

# 3D data and representations

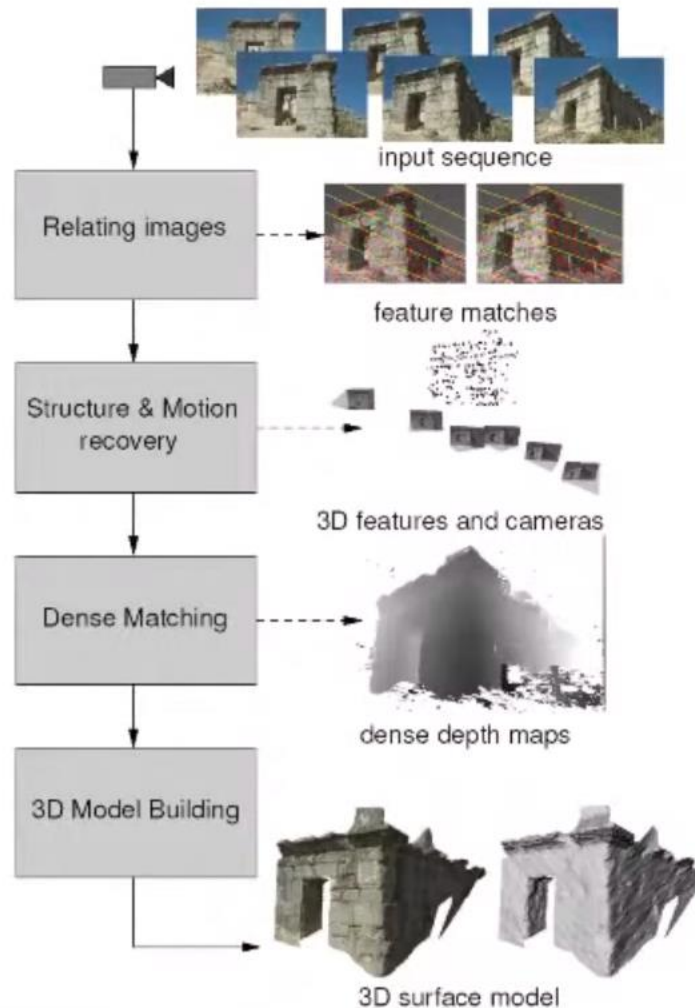
RRC SUMMER SCHOOL 2021

# 3D Data

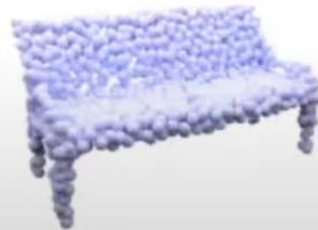
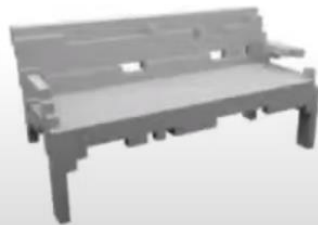
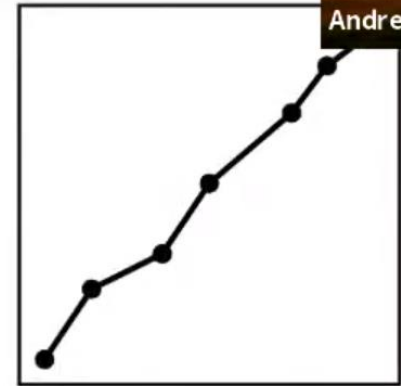
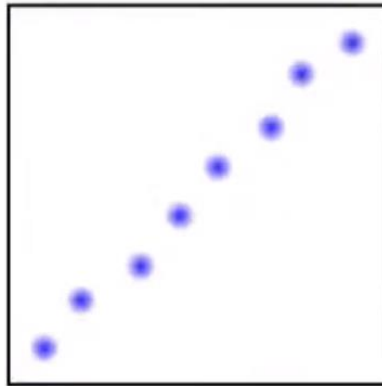
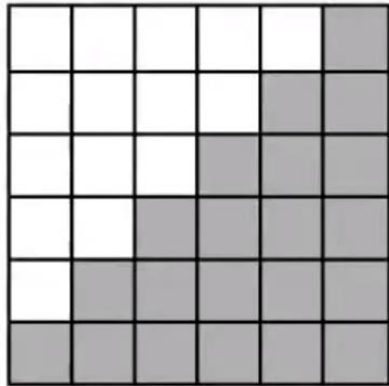
first full pipeline for detailed  
3D reconstruction in the wild

*Pollefeys et al. ICCV'98*  
*Pollefeys et al. 3DIM'99*

photo/video to 3D  
processing pipeline



# 3D Representations

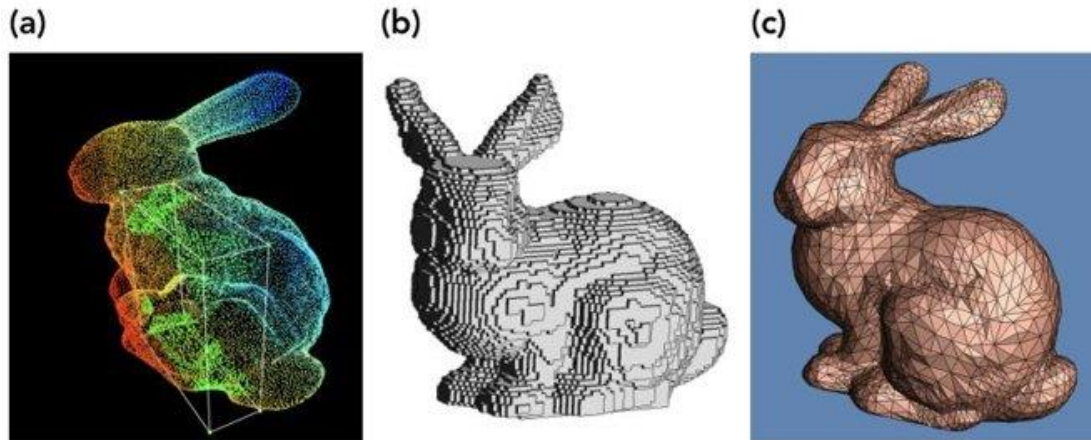


Voxels

Points

Meshes

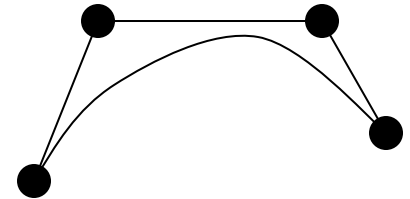
# 3D Representations



- Point cloud is a collection of points in three-dimensional space; each point is determined by a certain  $(x,y,z)$  position, and we can also specify other attributes (such as RGB color) for it. They are the original form of the lidar data when it is acquired.
- The voxel grid is developed from the point cloud. Voxel is like a pixel in a three-dimensional space. We can regard the voxel grid as a quantized point cloud with a fixed size.
- The polygon mesh consists of a set of convex polygonal surfaces with common vertices, which can approximate a geometric surface. We can think of a point cloud as a three-dimensional point set sampled from a basic continuous set surface.

# 3D Object Representation

- A surface can be analytically generated using its function involving the coordinates.
- An object can be represented in terms of its vertices, edges and polygons. (Wire Frame, Polygonal Mesh etc.)
- Curves and surfaces can also be designed using splines by specifying a set of few control points.



# 3D Object Representation

Representation schemes, two major classes :

## Boundary representation(B-reps)

- A **set of surfaces** that separate the object interior from the environment
- Examples: Polygon facets, spline patches

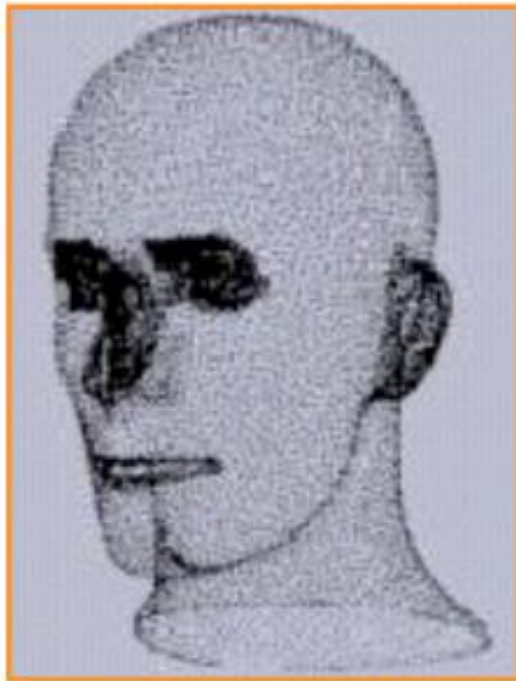
## Space-partitioning representation

- Used to describe interior properties.
- Partitioning the spatial region into a **set of small, non-overlapping, contiguous solids (usually cubes)**
- Examples: octree representation

# Point cloud

$(X,Y,Z)$ ,  $(X,Y,Z,r,g,b)$   
 $(X,Y,Z, r,g,b, \text{normal})$

- Unstructured set of 3D point samples
  - Acquired from range finder, computer vision, etc



Hoppe



Hoppe

# Mesh

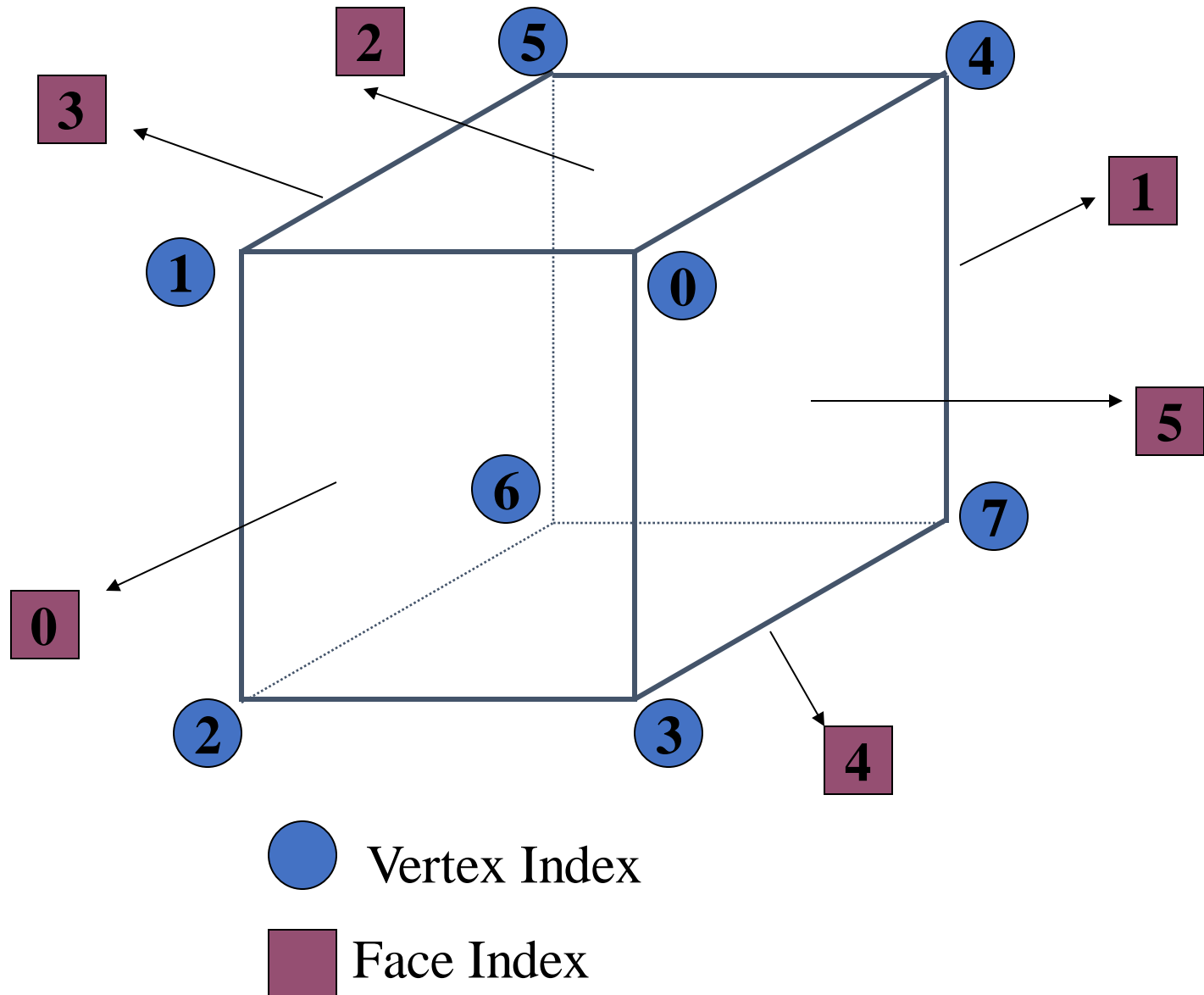
.obj, .fbx, .ply, .glTF, .3ds

The data for polygonal meshes can be represented in two ways.

- Method 1:
  - Vertex List
  - Normal List
  - Face List (Polygon List)
- Method 2:
  - Vertex List
  - Edge List
  - Face List (Polygon List)



# Vertices and Faces - Eg. Cube



# Data representation using vertex, face and normal lists:

## Vertex List

30	30	30
-30	30	30
-30	-30	30
30	-30	30
30	30	-30
-30	30	-30
-30	-30	-30
30	-30	-30

## Normal List

0.0	0.0	1.0
0.0	0.0	-1.0
0.0	1.0	0.0
-1.0	0.0	0.0
0.0	-1.0	0.0
1.0	0.0	0.0

## Polygon List

0	1	2	3
4	7	6	5
0	4	5	1
1	5	6	2
2	6	7	3
3	7	4	0

# Data representation using vertex, face and edge lists:

<u>Vertex List</u>			<u>Edge List</u>		<u>Polygon List</u>			
x[i]	y[i]	z[i]	L[j]	M[j]	P[k]	Q[k]	R[k]	S[k]
30	30	30	0	1	0	1	2	3
-30	30	30	1	2	4	7	6	5
-30	-30	30	2	3	0	4	5	1
30	-30	30	3	0	1	5	6	2
30	30	-30	4	5	2	6	7	3
-30	30	-30	5	6	3	7	4	0
-30	-30	-30	6	7				
30	-30	-30	7	4				
			0	4				
			1	5				
			2	6				
			3	7				

# Normal Maps

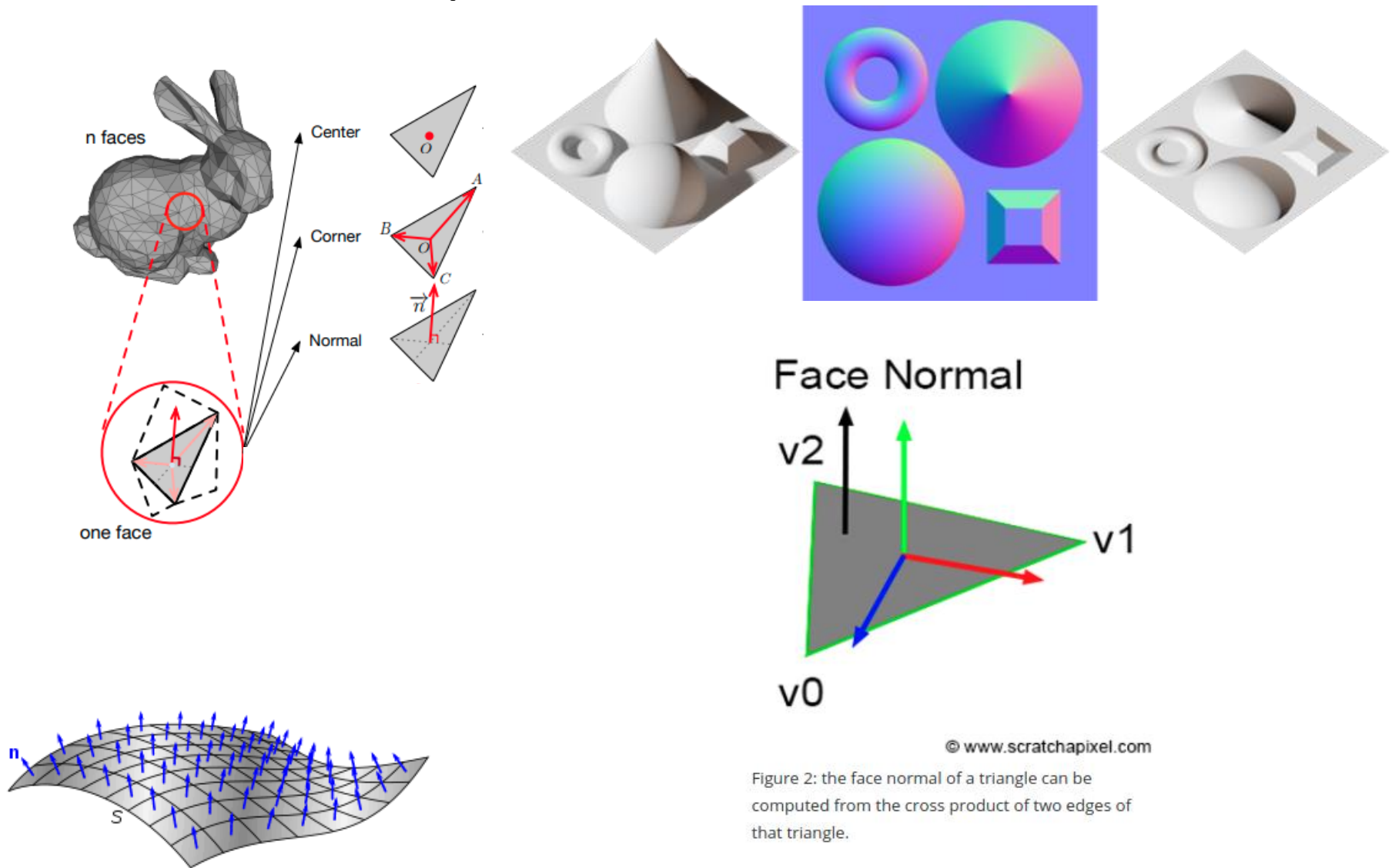
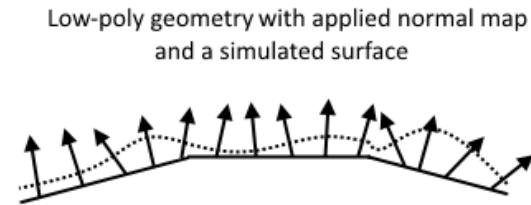
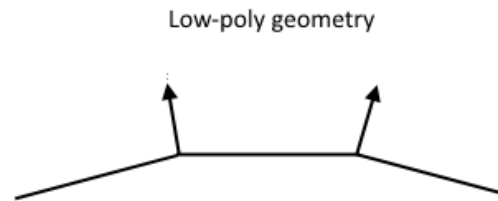
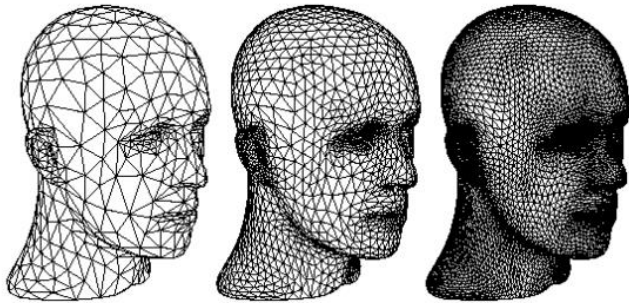


Figure 2: the face normal of a triangle can be computed from the cross product of two edges of that triangle.

# Meshes can only approximate!

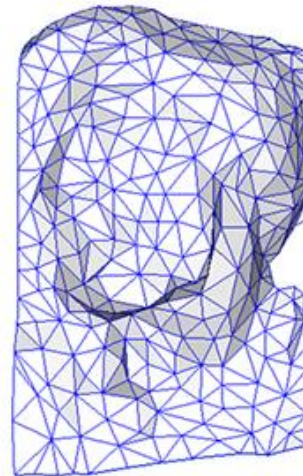
- Define surfaces as limit of refinement sequence
  - Guaranteed continuity, concise



Normal mapping interpolation helps here.



original mesh  
4M triangles

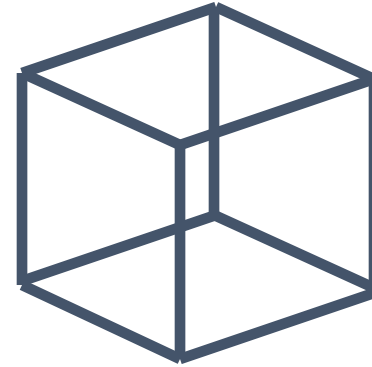


simplified mesh  
500 triangles



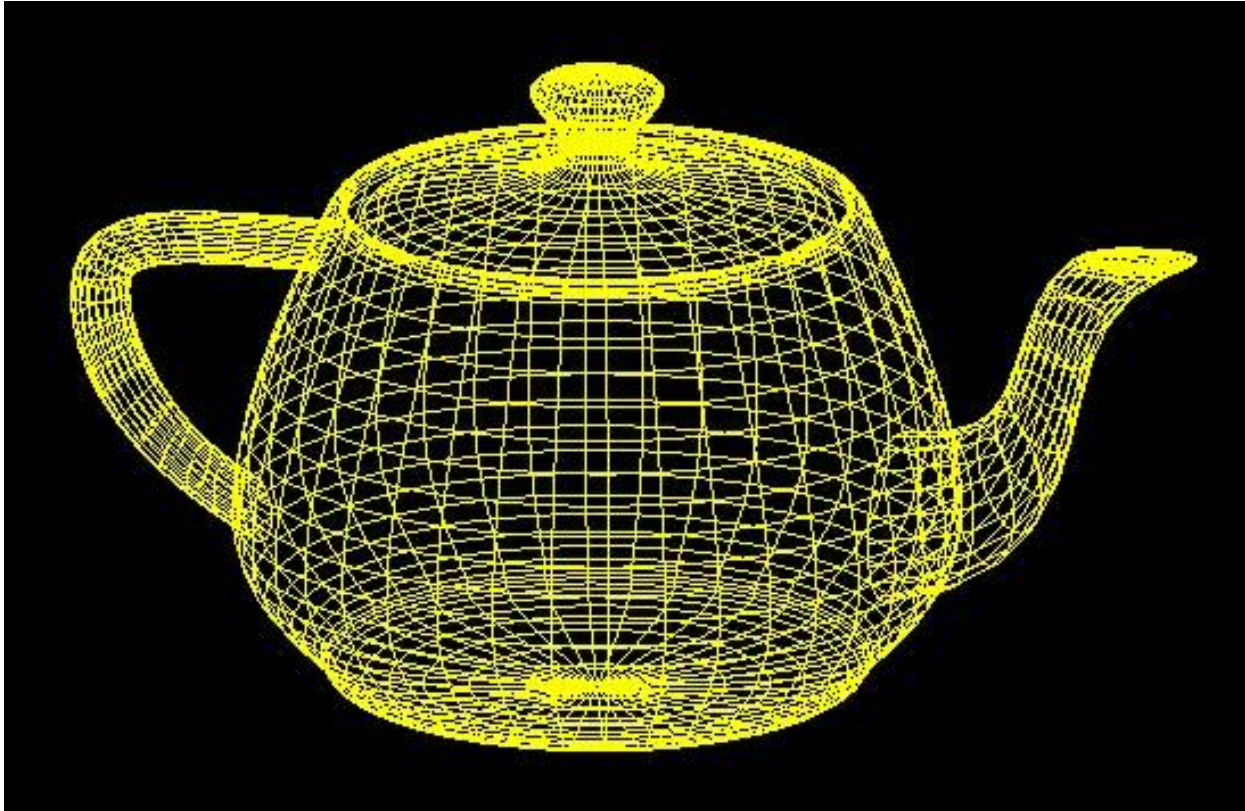
simplified mesh  
and normal mapping  
500 triangles

# Wire-Frame Models



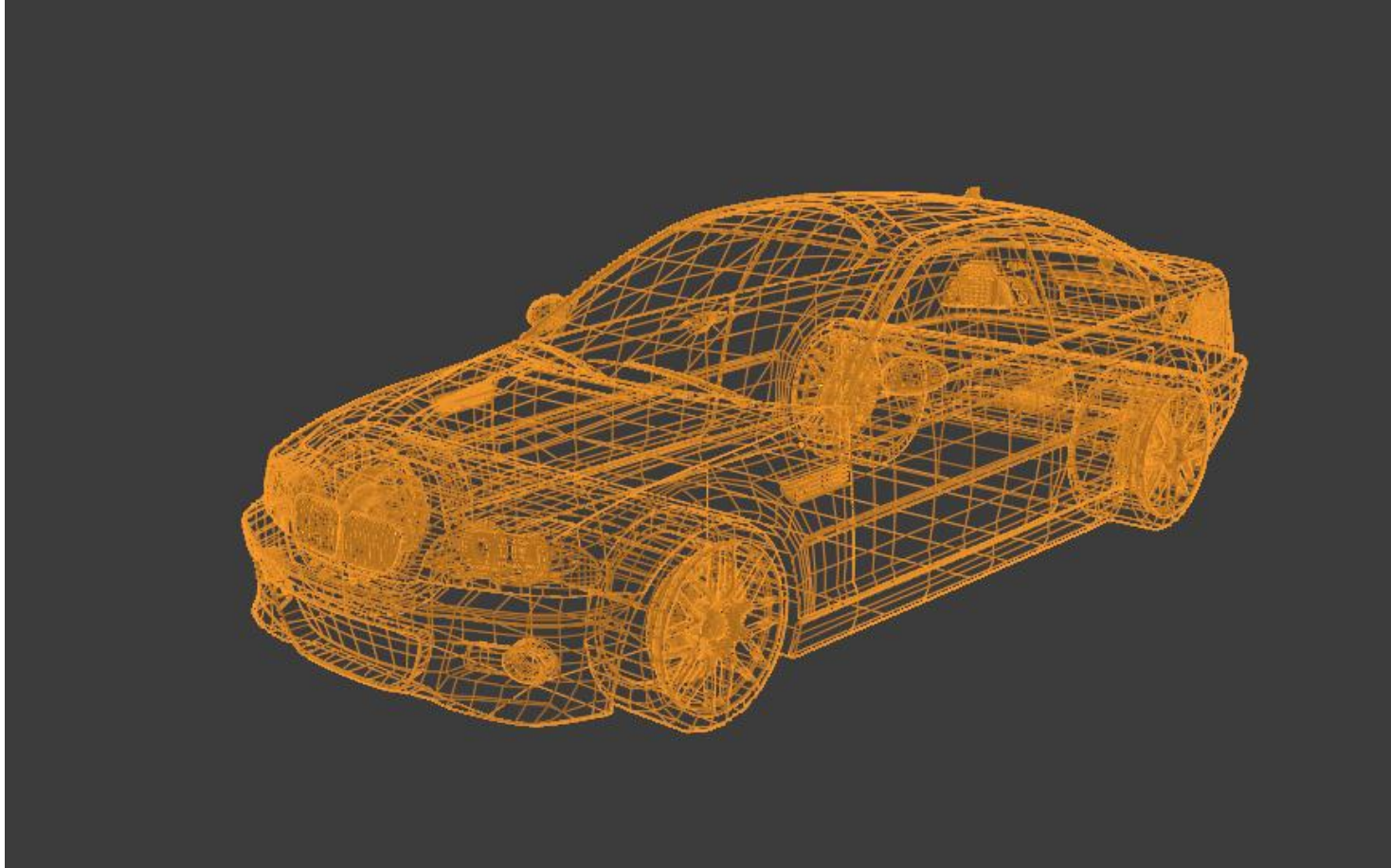
- If the object is defined only by a set of nodes (vertices), and a set of lines connecting the nodes, then the resulting object representation is called a wire-frame model.
  - Very suitable for engineering applications.
  - Simplest 3D Model - easy to construct.
  - Easy to clip and manipulate.
  - Not suitable for building realistic models.

# Wire Frame Model - The Teapot





# Wire Frame Model – BMW M3





# Polygon Meshes

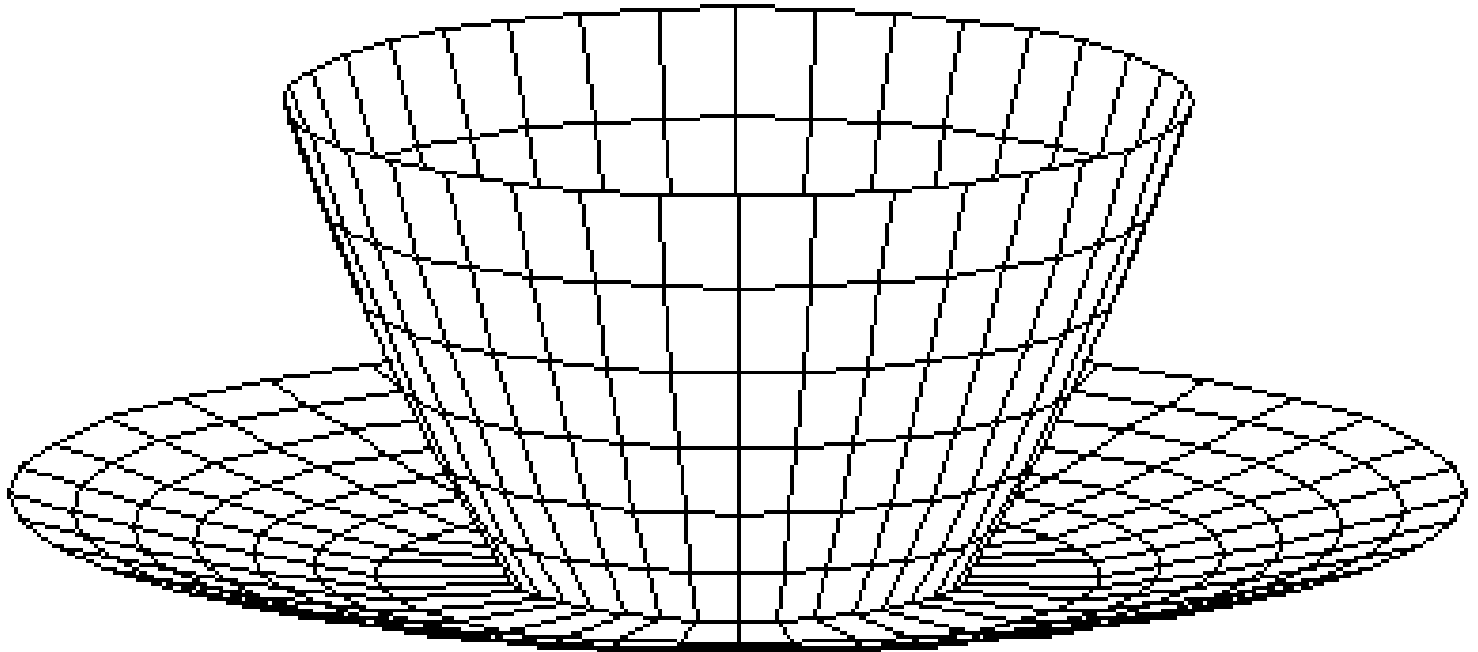
## Advantages

- It can be used to model almost any object.
- They are easy to represent as a collection of vertices.
- They are easy to transform.
- They are easy to draw on computer screen.

## Disadvantages

- Curved surfaces can only be approximately described.
- It is difficult to simulate some type of objects like hair or liquid.

# Polygonal Mesh - Example

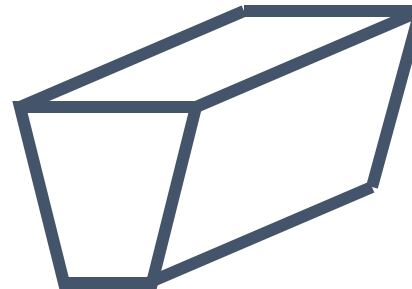


# Solid Modeling

- Polygonal meshes can be used in solid modeling.
- An object is considered **solid** if the polygons fit together to enclose a space.
- In solid models, it is necessary to incorporate directional information on each face by using the *normal vector* to the plane of the face, and it is used in the shading process.

# Solid Modeling - Polyhedron

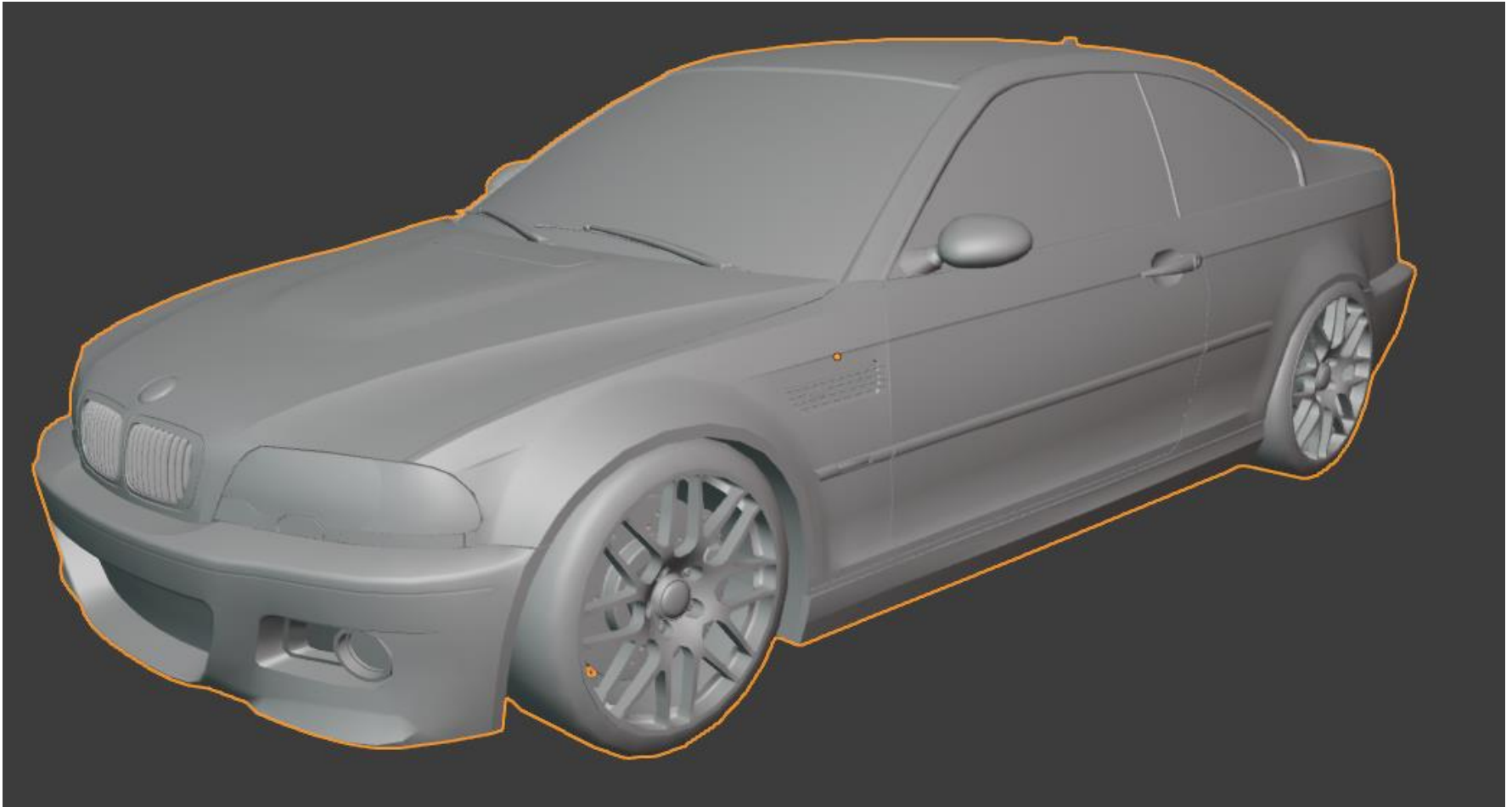
- A polyhedron is a connected mesh of simple planar polygons that encloses a finite amount of space.
- A polyhedron is a special case of a polygon mesh that satisfies the following properties:
  - Every edge is shared by exactly two faces.
  - At least three edges meet at each vertex.
  - Faces do not interpenetrate. Faces at most touch along a common edge.
- Euler's formula: If  $F$ ,  $E$ ,  $V$  represent the number of faces, vertices and edges of a polyhedron, then
$$V + F - E = 2.$$



# Solid Modeling



# Solid Modeling



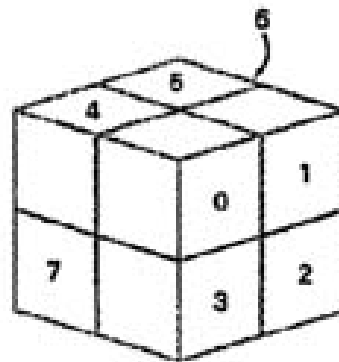
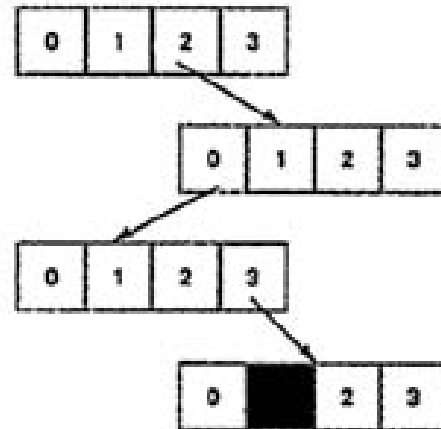
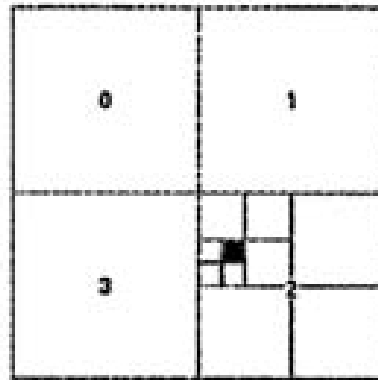
# Octrees

- An octree encoding scheme divide regions of 3-D space(usually a cube) in to octants and stores 8 data elements in each node of the tree.
- Individual elements of a 3-D space are called *volume elements* or *voxels*.
- When all voxels in an octant are of the same type, this type value is stored in the corresponding data element of the node. Any heterogeneous octant is subdivided into octants and the corresponding data element in the node points to the next node in the octree.

<https://www.youtube.com/watch?v=Qb93QSvxXrs>

showing how octree subdivides a 3d mesh

# Octrees



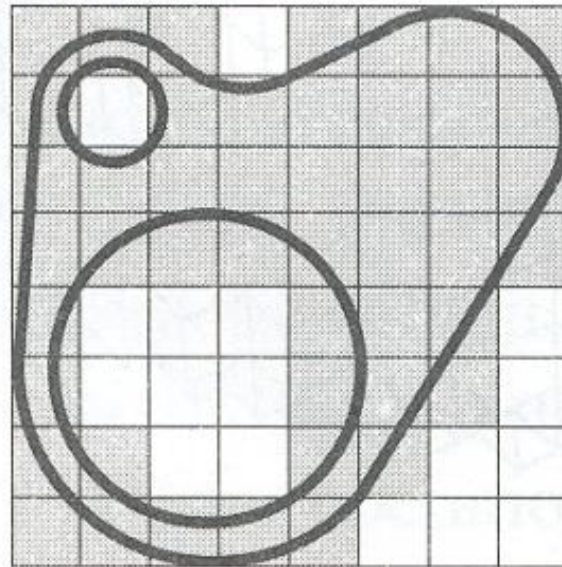
Region of a  
Three-Dimensional  
Space



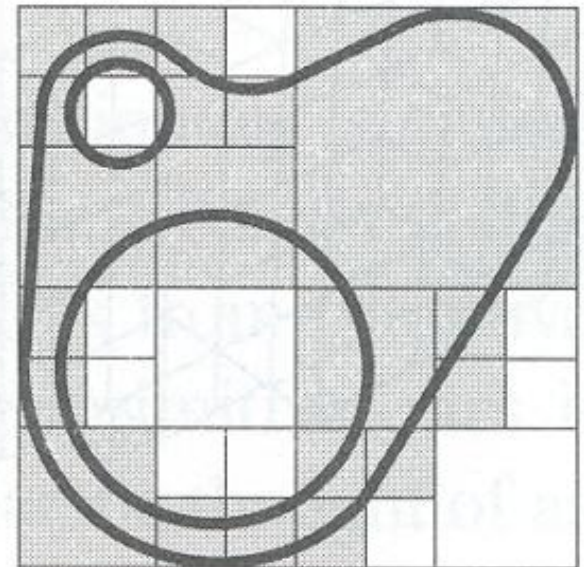
Data Elements  
in the Representative  
Octree Node



- Refine resolution of voxels hierarchically
  - More concise and efficient for non-uniform objects



Uniform Voxels

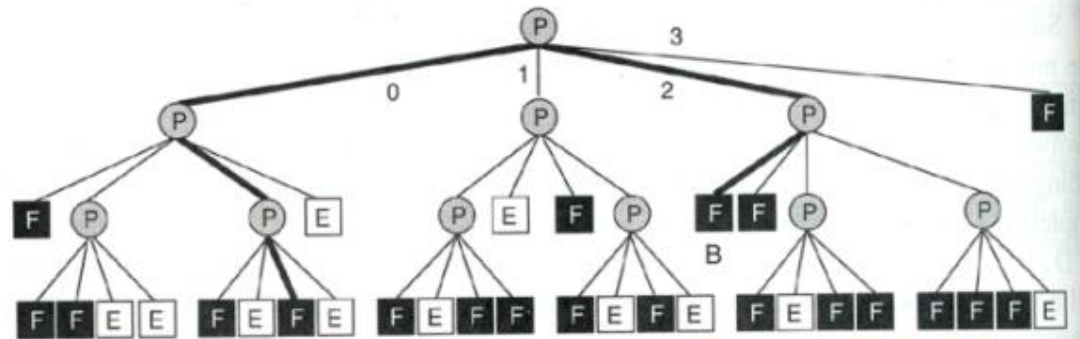
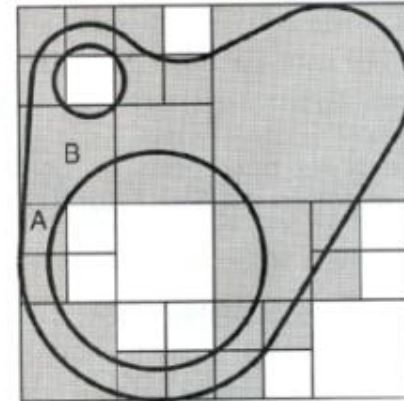


Quadtree (Octree in 3D)

Octree analogous to Quadtree(2D)

- Hierarchical versions of voxel methods

- Finding neighbor cell requires traversal of hierarchy:  
expected/amortized  $O(1)$



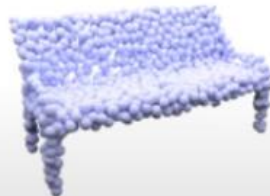
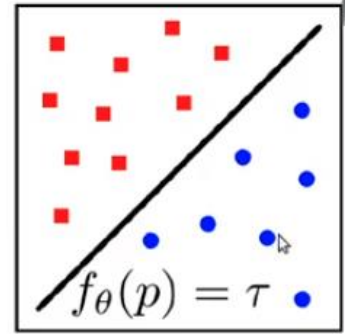
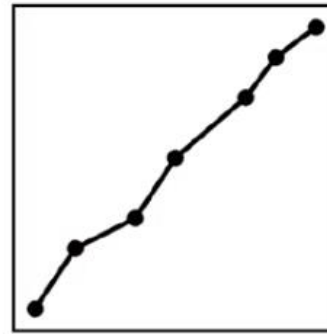
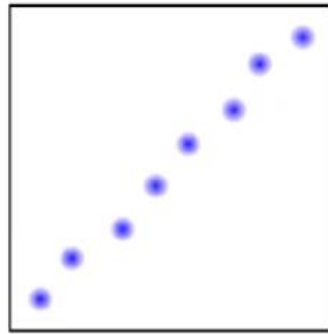
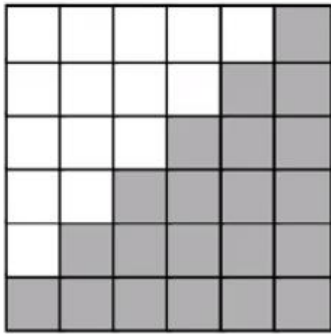
FvDFH Figure 12.25

# Voxels

---

- Advantages
  - Simple, intuitive, unambiguous
  - Same complexity for all objects
  - Natural acquisition for some applications
  - Trivial boolean operations
- Disadvantages
  - Approximate
  - Not affine invariant
  - Expensive display
  - Large storage requirements

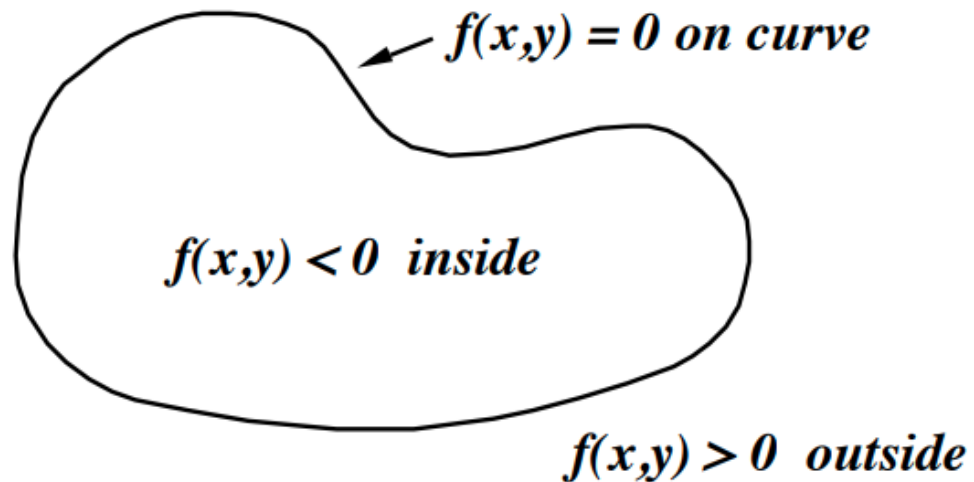
# Implicit Representations



- ▶ Traditional Explicit Representations  $\Rightarrow$  **Discrete**
- ▶ Implicit Neural Representation  $\Rightarrow$  **Continuous**

# Implicit Representations

- Surface defined implicitly by function:
  - $f(x, y, z) = 0$  (on surface)
  - $f(x, y, z) < 0$  (inside)
  - $f(x, y, z) > 0$  (outside)



# Occupancy Networks

<https://autonomousvision.github.io/occupancy-networks/>

## Key Idea:

- ▶ Do not represent 3D shape explicitly
- ▶ Instead, consider surface **implicitly** as **decision boundary** of a non-linear classifier:

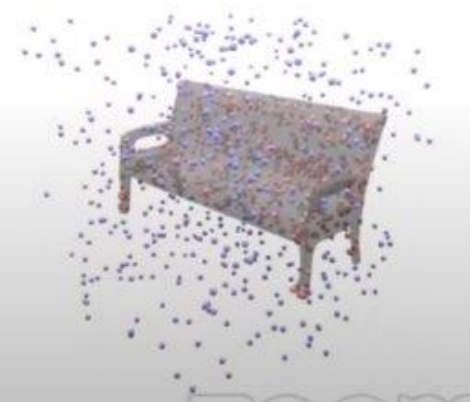
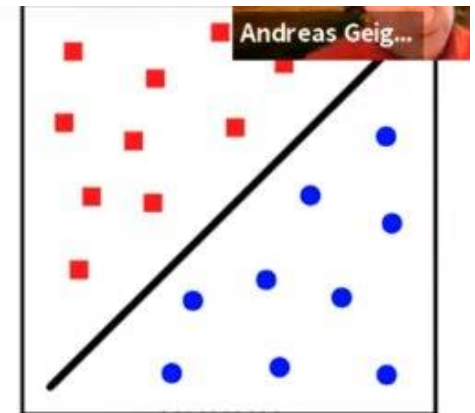
$$f_{\theta} : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1]$$

$\uparrow$                        $\uparrow$                        $\nwarrow$   
3D Location           Condition (eg, Image)           Occupancy Probability


## Concurrent work:

- ▶ DeepSDF [Park et al., CVPR 2019]

▶ IM-NET [Chen et al., CVPR 2019]



Look at SDF, TSDF representations as well



# Open3D, Blender, Examples

<http://www.open3d.org/docs/release/introduction.html>

<https://www.blender.org/>

(sim2real, synthetic datasets)

<https://cdinstitute.github.io/Building-Dataset-Generator/>

<https://anuragsahu.github.io/WareSynth/>

## RESOURCES

- How to represent 3D Data?

<https://towardsdatascience.com/how-to-represent-3d-data-66a0f6376afb>

- 3D ML compilation

<https://github.com/timzhang642/3D-Machine-Learning>

## DATASETS

- [http://www.cvlibs.net/datasets/kitti/eval\\_3dobject.php](http://www.cvlibs.net/datasets/kitti/eval_3dobject.php)
- <https://shapenet.org/>
- <https://structured3d-dataset.org/>