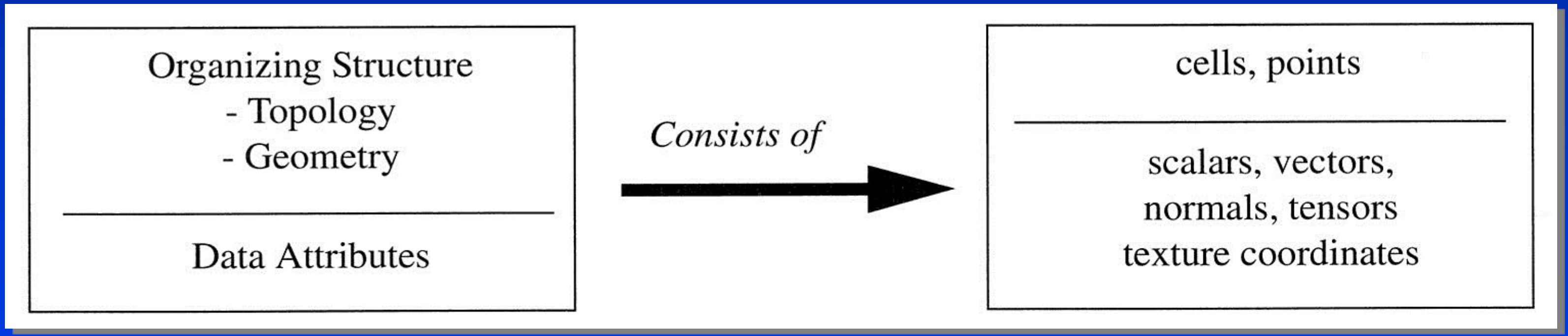
Basic Data Representations for Visualization

Data Representations

- There are many ways to represent datasets
- Points (e.g., 3D raster, point cloud)
- Lines
- Vectors
- These are all **discrete** data representations
- Data can be **regular** or **irregular**
- Regular = relationship exists between data points
- Compare: 3D raster vs. point cloud
- Data also has **dimension**: 1, 2, 3, …., n, …

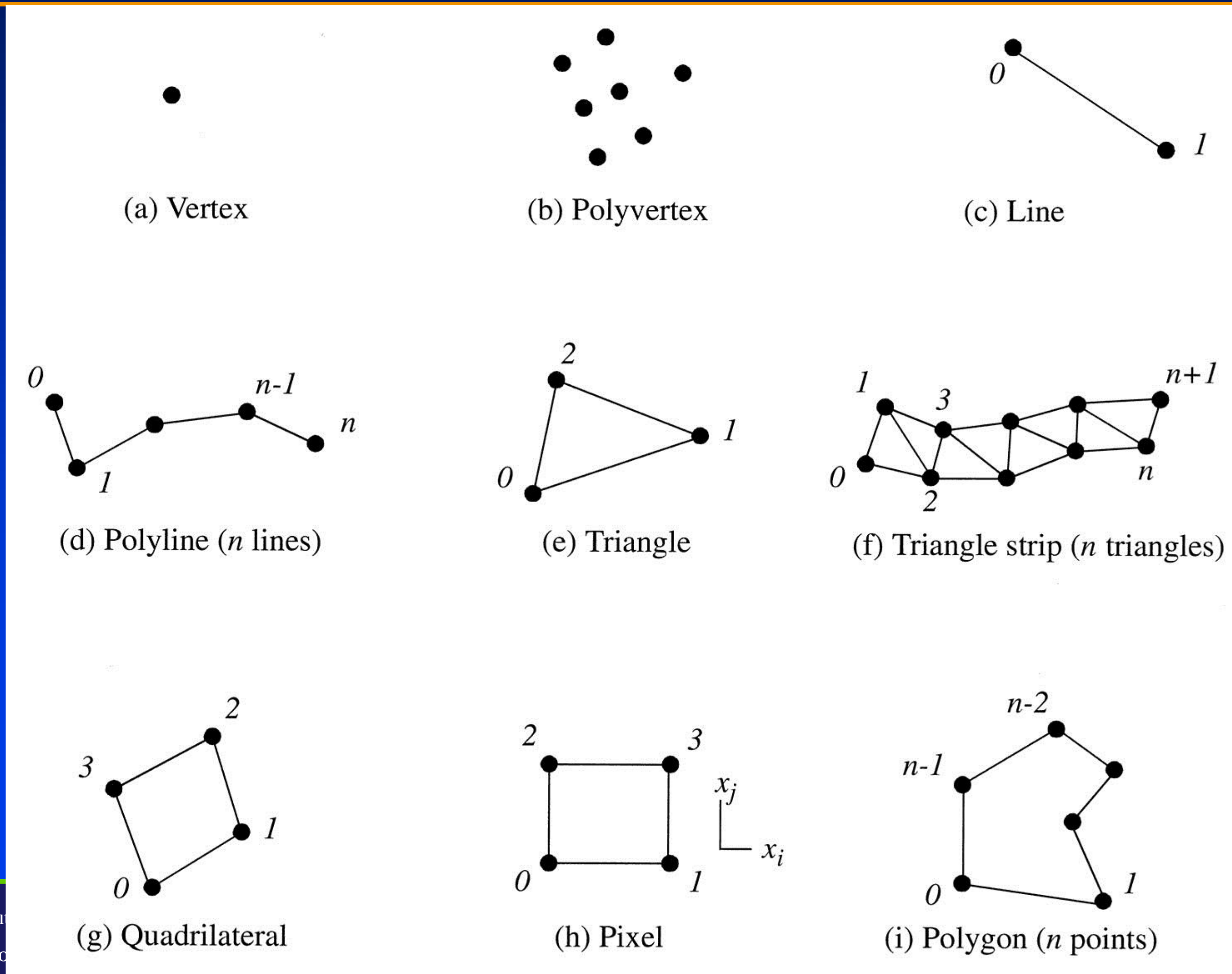
Dataset = Structure + Attributes

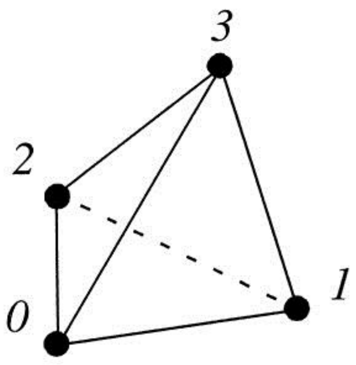
- Structure = topology and geometry
- Topology refers to characteristics unchanged by transformations (holes, handles, branches)
- Geometry refers to (x,y,z) positions of data points



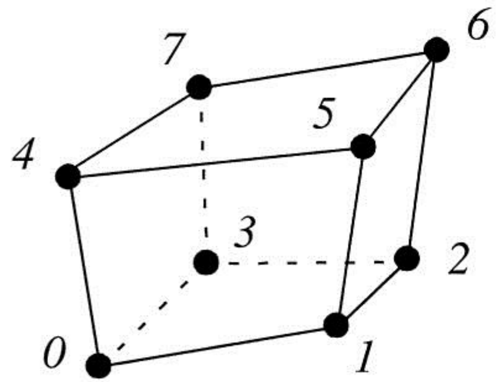
- Here **cells** define topology, **points** define geometry
- There could be a large variety of different **cell types**
- Linear cell types and non-linear cell types

Cell Topology (Connectivity)

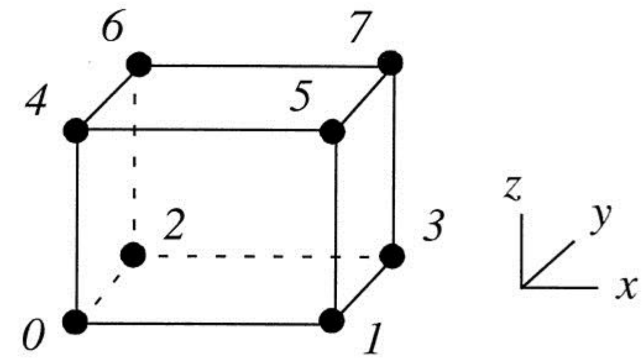




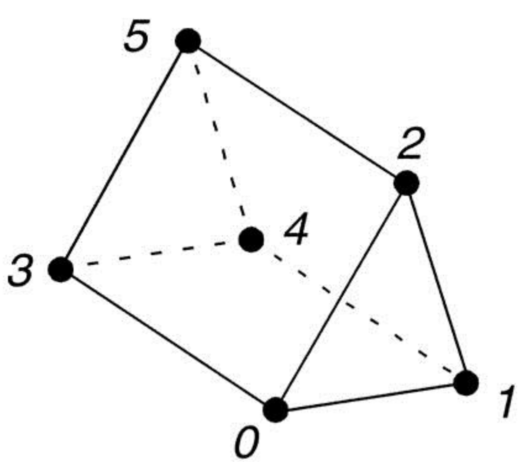
(j) Tetrahedron



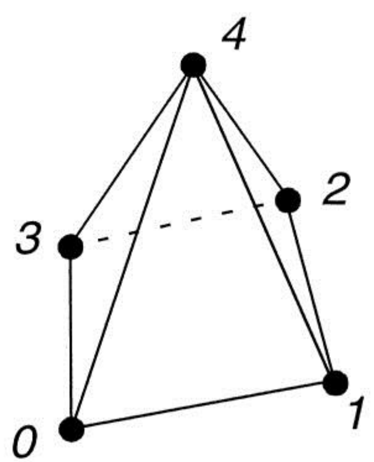
(k) Hexahedron



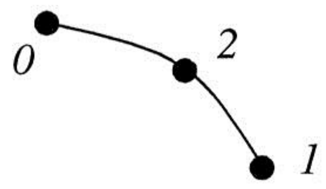
(l) Voxel



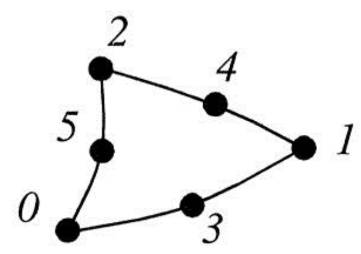
(m) Wedge



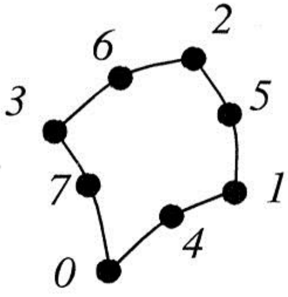
(n) Pyramid



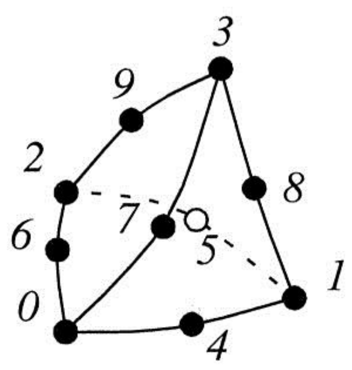
(a) Quadratic Edge



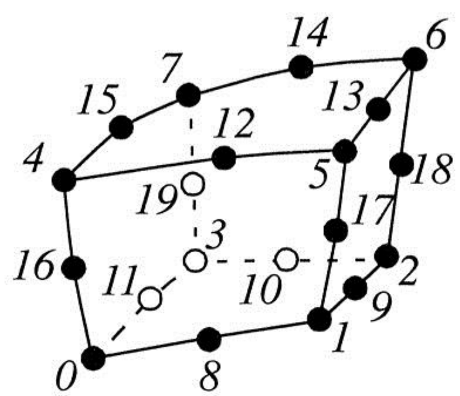
(b) Quadratic Triangle



(c) Quadratic Quadrilateral



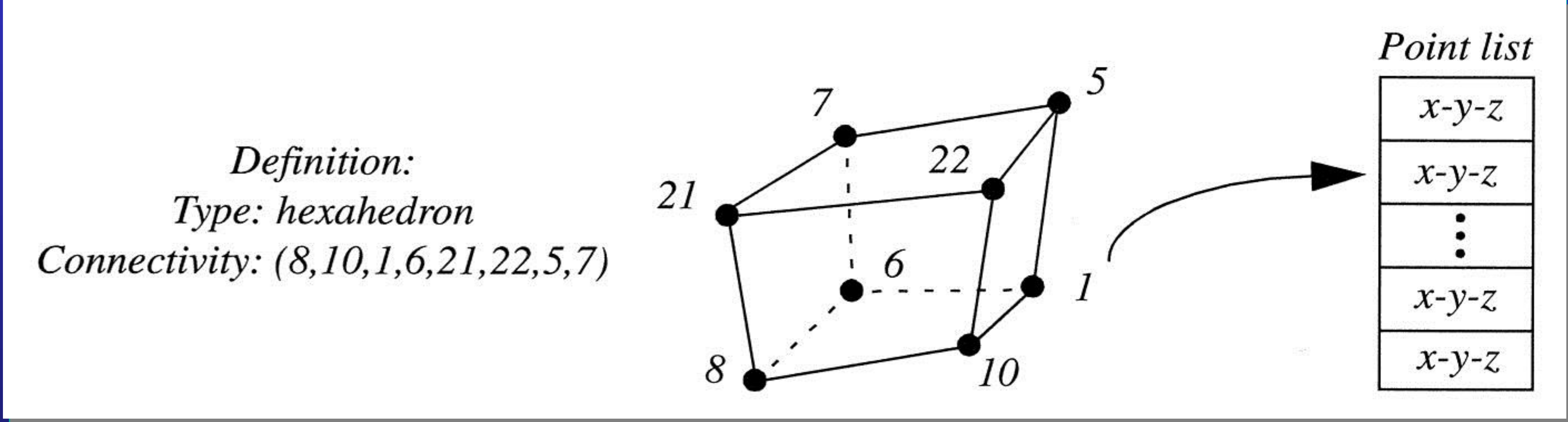
(d) Quadratic Tetrahedron



(e) Quadratic Hexahedron

Cell Example: Hexahedron

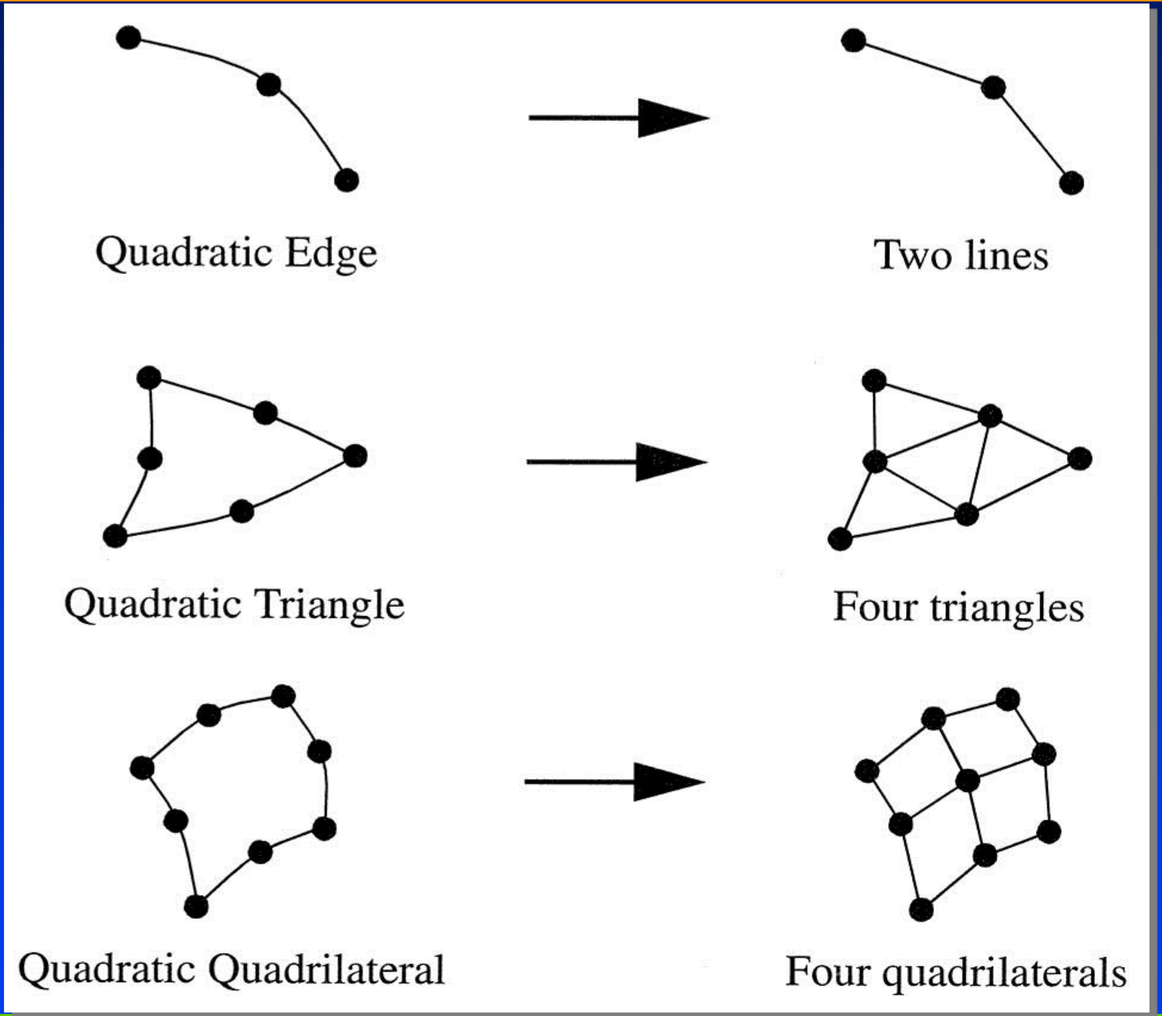
- Vertices listed in special order define topology



Non-Linear Cell Decomposition

- Non-linear cells must be linearized for visualization
- Break non-linear cells into linear cells

Non-Linear Cell Decomposition



Attribute Data

- Data values (attributes) usually assigned to vertices, as opposed to edges or faces
- Why?
- Interpolation concept easy to apply across edges and faces
- Common attributes include:
  - Temperature, density, velocity, pressure, heat flux, chemical concentration, others
- Scalars, vectors, tensors

Attribute Data

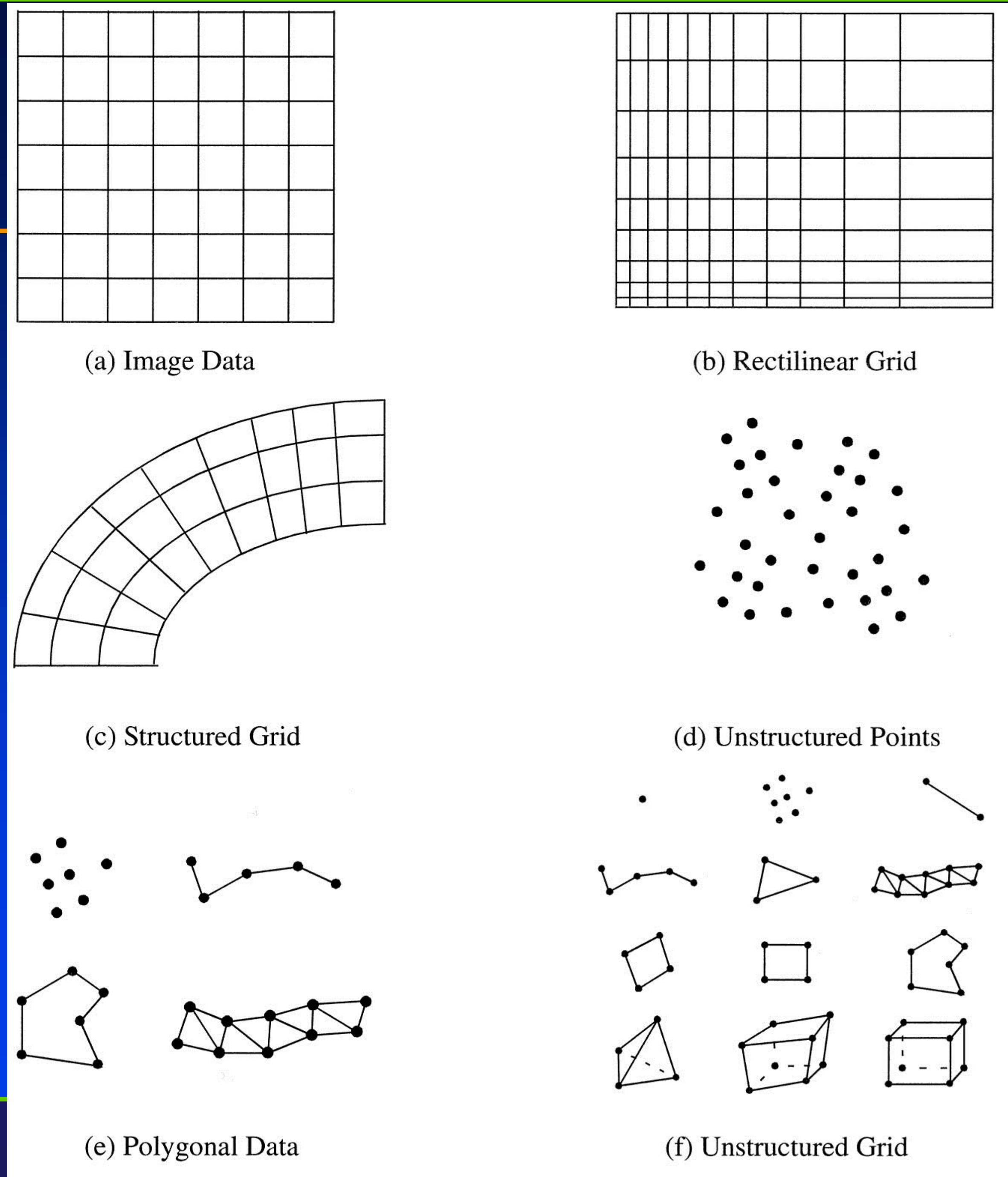
- **Scalar** data is data that is single-valued at all locations in a data-set
- Examples: temperature, stock price, elevation
- **Vector** data is data with magnitude and direction
- Examples: position, velocity, acceleration
- **Normals** (direction vectors) are vectors of magnitude 1
- **Texture coordinates** map a point from Cartesian space into a 1-D, 2-D or 3-D texture space
- Textures let us add color, transparency and other details to geometric shapes

Attribute Data

- **Tensors** are mathematical generalizations of vectors and scalars
- Usually written as matrices
- Tensor visualization is extremely difficult

Types of Data-sets

- Regular vs. irregular structure – refers to topology of data-set
- Data-sets with regular topology, we do not need to store connectivity information
- Points themselves can be regular or irregular
- If irregular, we need to store the positions
- Unstructured data must be explicitly represented
- High computational and storage costs usually


(a) Image Data
(b) Rectilinear Grid
(c) Structured Grid
(d) Unstructured Points
(e) Polygonal Data
(f) Unstructured Grid

Polygonal Data

- Vertices, edges, polygons, polylines, triangle strips, etc.
- Triangle strips can represent n triangles using only $n+2$ points, vs. $3n$ points normally required

Image Data

- Collection of points and cells on a regular, rectangular grid
- Also called a "raster"
- (Book uses word "lattice" – avoid!)
- 2D grid → image
- 3D grid → volume
- *i*-*j*-*k* coordinate system parallel to global *x*-*y*-*z* coordinate system
- Simple representation, but "curse of dimensionality"

Rectilinear Grid

- Regular grid, but spacing along axes can vary
- Need to store 3 extra arrays of length n_x, n_y, n_z – dimensions of the grid
- Each array stores spacing, basically

Structured Grid

- Regular topology, irregular geometry
- Curvilinear grids most common type

Unstructured Points

- No topology, irregular geometry
- Also called **point clouds**

Unstructured Grid

- Irregular topology and geometry
- Any combination of cells permitted
- Encountered in relatively few applications
- e.g., computational geometry

VTK Data Representations

- vtkFloat Array
- vtkImageData
- vtkRectilinearGrid
- vtkStructuredGrid
- vtkPolyData
  - vtkCellArray
- vtkUnstructuredGrid

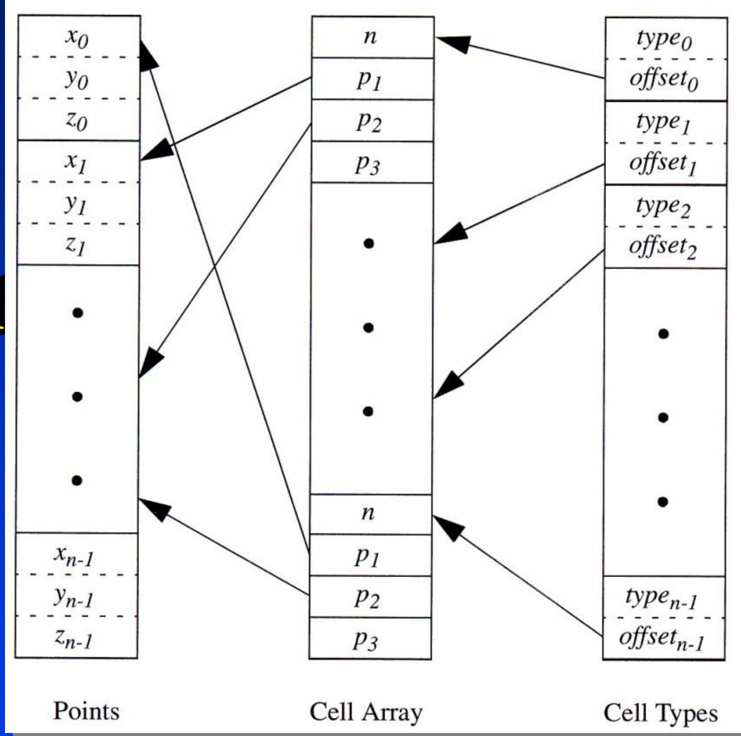


Figure 5–13 The data structure of the class `vtkUnstructuredGrid`. (This is a subset of the complete structure. See Chapter 8 for complete details.)

VTK Data Representations

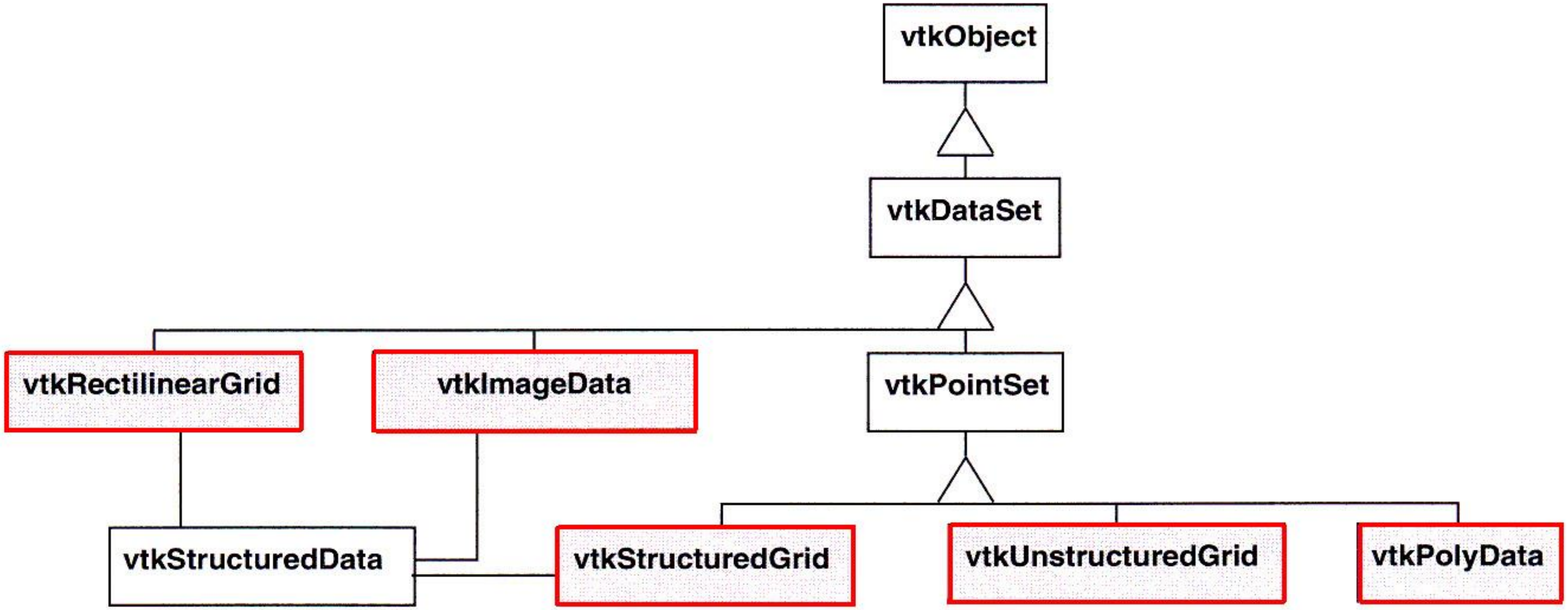


Figure 5–14 Dataset object diagram. The five datasets (shaded) are implemented in VTK.

VTK Cell Types

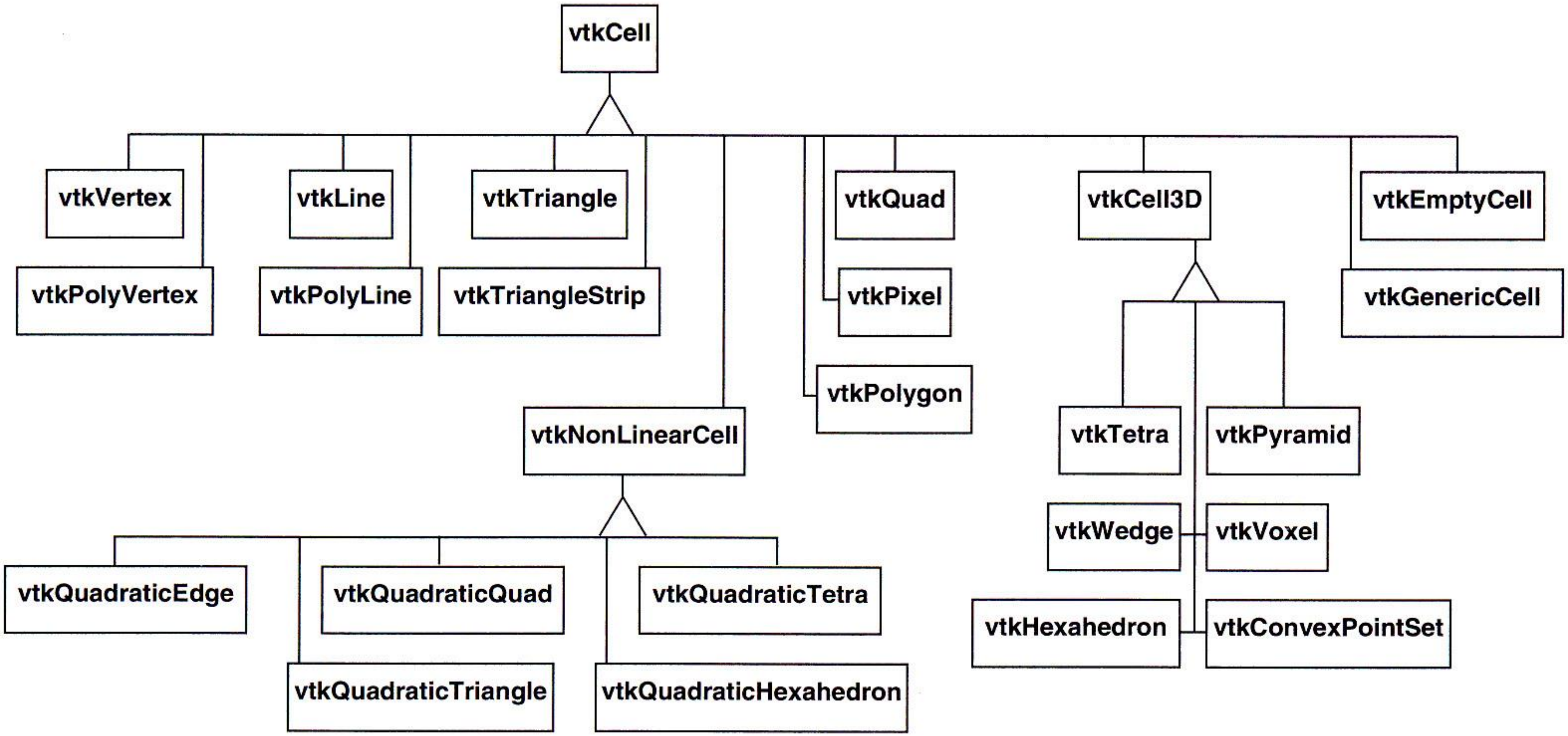
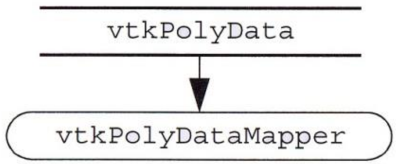
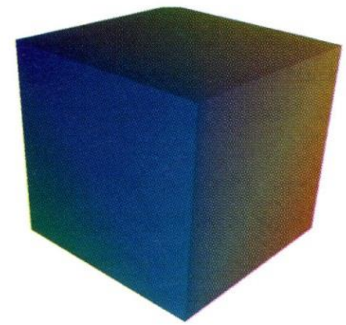


Figure 5–15 Object diagram for twenty concrete cell types in VTK. vtkEmptyCell represents NULL cells. vtkGenericCell can represent any type of cell. Three-dimensional cells are subclasses of vtkCell3D. Higher order cells are subclasses of vtkNonLinearCell.

Example: Cube.cxx

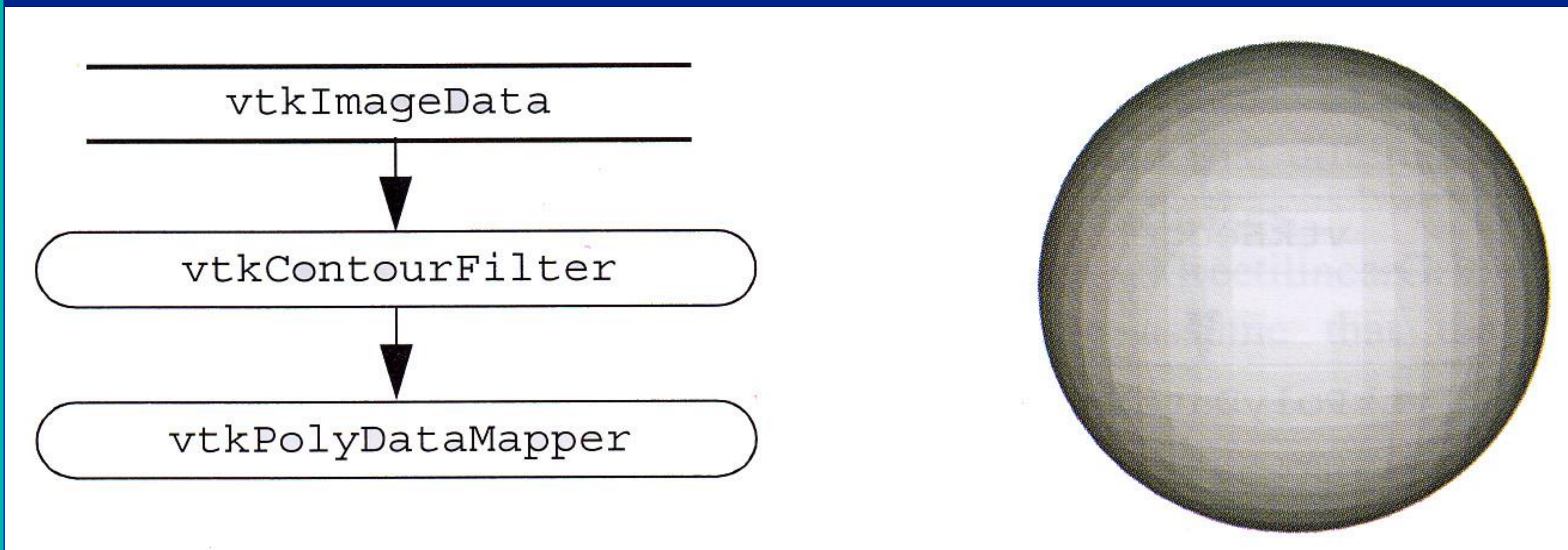




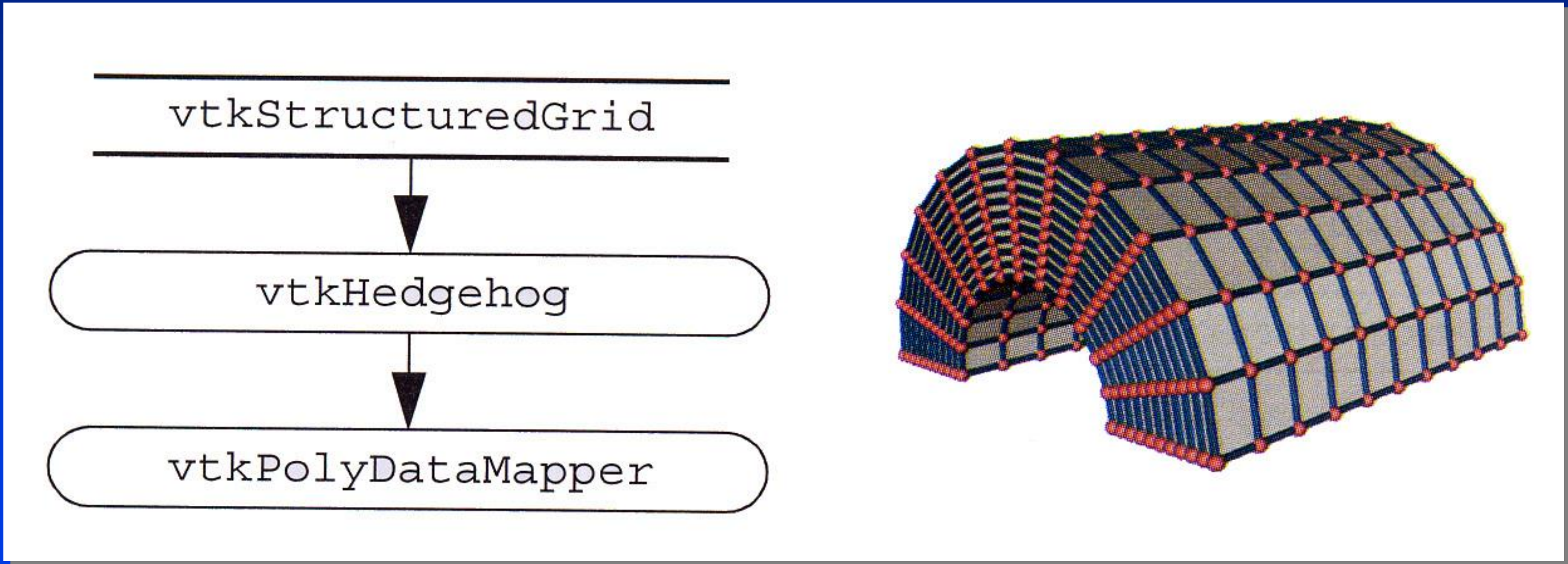
```
vtkPolyData *cube = vtkPolyData::New();
vtkPoints *points = vtkPoints::New();
vtkCellArray *polys = vtkCellArray::New();
vtkFloatArray *scalars = vtkFloatArray::New();

for (i=0; i<8; i++) points->InsertPoint(i,x[i]);
for (i=0; i<6; i++) polys->InsertNextCell(4,pts[i]);
for (i=0; i<8; i++) scalars->InsertTuple1(i,i);

cube->SetPoints(points);
points->Delete();
cube->SetPolys(polys);
polys->Delete();
cube->GetPointData()->SetScalars(scalars);
scalars->Delete();
```

Figure 5–17 Creation of polygonal cube (`Cube.cxx`).

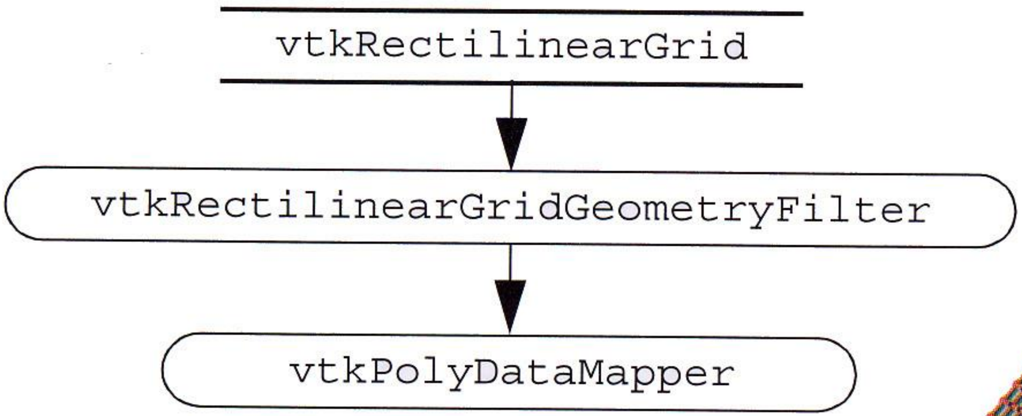
Example: Vol.cxx

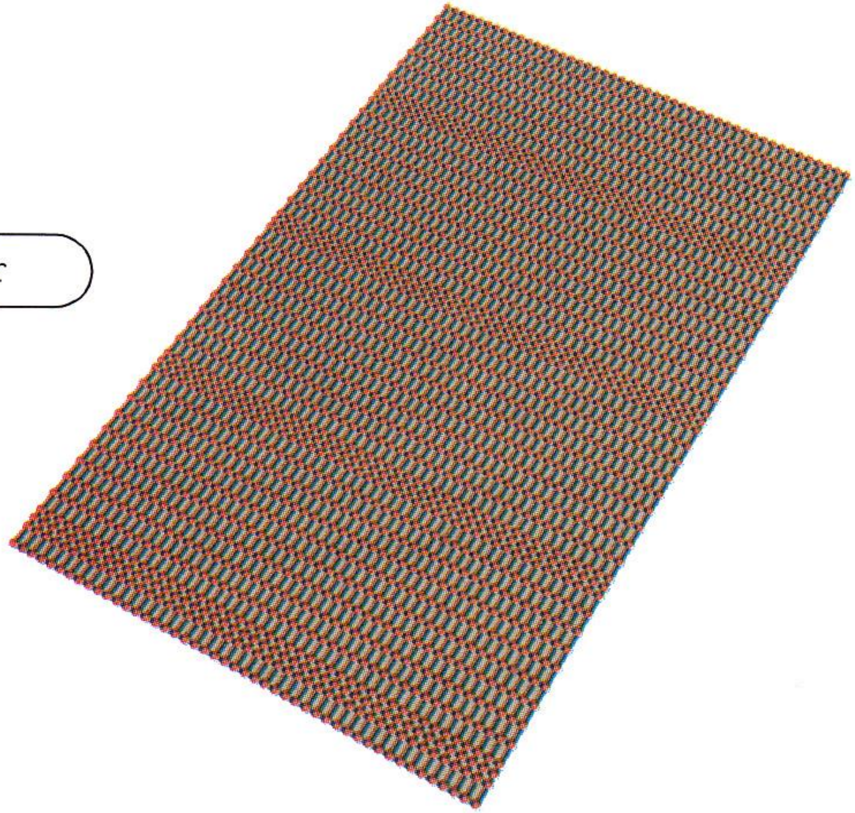


Example: SGrid.cxx

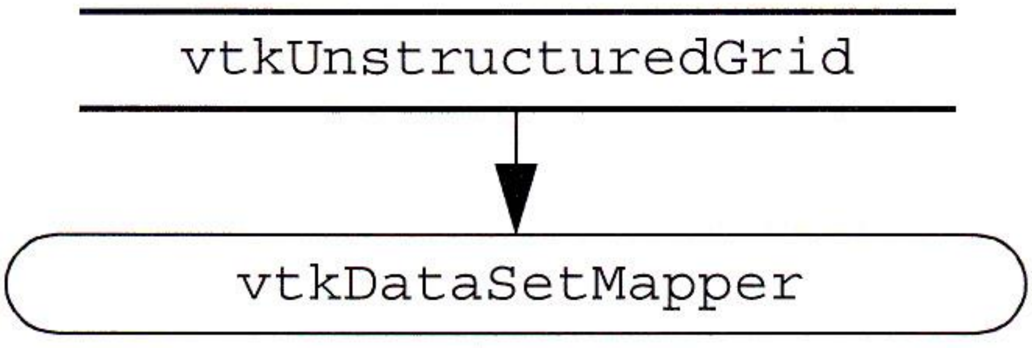


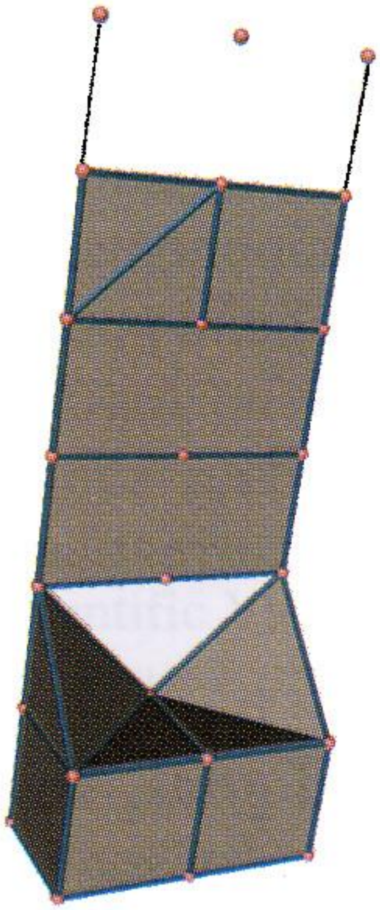
Example: RGrid.cxx





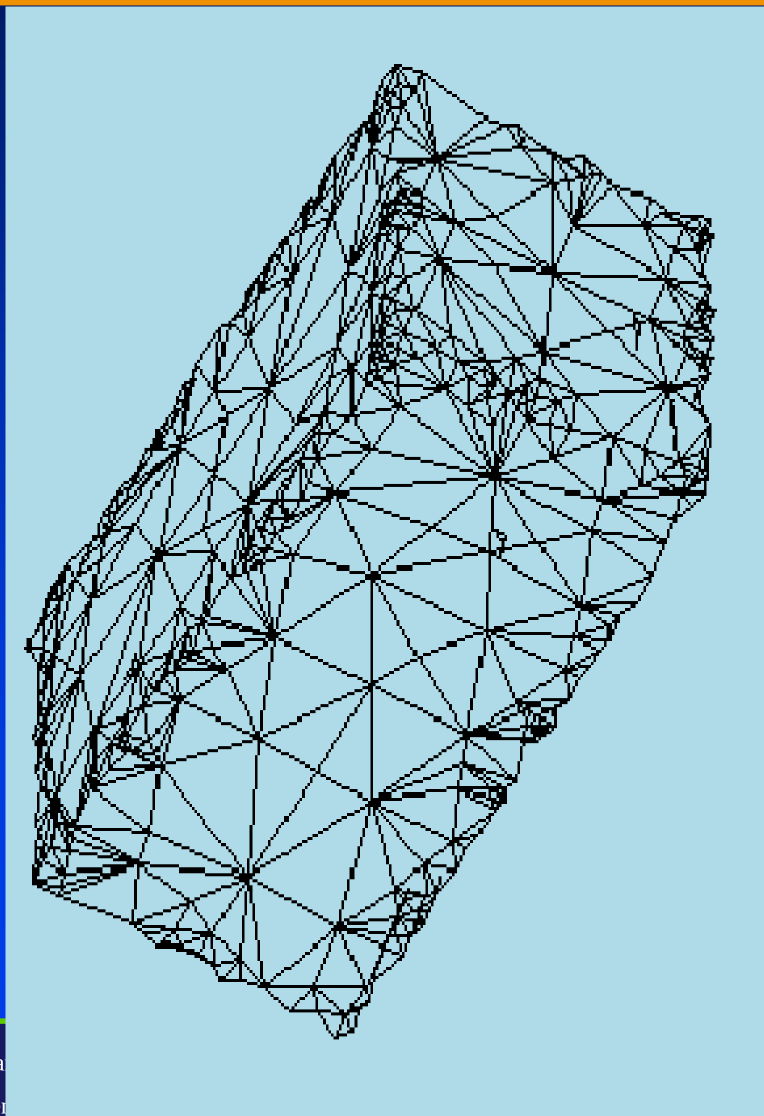
Example: UGrid.cxx

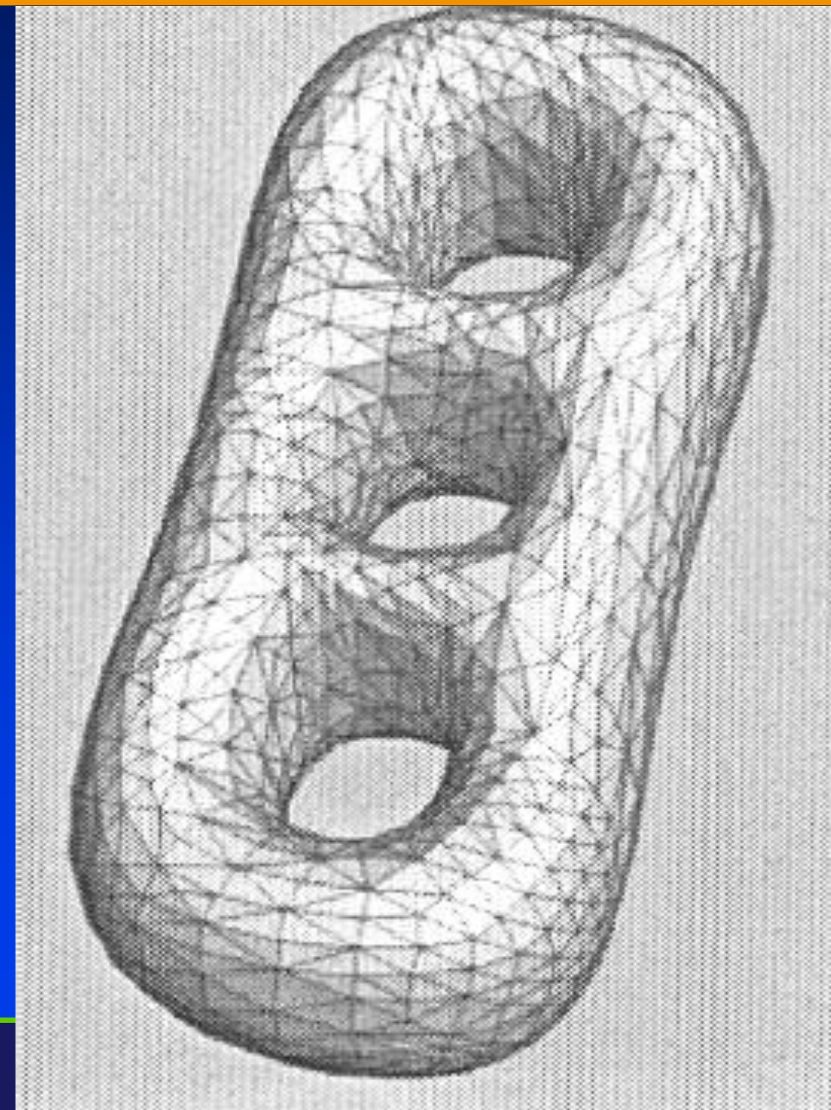




From 3D data clouds to surface meshes: Triangulation of data sets

Mesh Objects





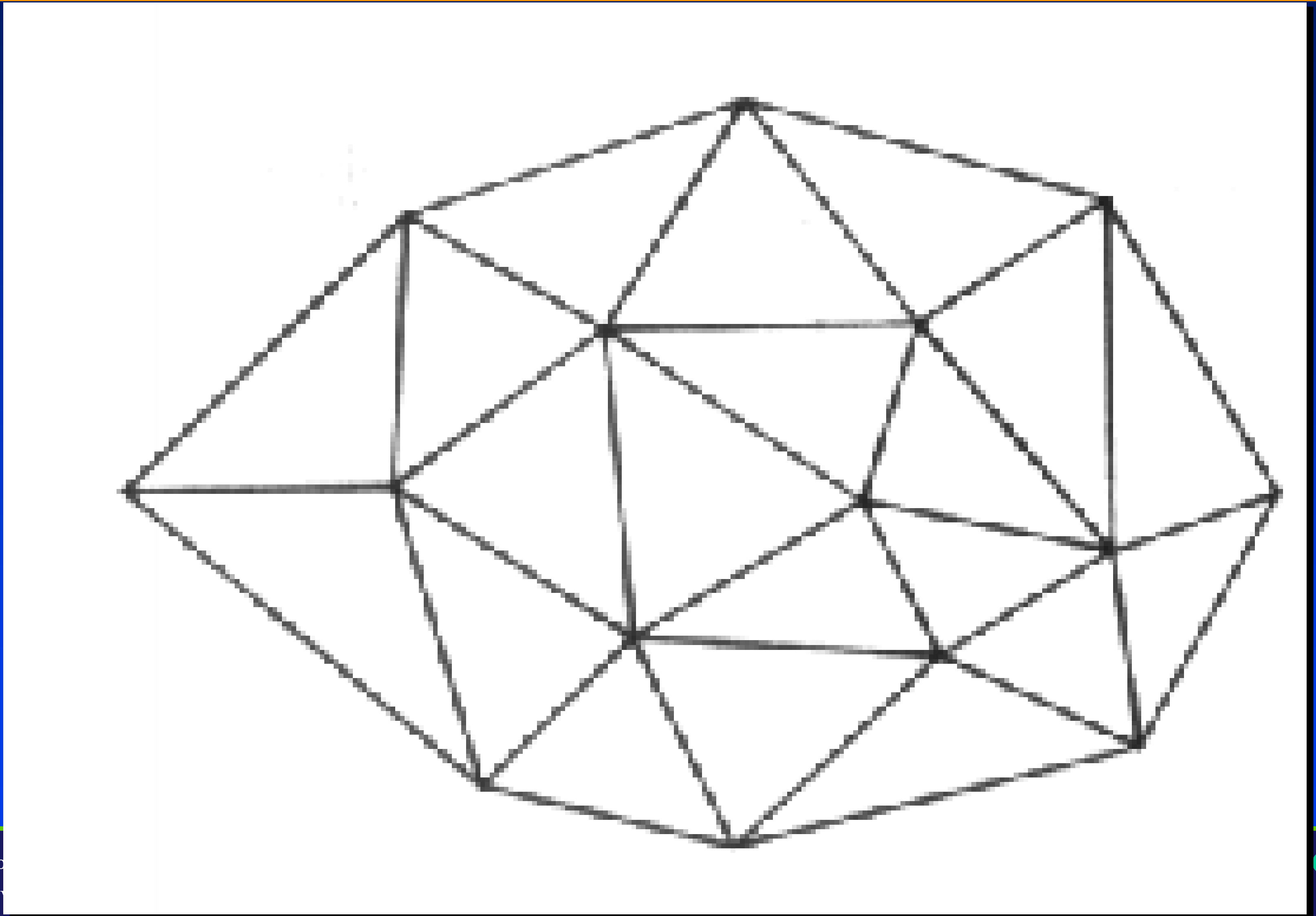
Why Triangular Meshes are Needed?

- A simple piecewise linear approximation of 3D shapes of complex objects
- Appropriate for processing in graphics hardware
- Suitable for deformation and manipulation of the object surfaces

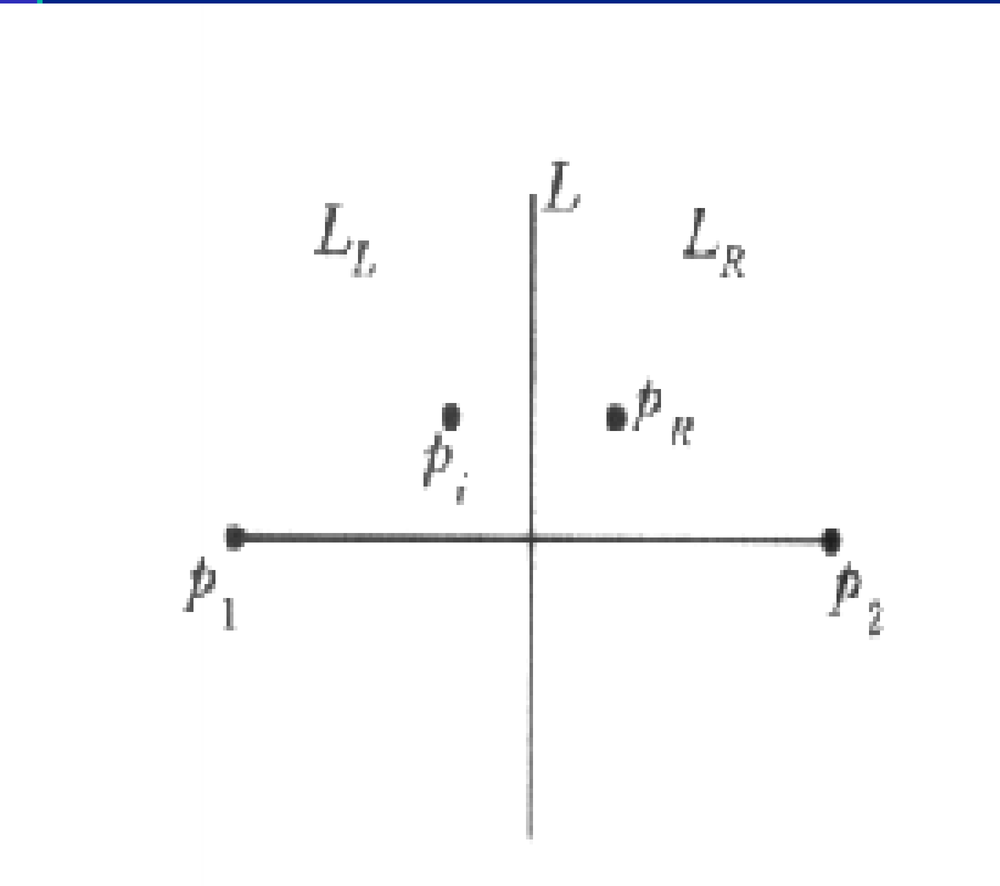
Main Topics

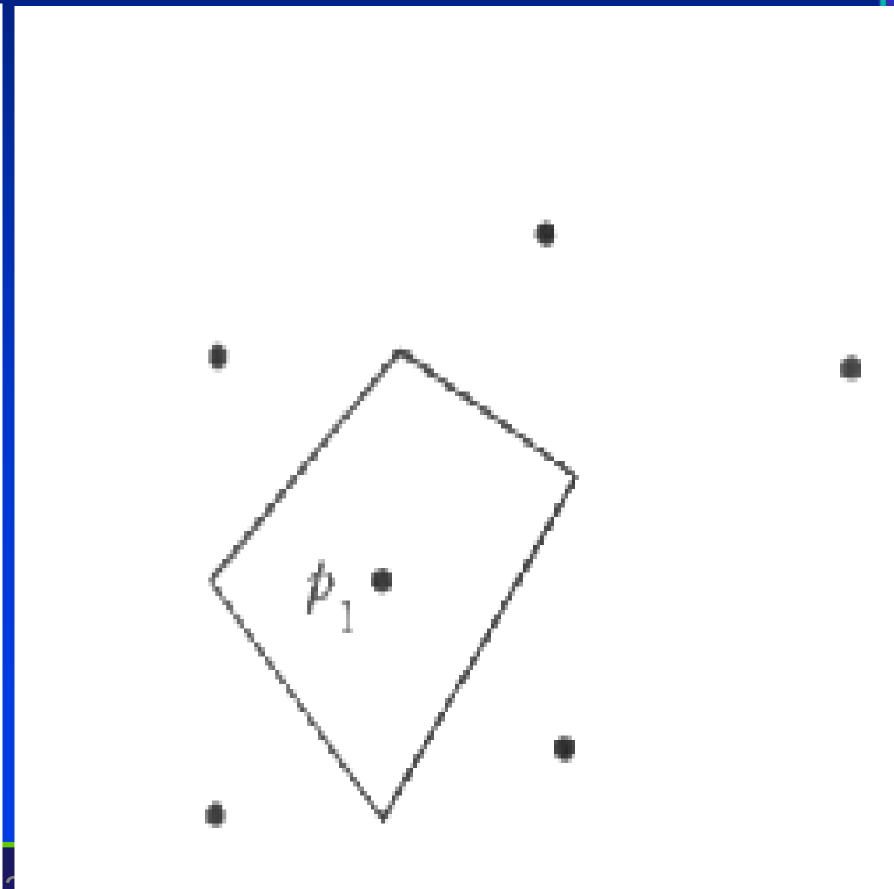
- Planar triangulation
  - Voronoi diagram
  - Delaunay triangulation
- 3D triangulation based on a physical model --- balloon inflation
- Marching cubes

Planar Triangulation

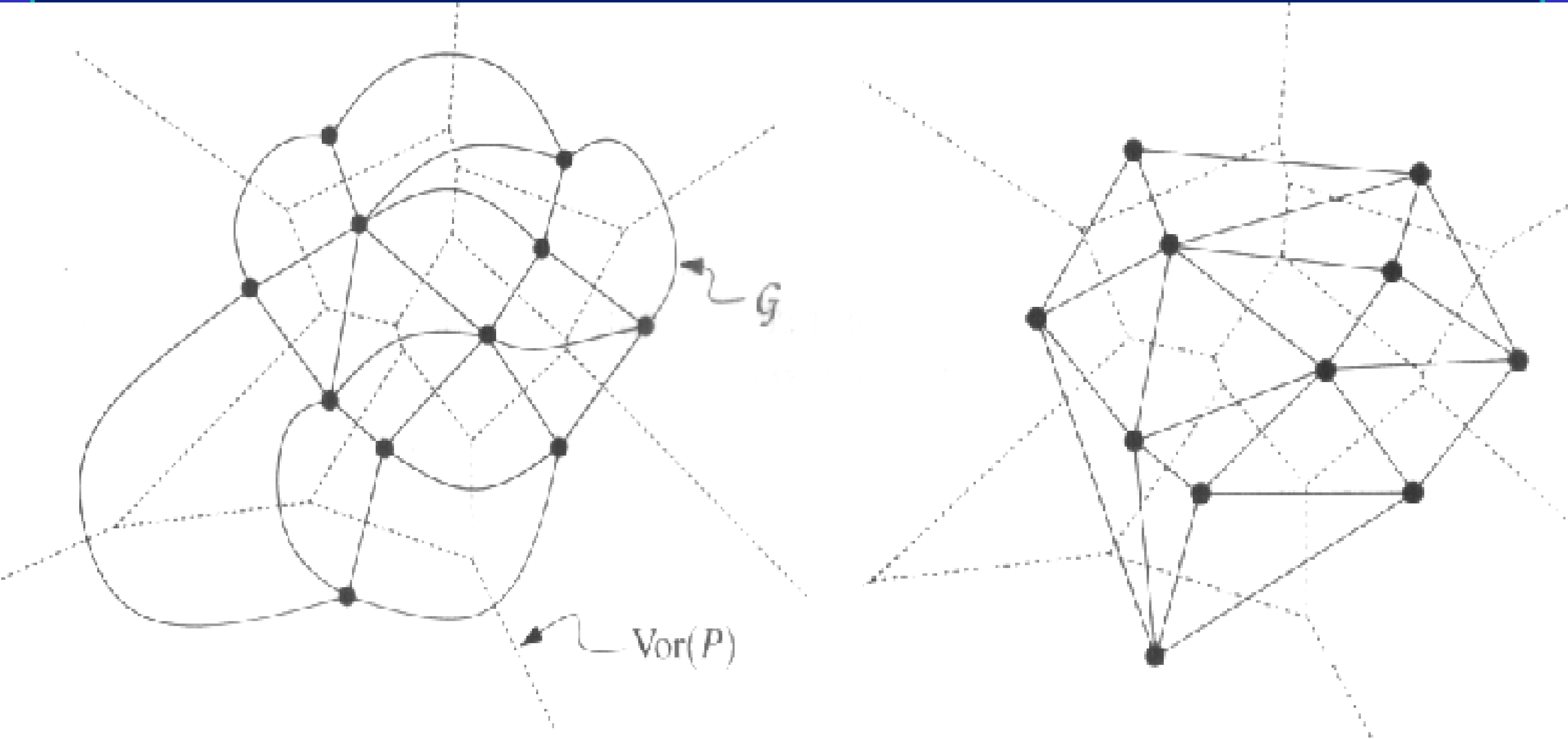


Voronoi Diagrams

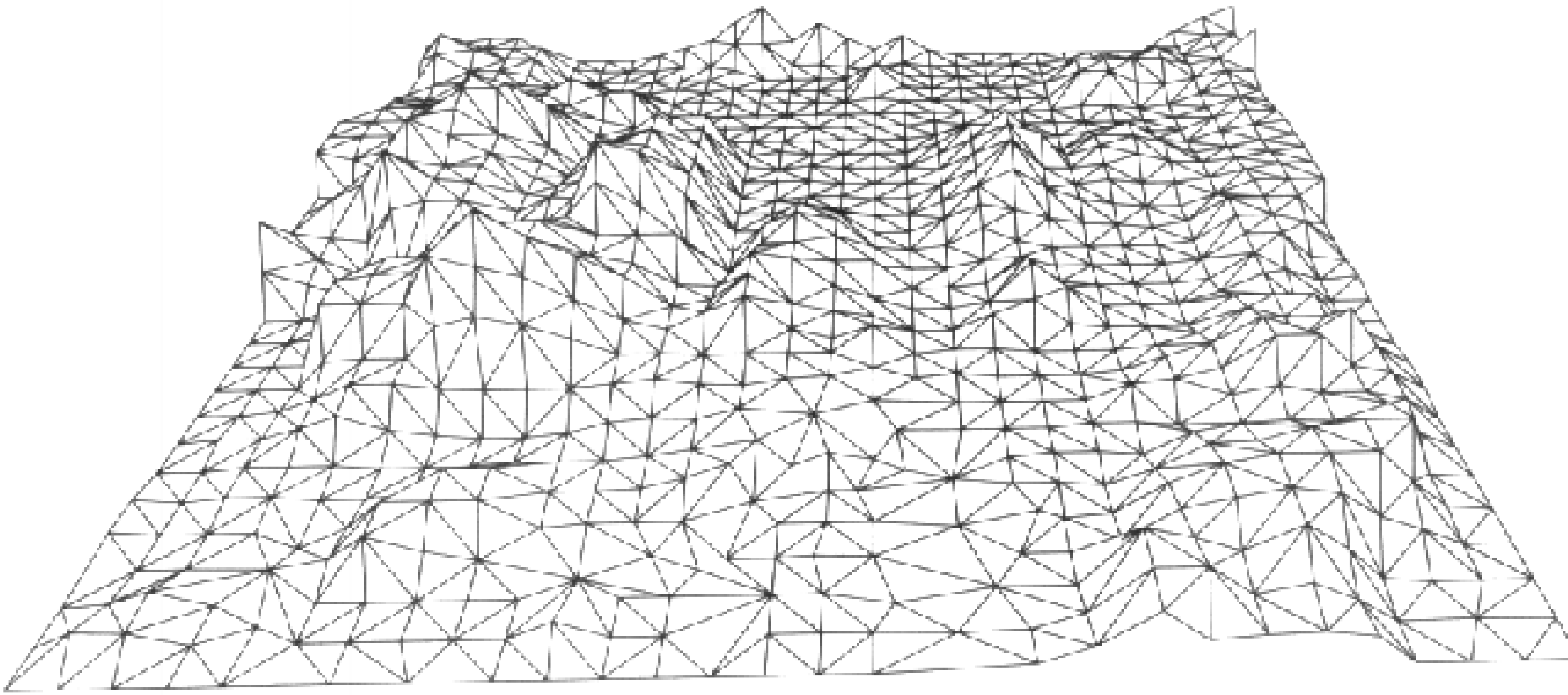




Dual Graph of a Voronoi Diagram



Triangulation of Terrain Data



Edge Flipping

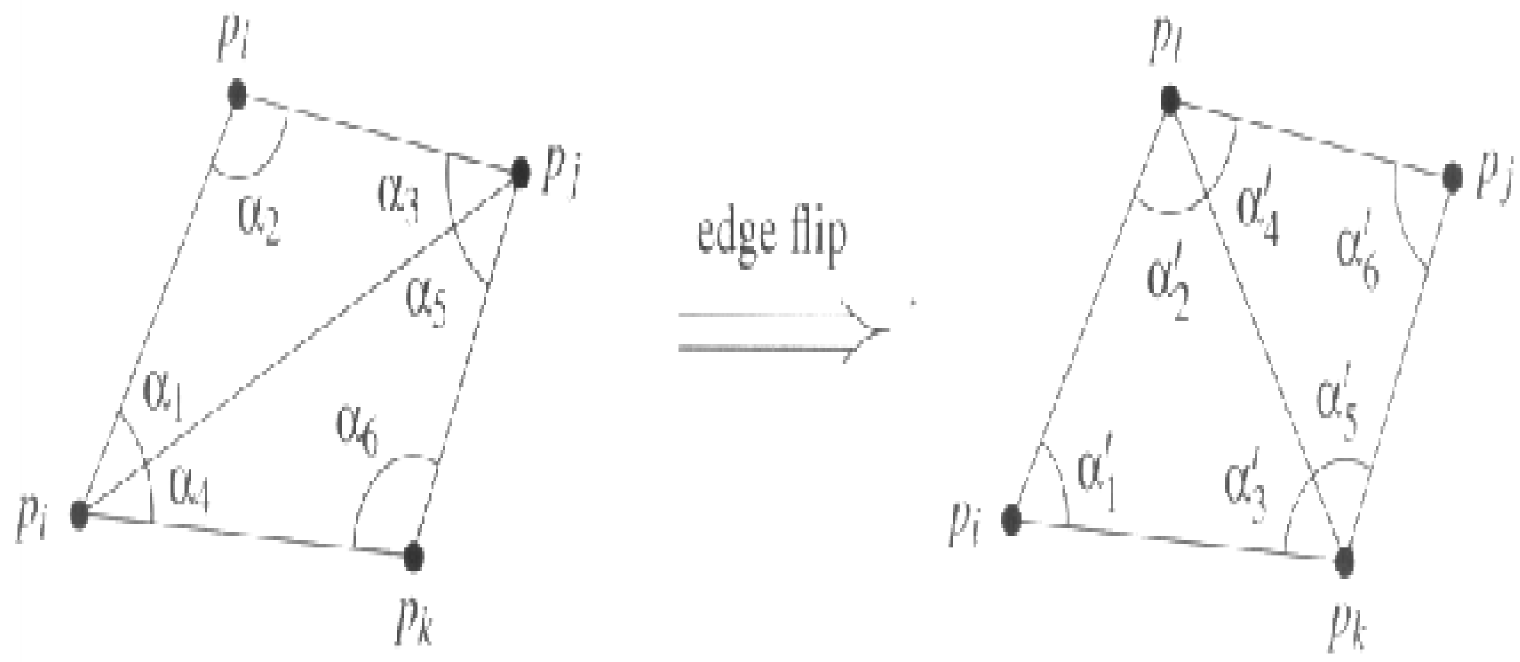
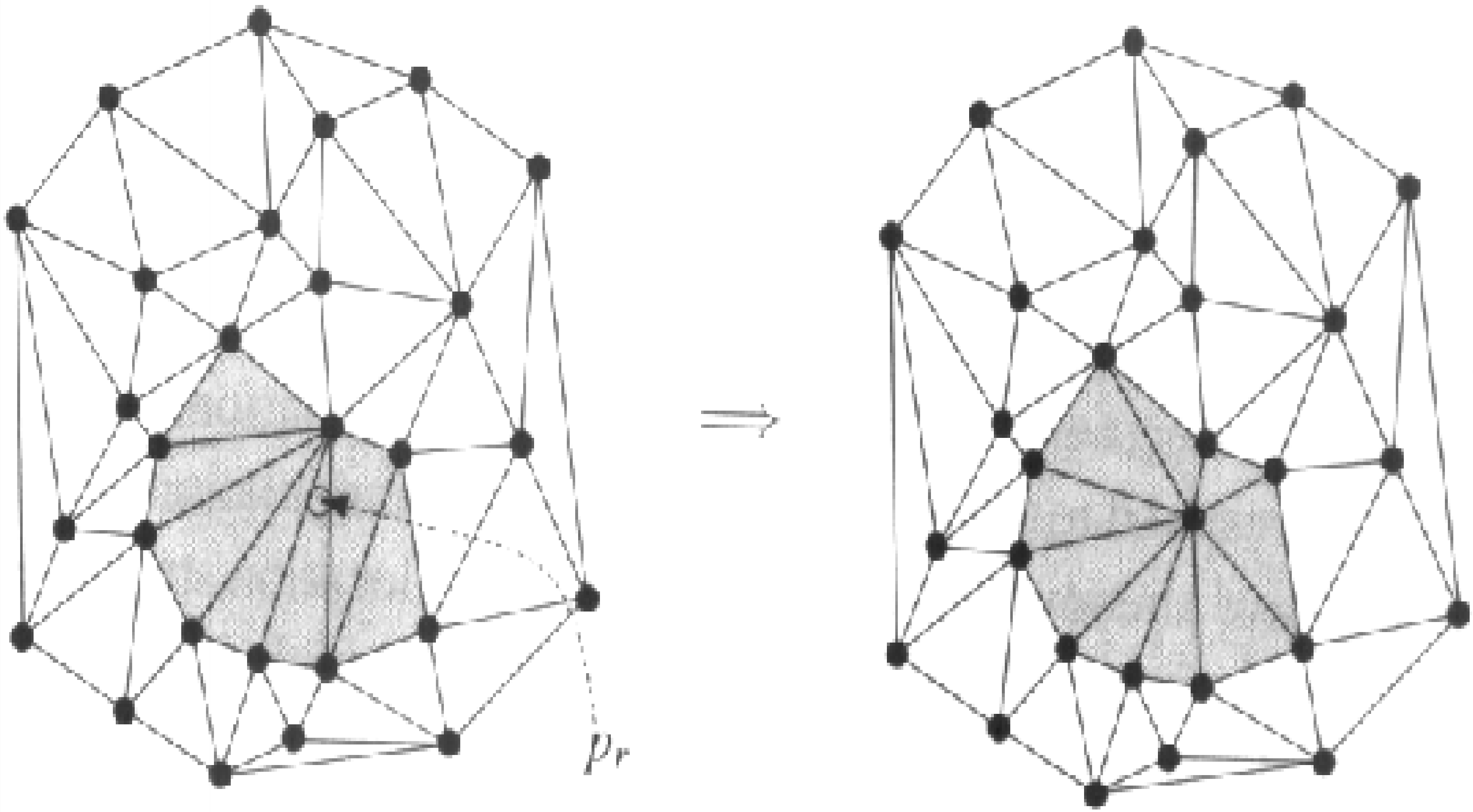


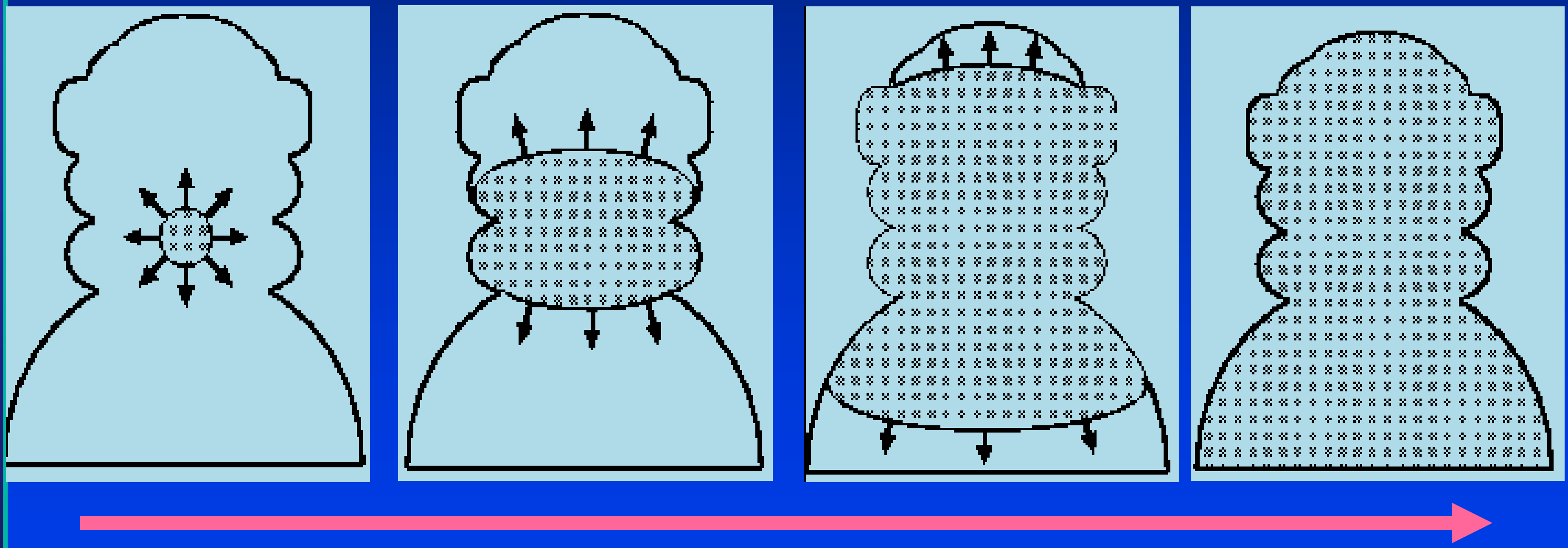
Figure 9.4
Flipping an edge

Incremental Triangulation Algorithm



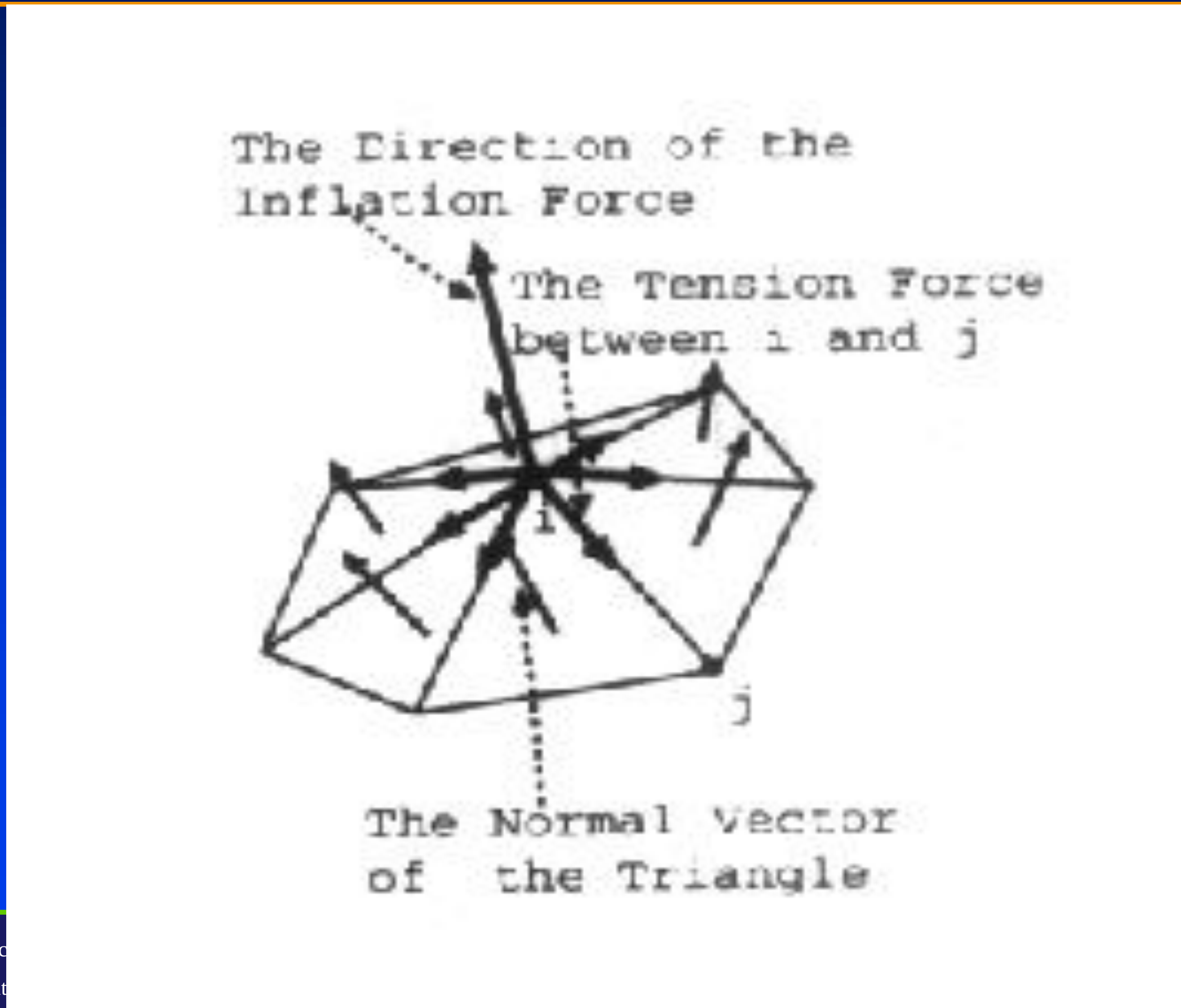
Progressive Balloon Inflation

- Balloon inflation for surface fitting

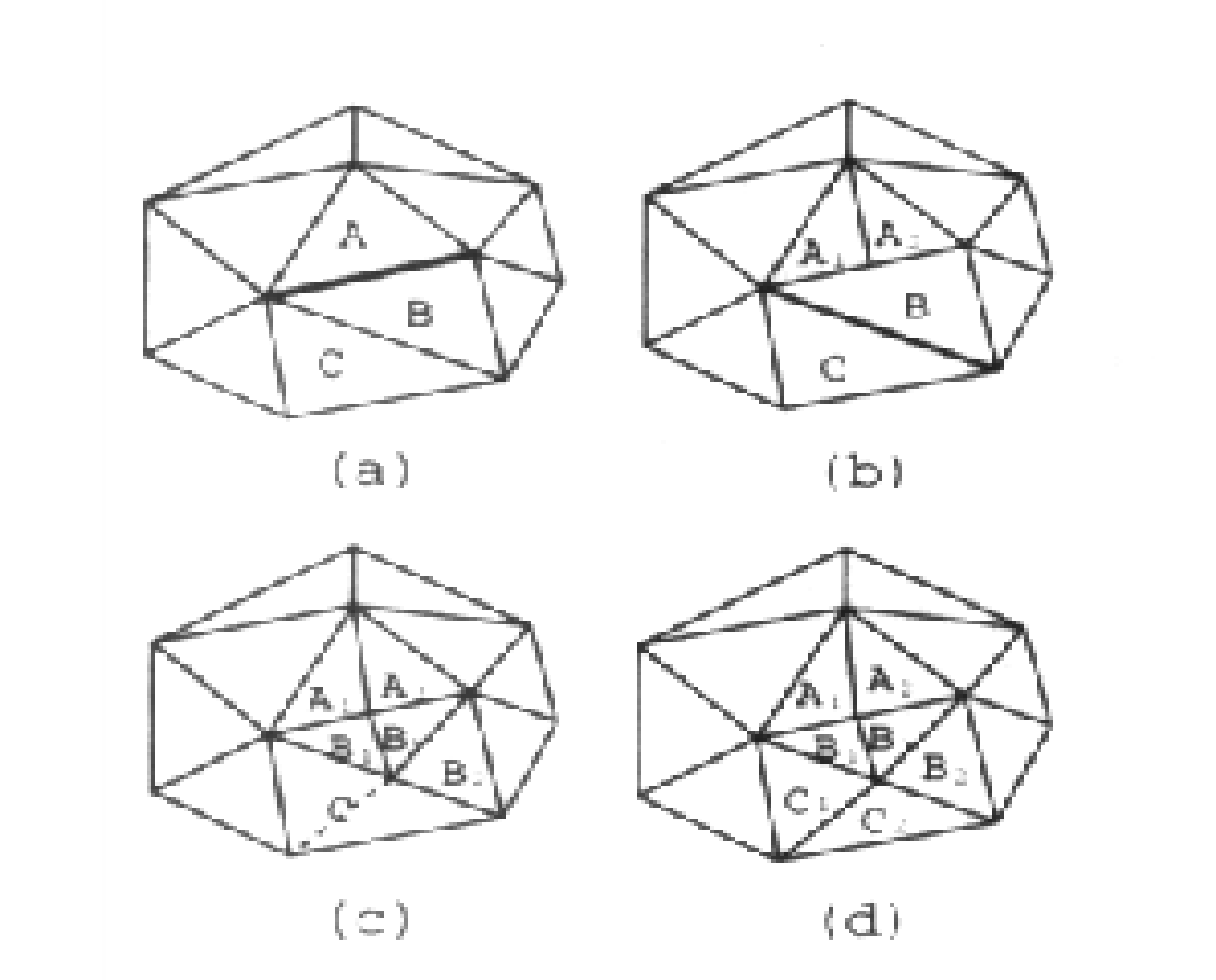


STONY BROOK
STATE UNIVERSITY OF NEW YORK

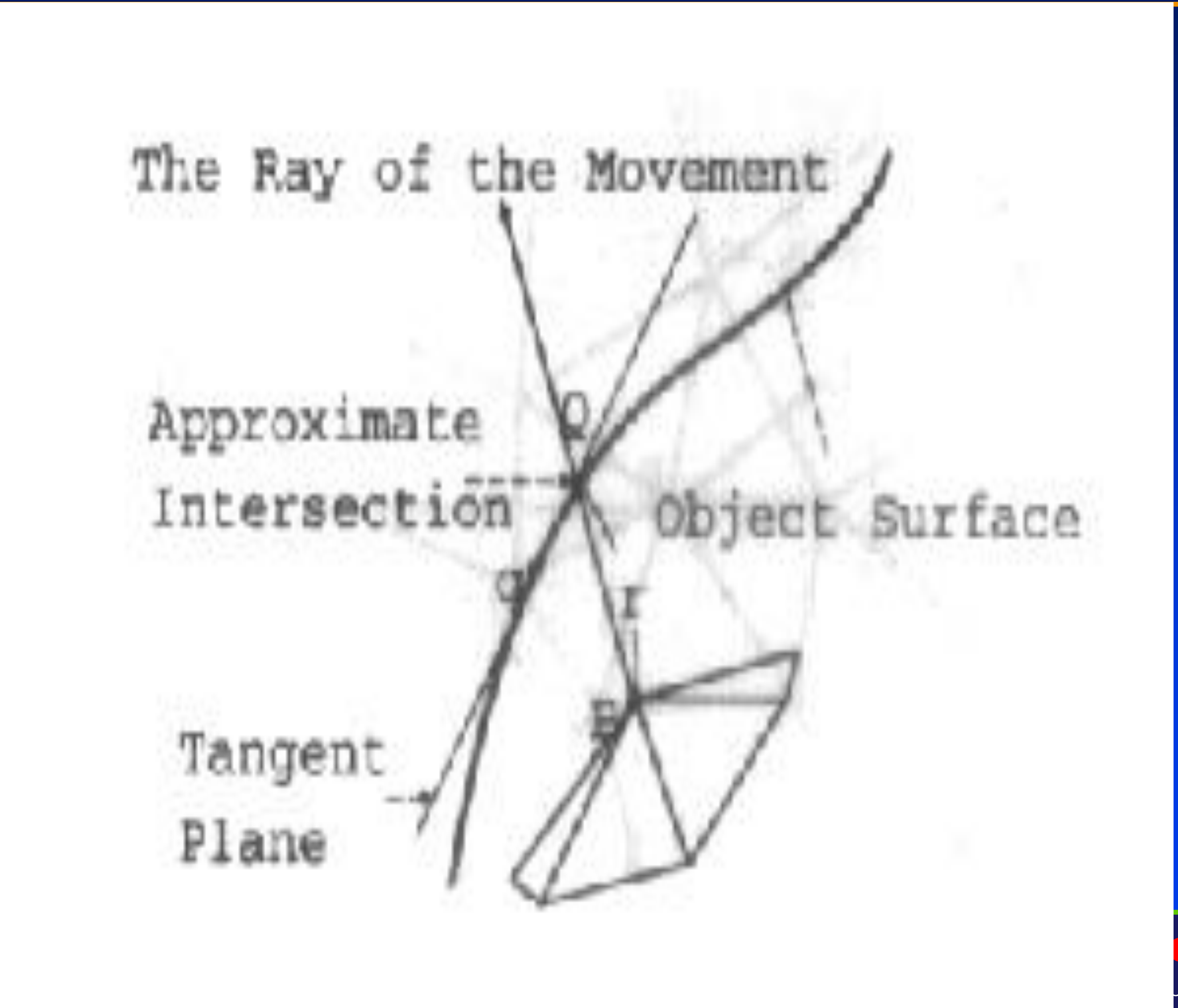
Physical Model for Balloon Inflation



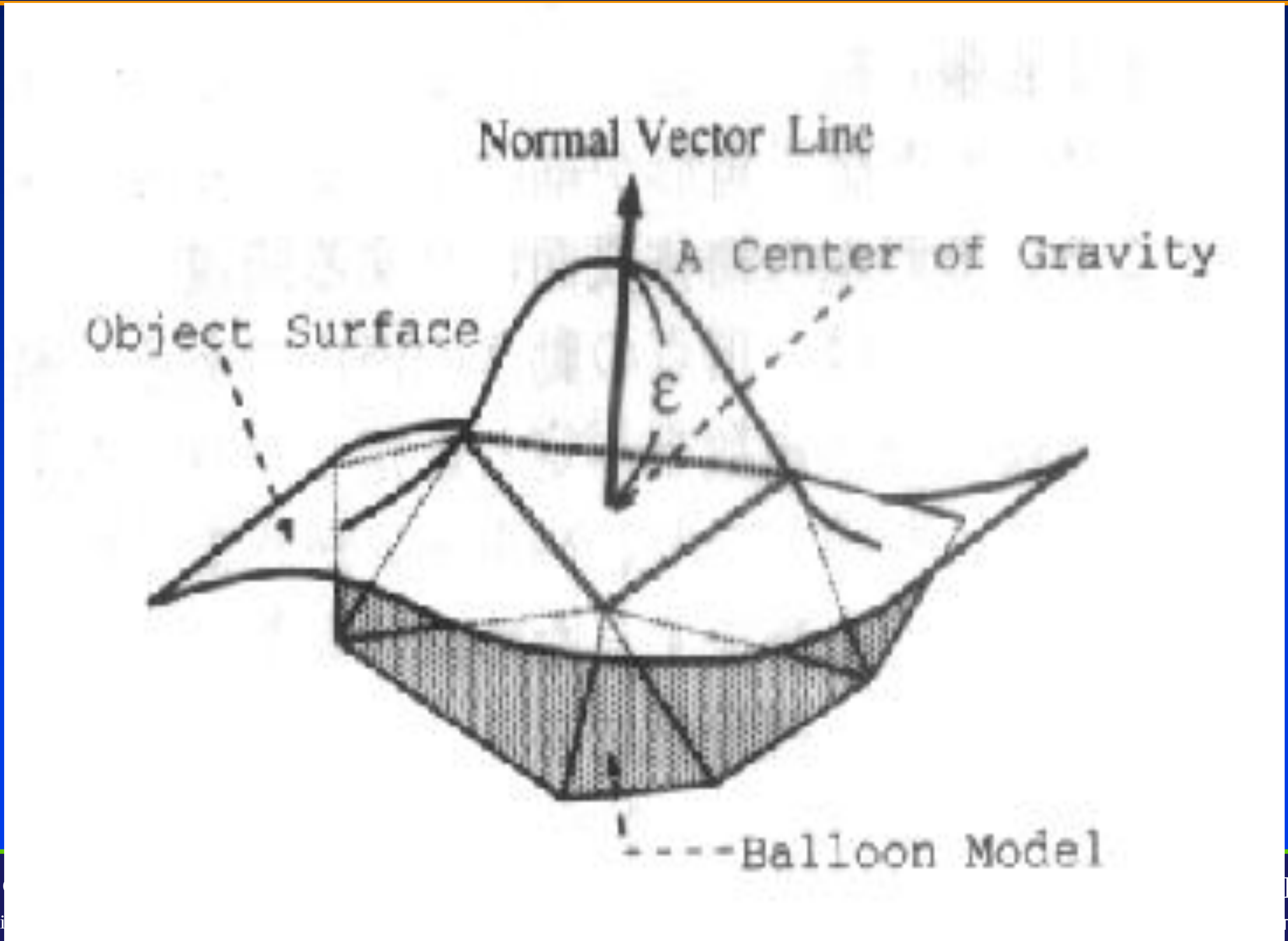
Subdivision of Triangular Faces



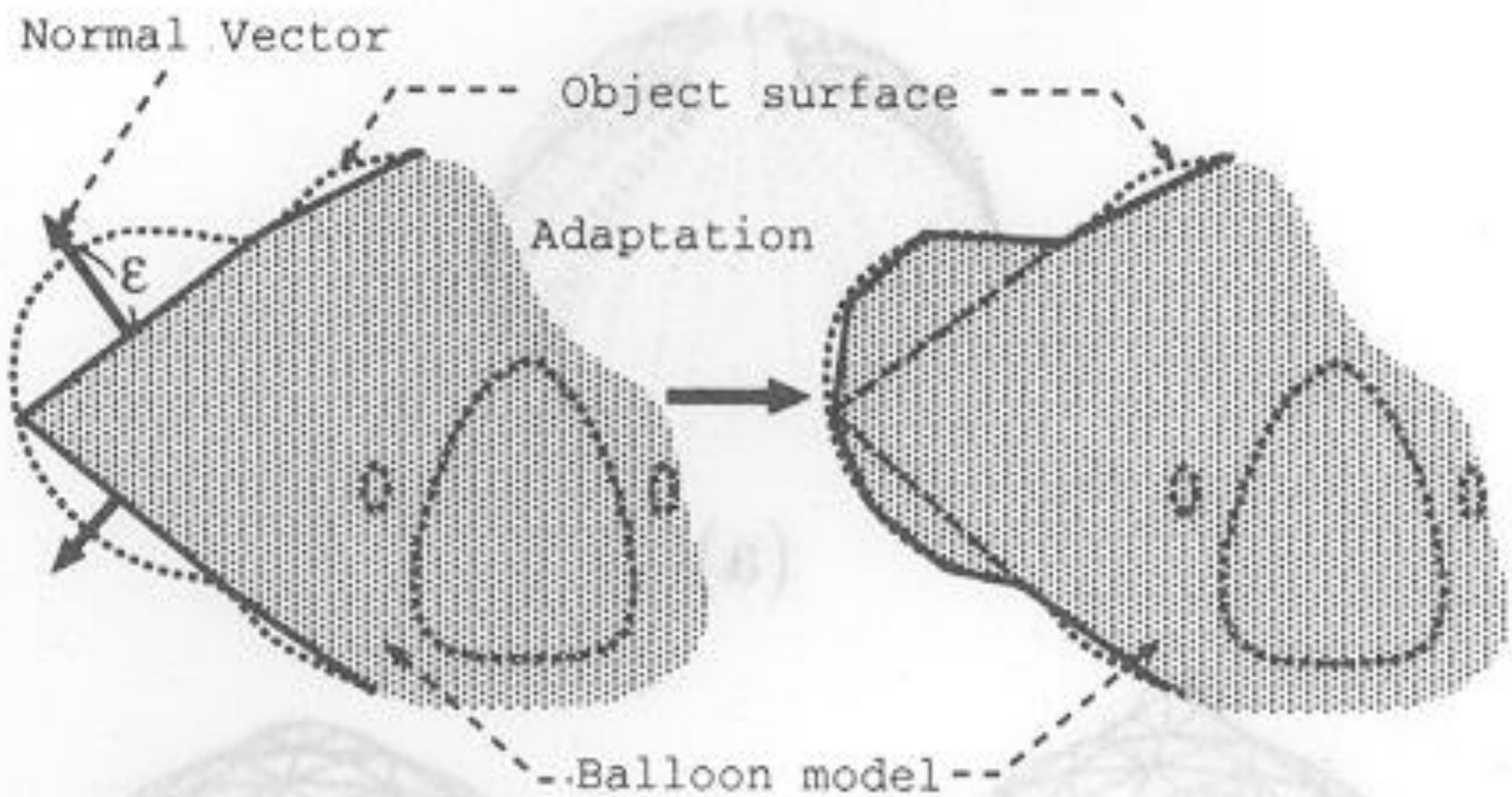
Touching of Balloon at Data



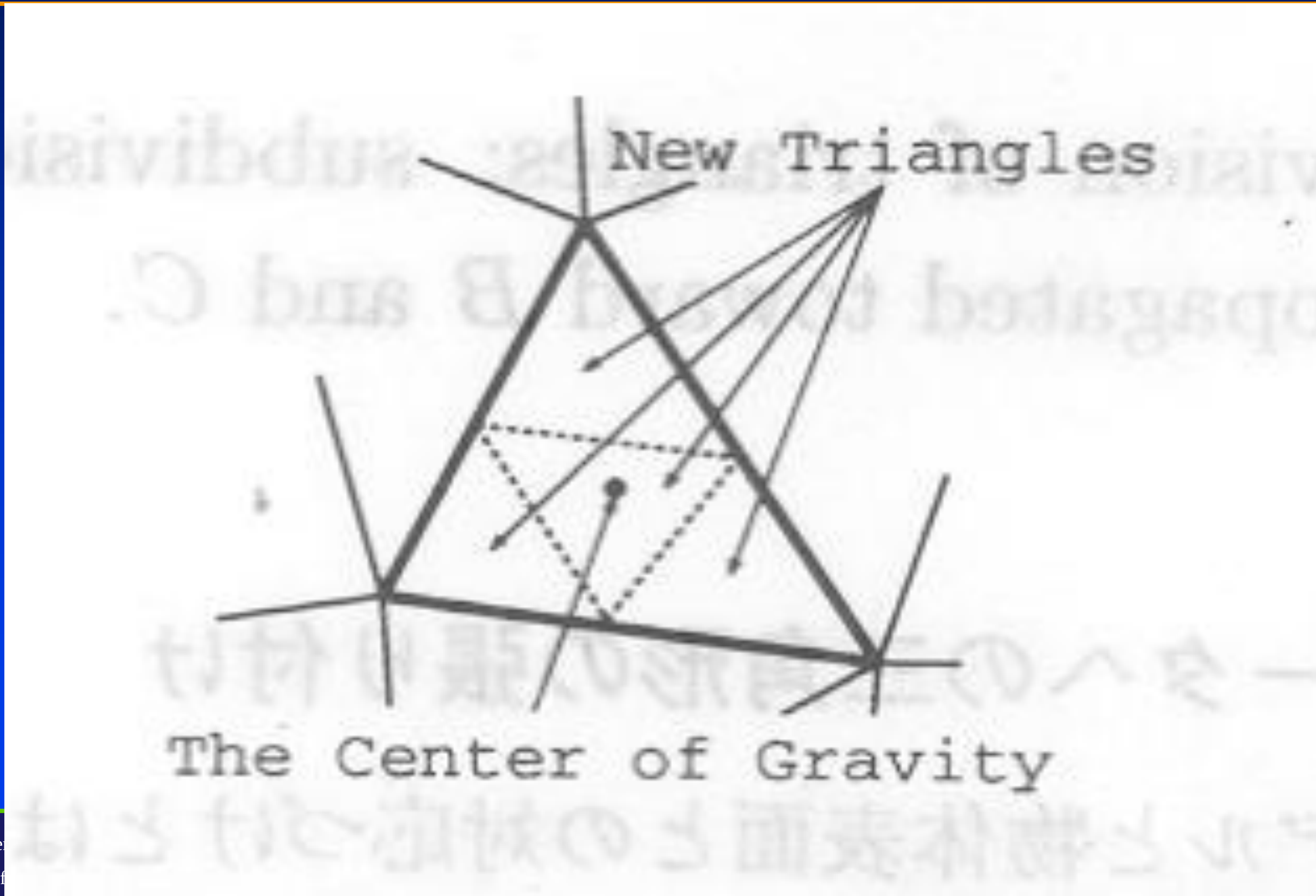
Approximating Errors



Adaptation of Local Fitting

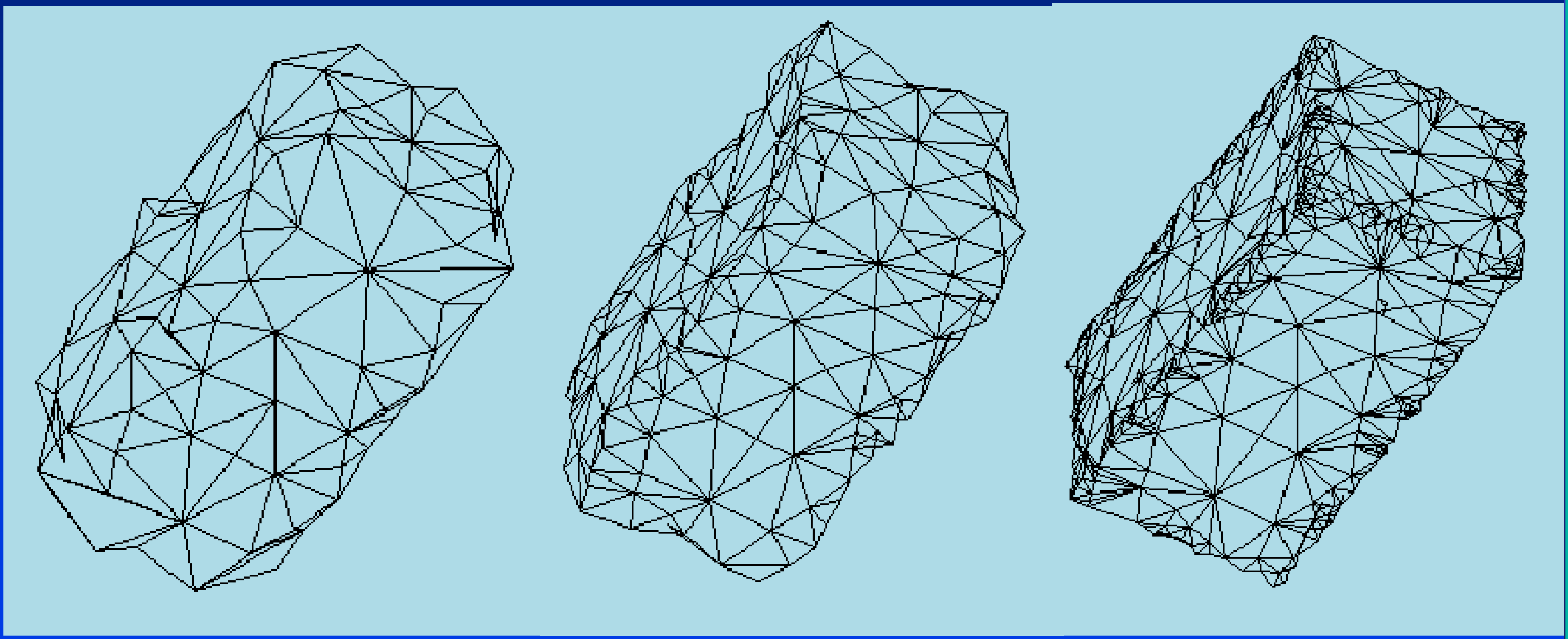


Subdivision for Fitting

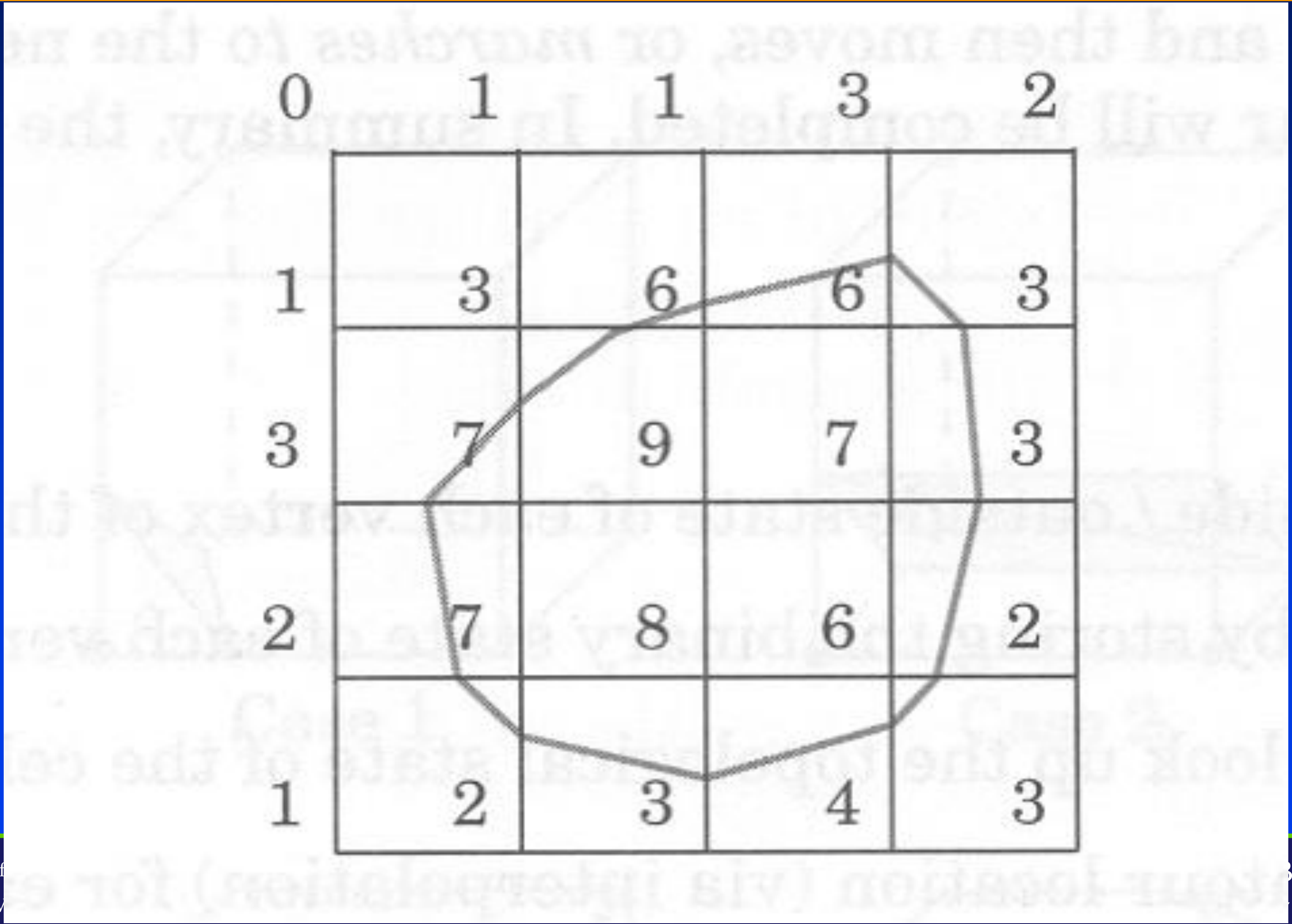


Hierarchy of Triangular Meshes

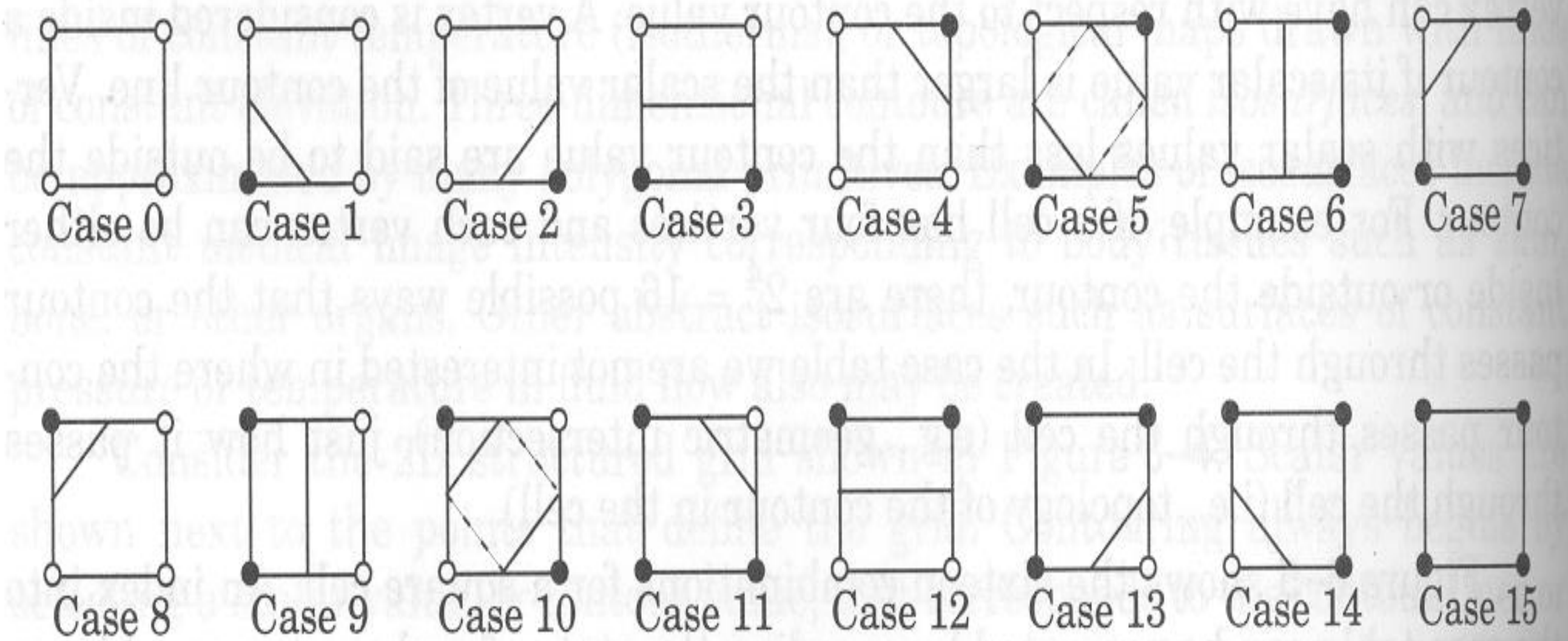
- Meshes with triangles of different sizes



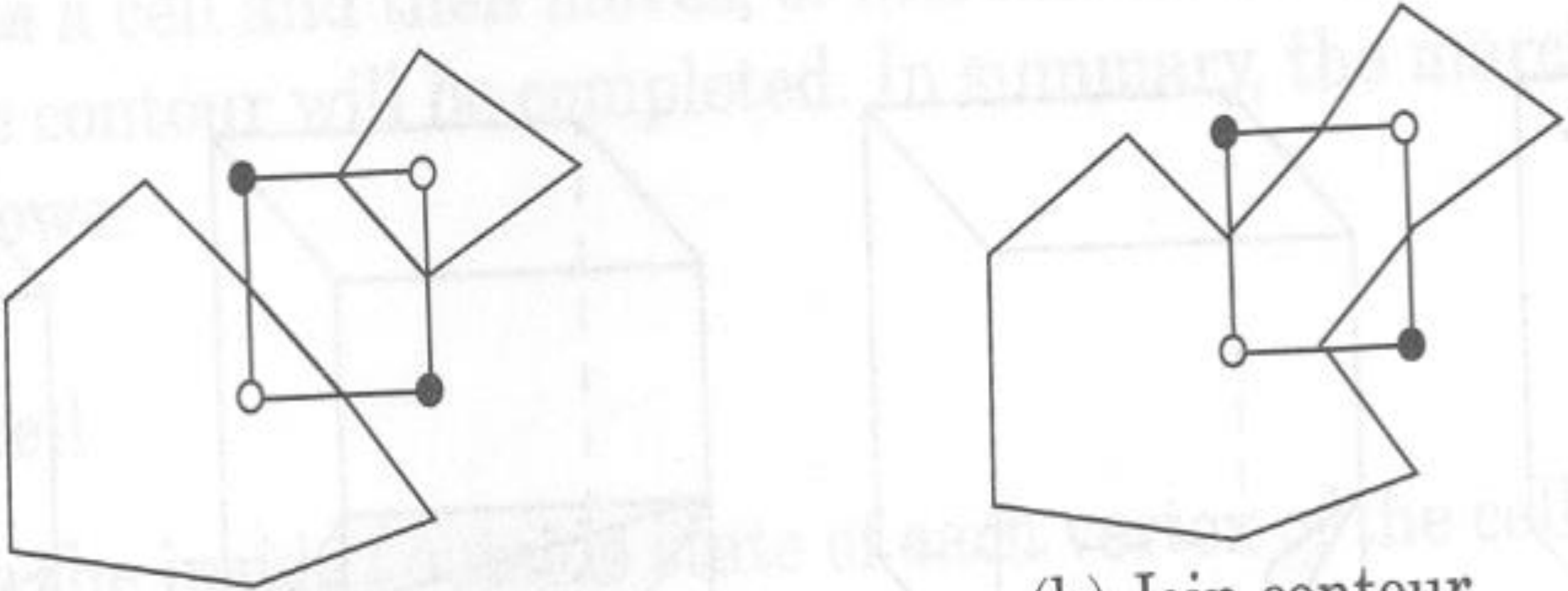
Contouring Using Marching Squares



Marching Squares Cases



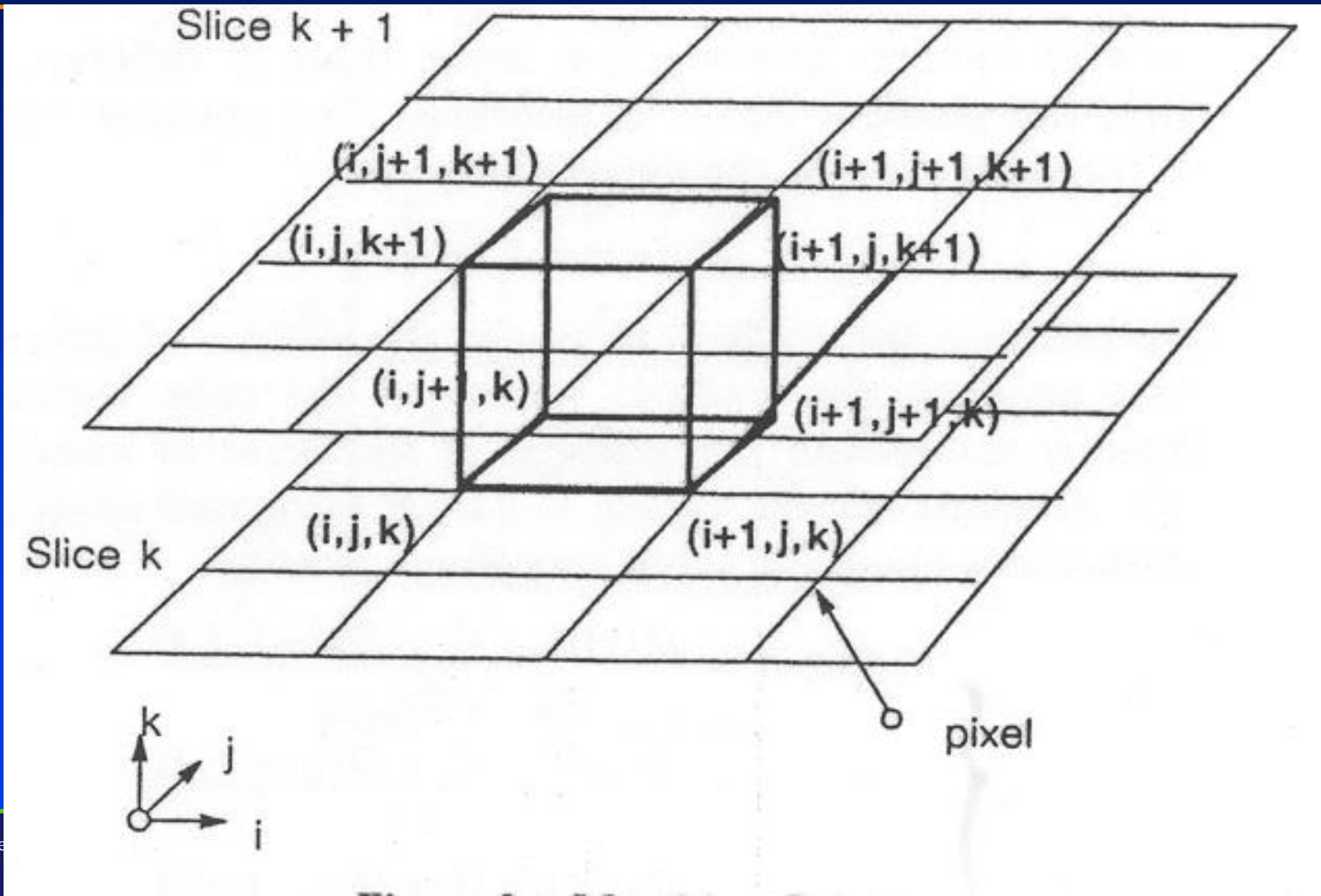
Ambiguity in Connecting Edges



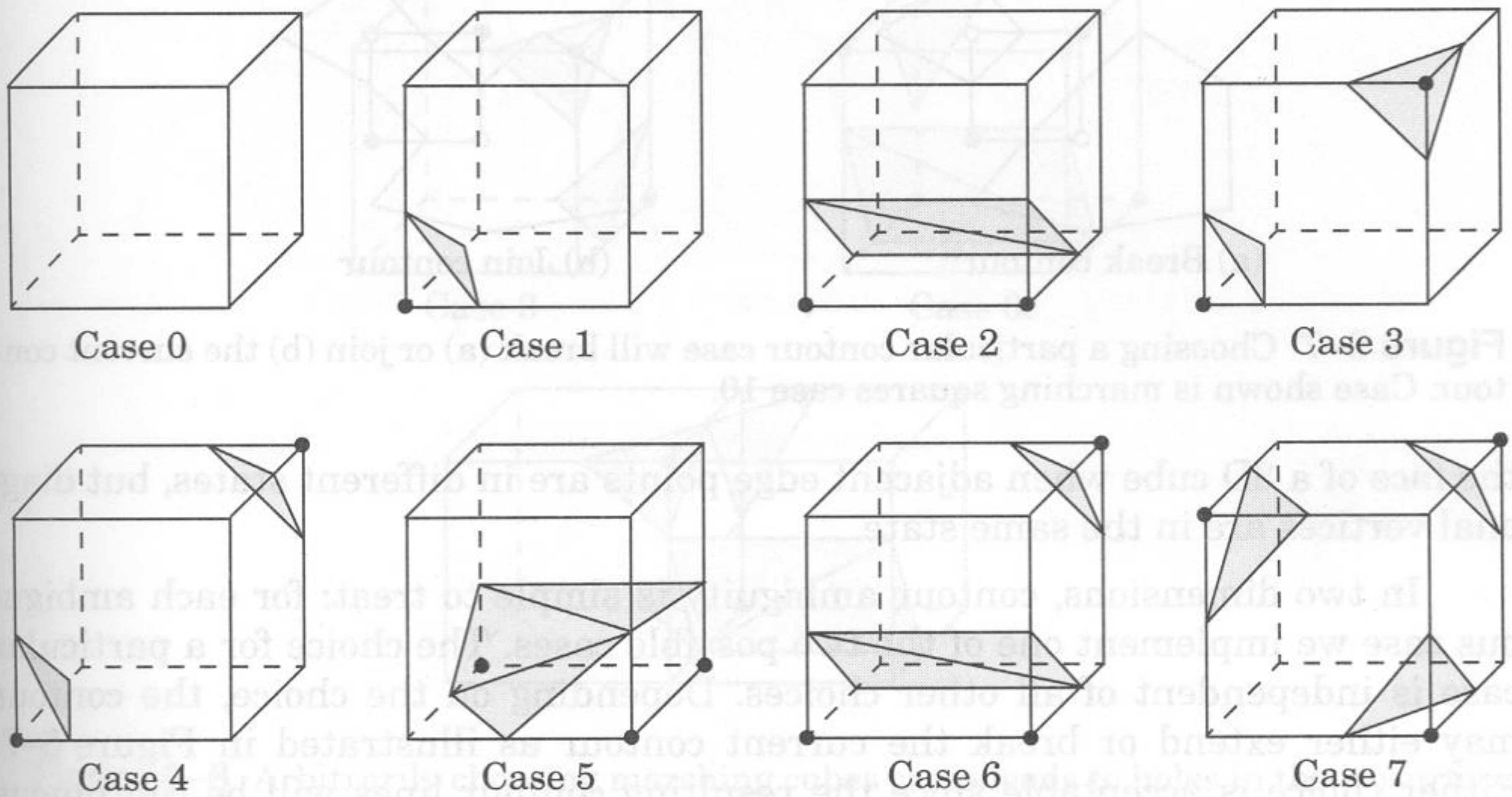
(a) Break contour

(b) Join contour

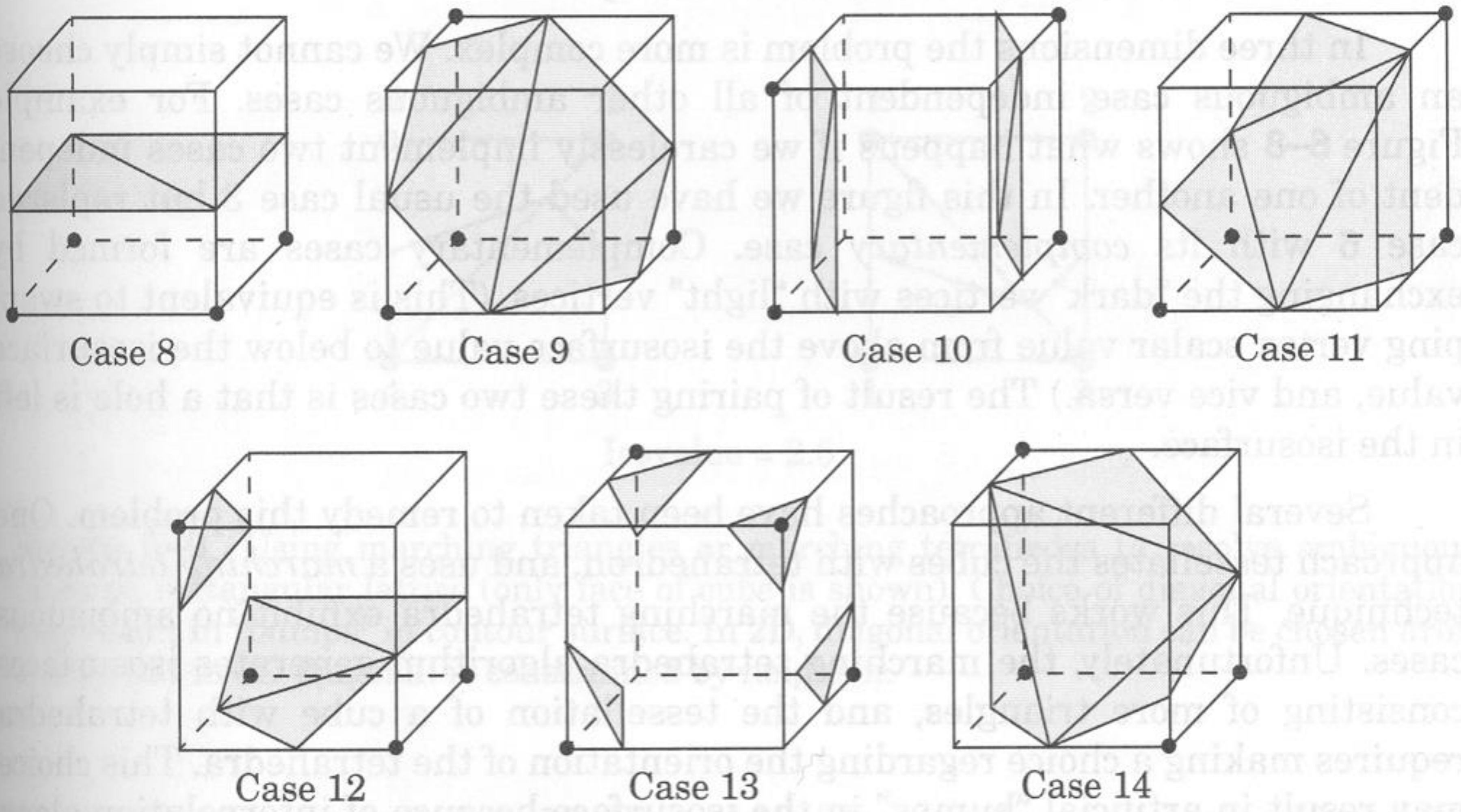
Marching Cubes



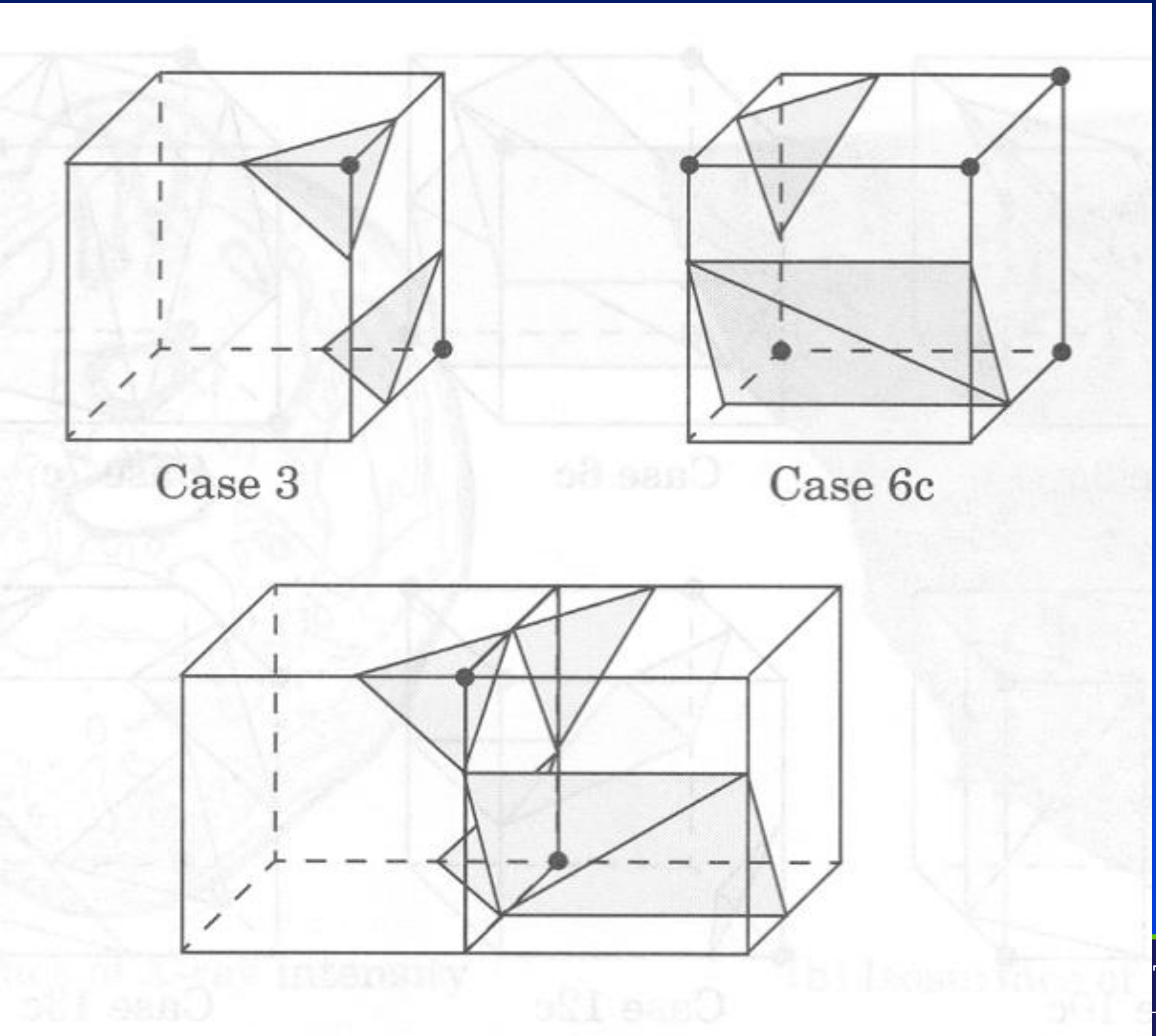
Marching Cubes Cases



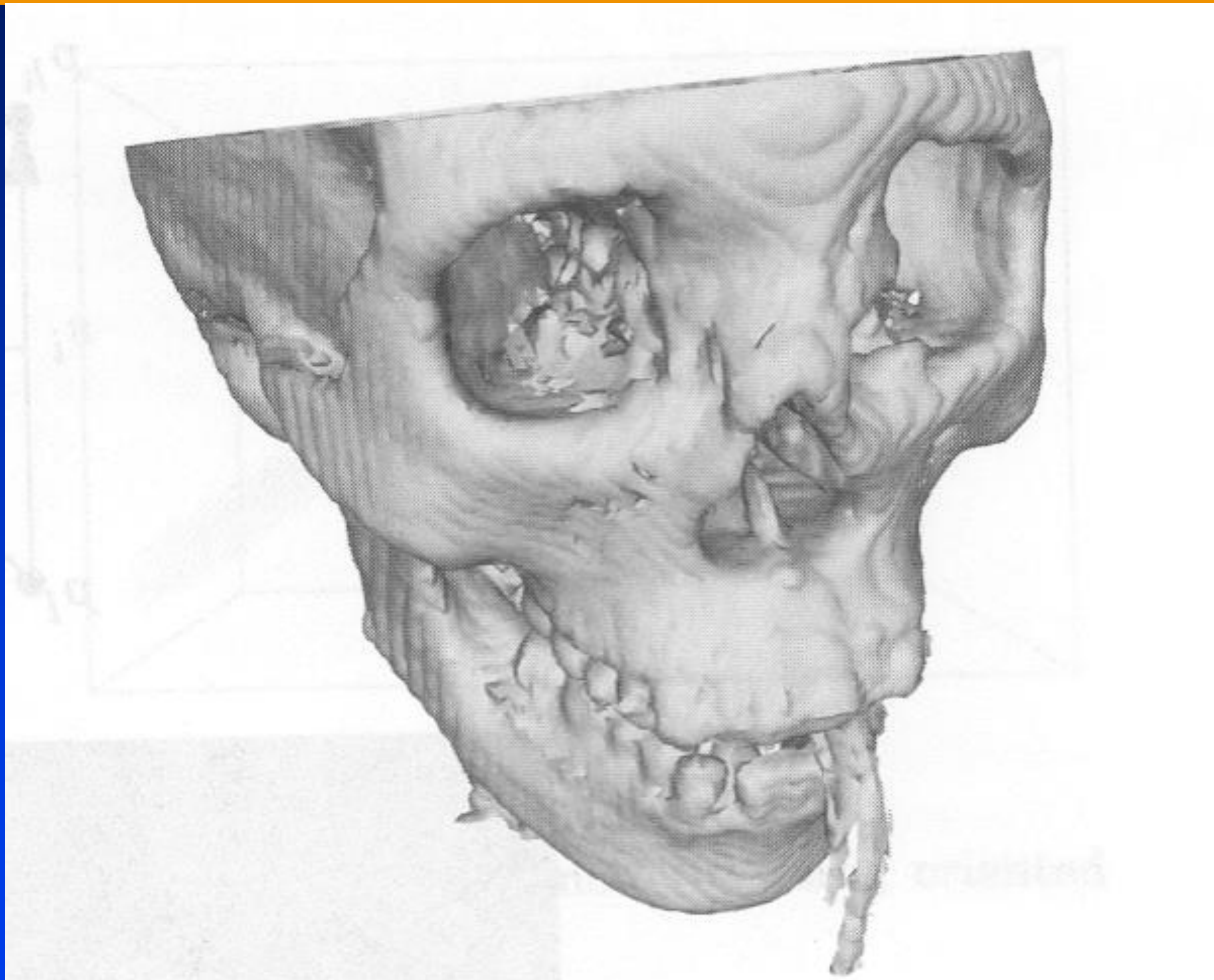
Marching Cubes Cases



Ambiguity in Connecting Edges

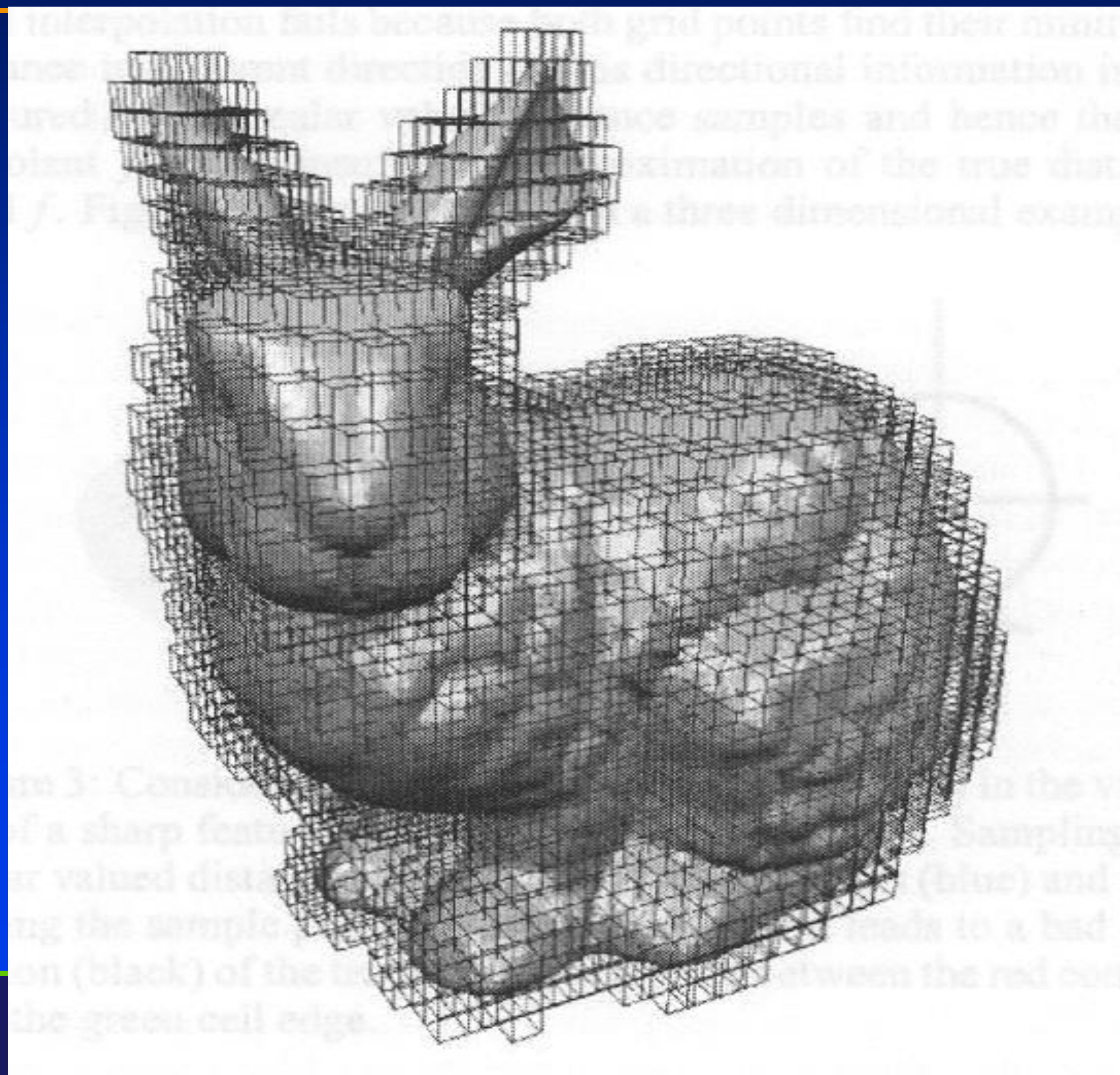


An Example of Extracted Isosurfaces



(b) Isosurface of human skull

Marching Cubes for 3-D Data Clouds



An Example of Extracted Meshes