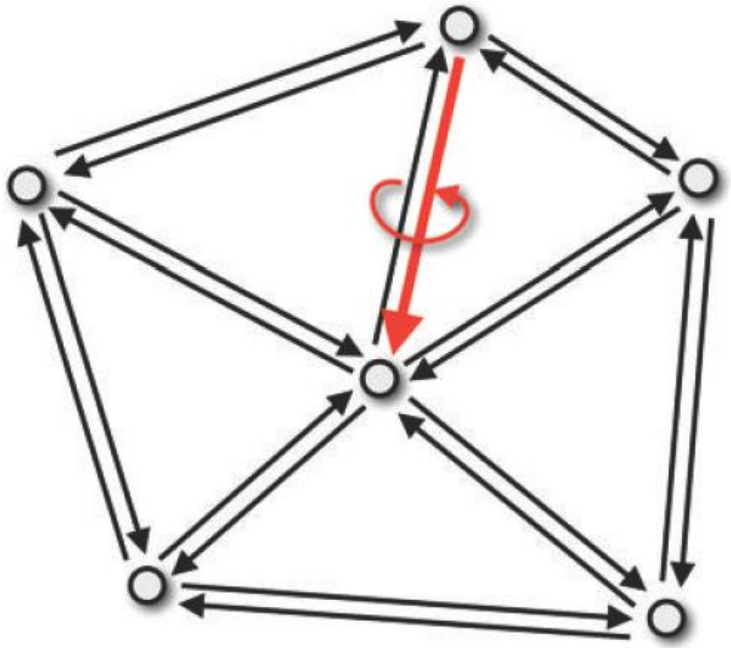


Next halfedge

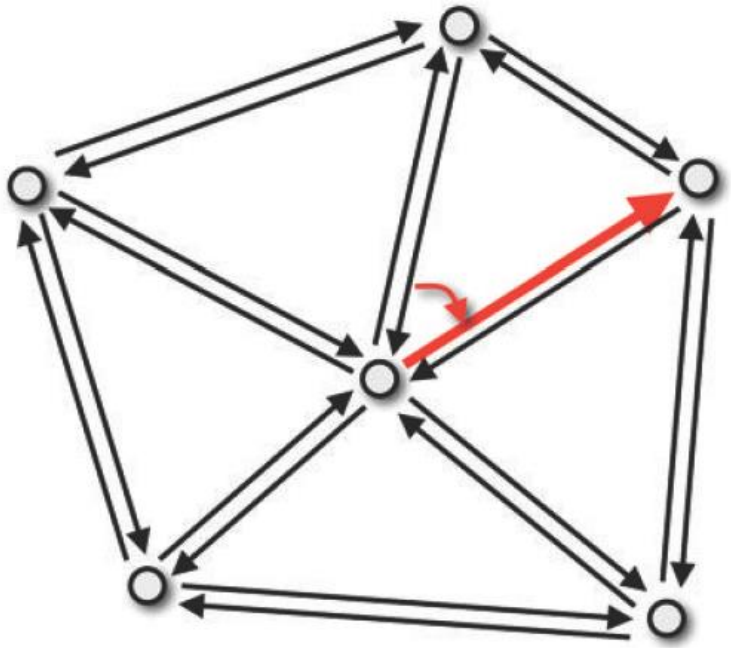
One-ring neighbors



One outgoing halfedge



Opposite halfedge



Next halfedge

Implementation thinking

- Code: halfedge data implementation
  - structure
  - std::vector

Outline

- Point cloud
- Signed distance field
- Implicit function
- Mesh
  - Triangle, Tetrahedron
- Grid
  - Pixel, Voxel
  - Quad-tree, Octree
- Data structure
  - Halfedge
- File format
  - Obj, Off

File format – obj

https://en.wikipedia.org/wiki/Wavefront_.obj_file

```
# List of geometric vertices, with (x,y,z) coordinates
v 0.123 0.234 0.345
……
# List of texture coordinates, in (u, v) coordinates
vt 0.500 1
……
# List of vertex normals in (x,y,z) form
vn 0.707 0.000 0.707
……
 # Polygonal face element (see below)
f 6/4/1 3/5/3 7/6/5       (f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3) from 1
……
```

File format – off

http://people.sc.fsu.edu/~jburkardt/data/off/off.html

OFF	#Line 1
vertex_count face_count edge_count	#Line 2
x y z	#One line for each vertex
……	
n v1 v2 … Vn	One line for each polygonal face, from 0
……	

Off - example

```
OFF
8 6 0
-0.500000 -0.500000 0.500000
0.500000 -0.500000 0.500000
-0.500000 0.500000 0.500000
0.500000 0.500000 0.500000
-0.500000 0.500000 -0.500000
0.500000 0.500000 -0.500000
-0.500000 -0.500000 -0.500000
0.500000 -0.500000 -0.500000
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```