# Introduction to Scientific Visualization

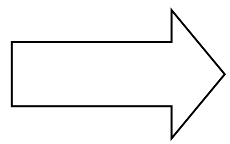
(some slides from Hong Qin)

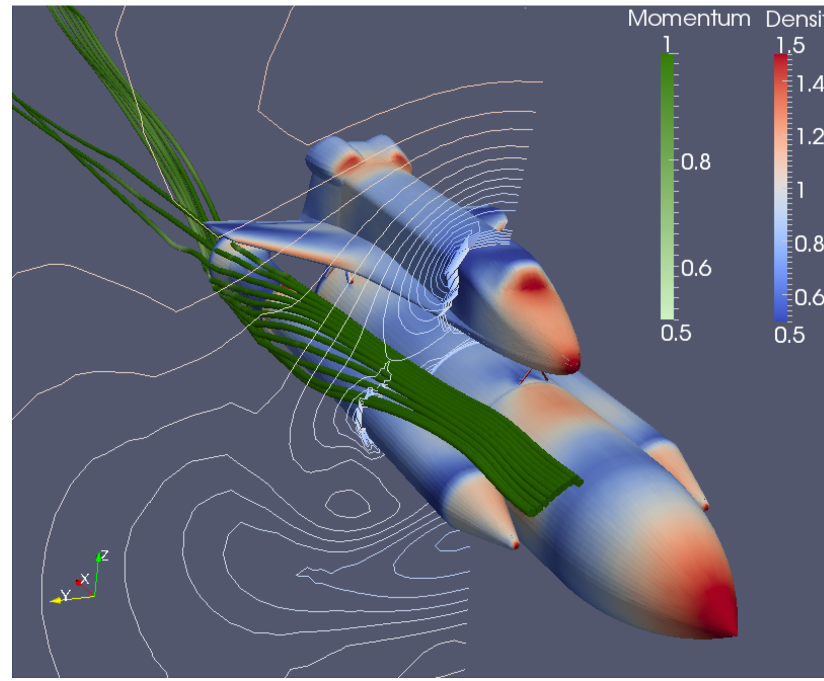
# What is Scientific Visualization

- 1987 the US National Science started "Visualization in scientific computing" as a new discipline,
  - ACM coined the term "scientific visualization"
- Scientific visualization, briefly defined:

*The use of computer graphics*
*for* ***analysis and presentation***
*of computed or measured scientific data*

0265640 132304 133732 032051 037334 024721 015013 052226 001662
0265660 025537 064663 054606 043244 074076 124153 135216 126614
0265700 144210 056426 044700 042650 165230 137037 003655 006254
0265720 134453 124327 176005 027034 107614 170774 073702 067274
0265740 072451 007735 147620 061064 157435 113057 155356 114603
0265760 107204 102316 171451 046040 120223 001774 030477 046673
0266000 171317 116055 155117 134444 167210 041405 147127 050505
0266020 004137 046472 124015 134360 173550 053517 044635 021135
0266040 070176 047705 113754 175477 105532 076515 177366 056333
0266060 041023 074017 127113 003214 037026 037640 066171 123424
0266100 067701 037406 140000 165341 072410 100032 125455 056646
0266120 006716 071402 055672 132571 105645 170073 050376 072117
0266140 024451 007424 114200 077733 024434 012546 172404 102345
0266160 040223 050170 055164 164634 047154 126525 112514 032315
0266200 016041 176055 042766 025015 176314 017234 110060 014515
0266220 117156 030746 154234 125001 151144 163706 136237 164376
0266240 137055 062276 161755 115466 005322 132567 073216 002655
0266260 171466 126161 117155 065763 016177 014460 112765 055527
0266300 003767 175367 104754 036436 172172 150750 043643 145410
0266320 072074 000007 040627 070652 173011 002151 125132 140214
0266340 060115 014356 015164 067027 120206 070242 033065 131334
0266360 170601 170106 040437 127277 124446 136631 041462 116321
0266400 020243 005602 004146 121574 124651 006634 071331 102070
0266420 157504 160307 166330 074251 024520 114433 167273 030635
0266440 133614 106171 144160 010652 007365 026416 160716 100413
0266460 026630 007210 000630 121224 076033 140764 000737 003276
0266500 114060 042647 104475 110537 066716 104754 075447 112254
0266520 030374 144251 077734 015157 002513 173526 035531 150003
0266540 146207 015135 024446 130101 072457 040764 165513 156412
0266560 166410 067251 156160 106406 136770 030516 064740 022032
0266600 142166 123707 175121 071170 076357 037233 031136 015232
0266620 075074 016744 044055 102230 110063 033350 052765 172463



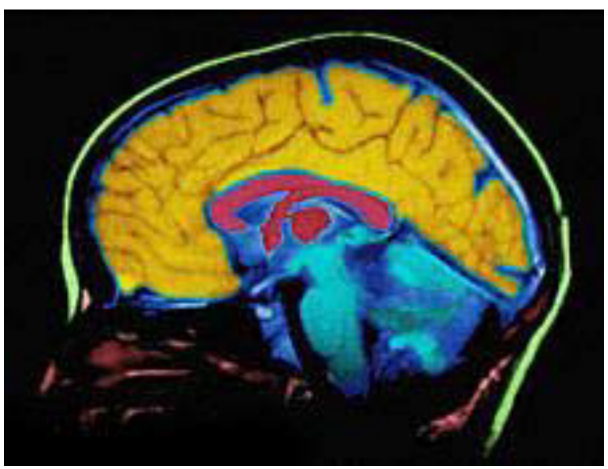
# Motivations of Visualization

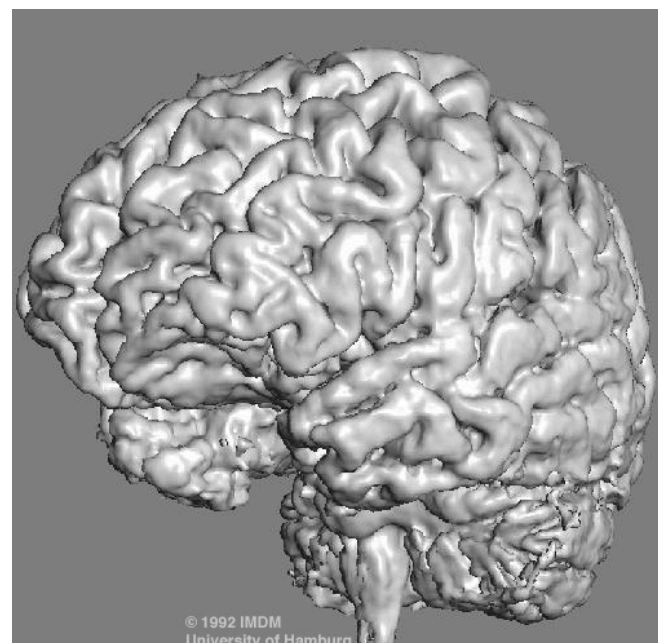
- Make sense of huge data-sets
- NYSE makes hundreds of millions of transactions per day
- The Large Hadron Collider (LHC) produces 25Gb/sec of data with each experiment
- Uncover insights hidden in the data
- Extract important features and meaningful knowledge of the data to assist in the decision-making process

# Examples: Medical

Medical imaging

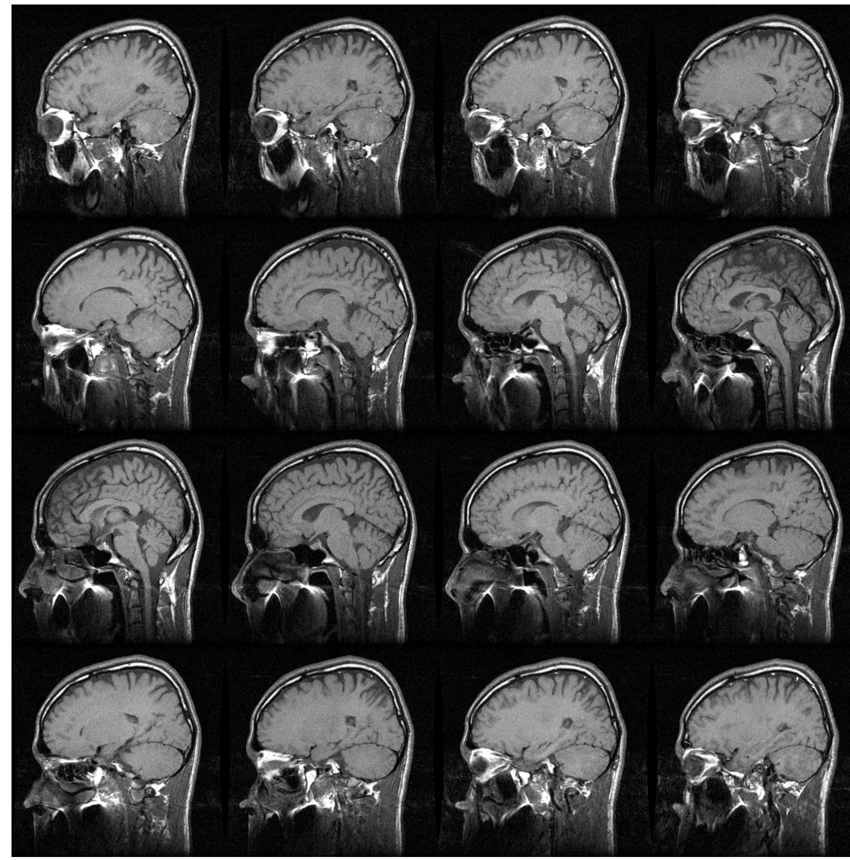
- X-ray Computed Tomography (CT)
- Magnetic Resonance Imaging (MRI)





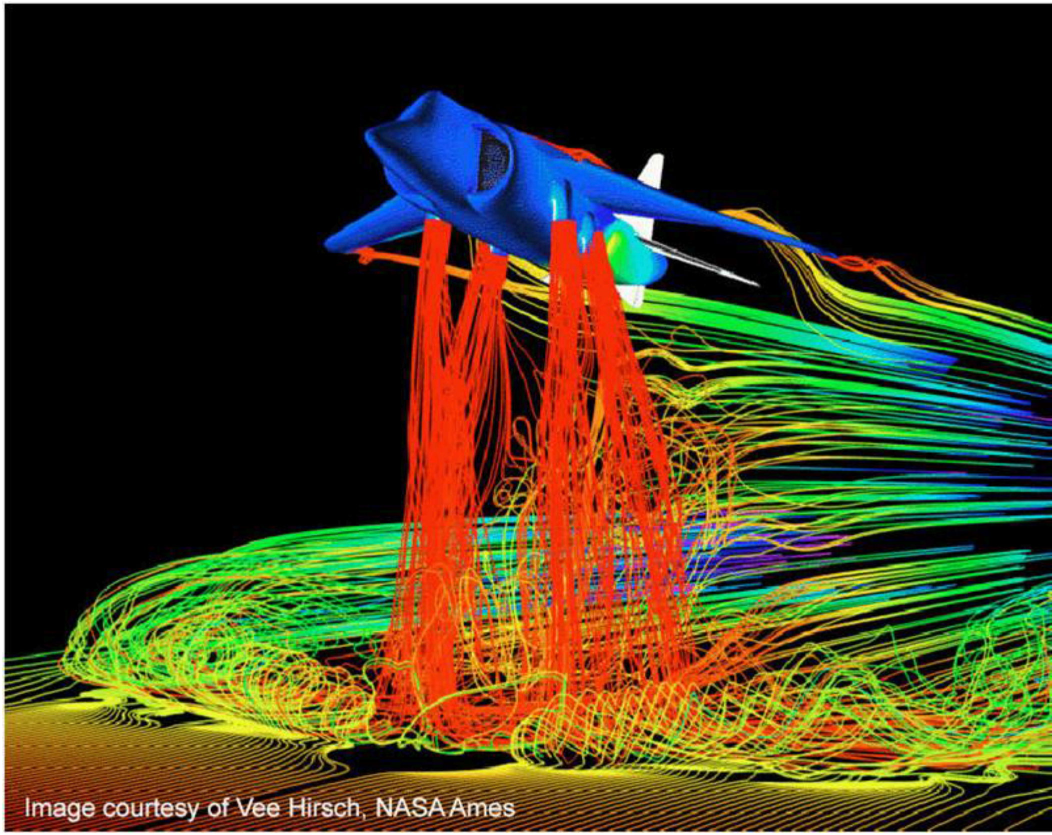
# Example Medical

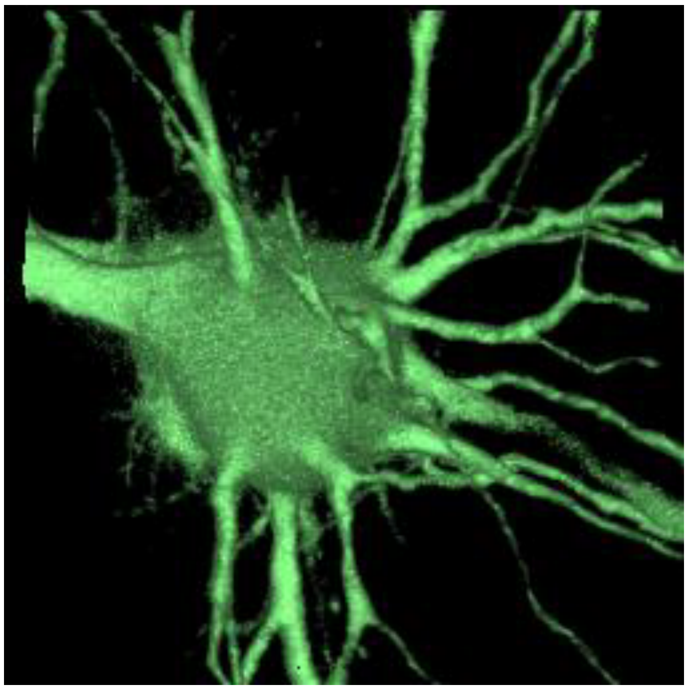
- CT and MRI generate slides
  - Cross-sections of the patient
  - Slices are combined to produce a volumetric representation
- But CT and MRI machines just output numbers – where do the gray values come from?

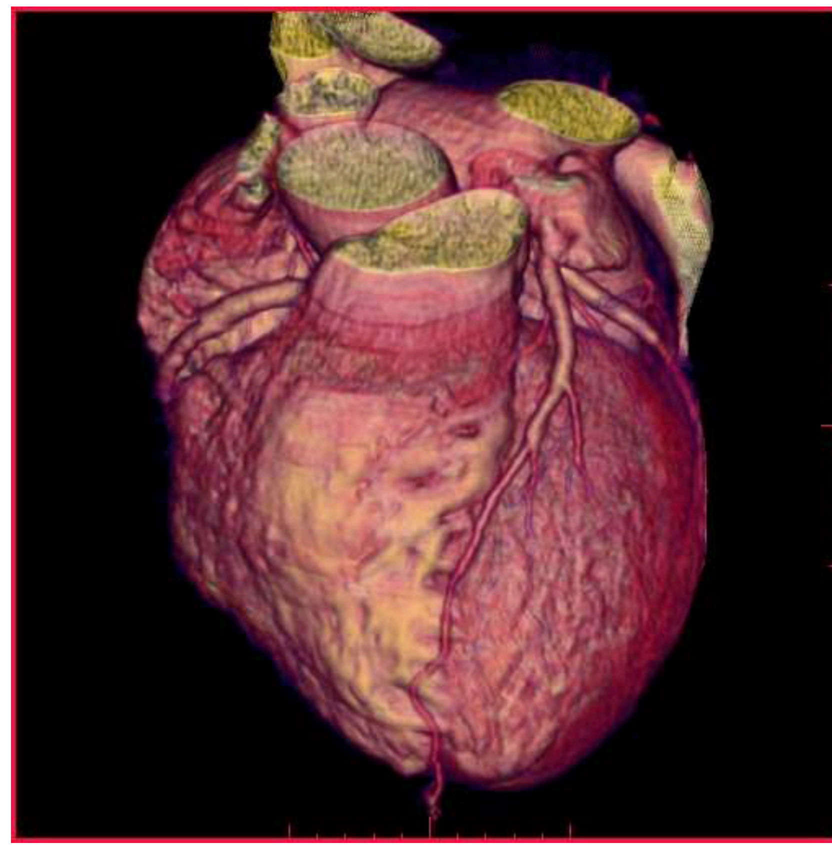


# Example Simulations

- Scientific simulations
- Visualize the results of very sophisticated super-computer simulations
- Computational fluid dynamics example:
  - What quantities are being visualized?





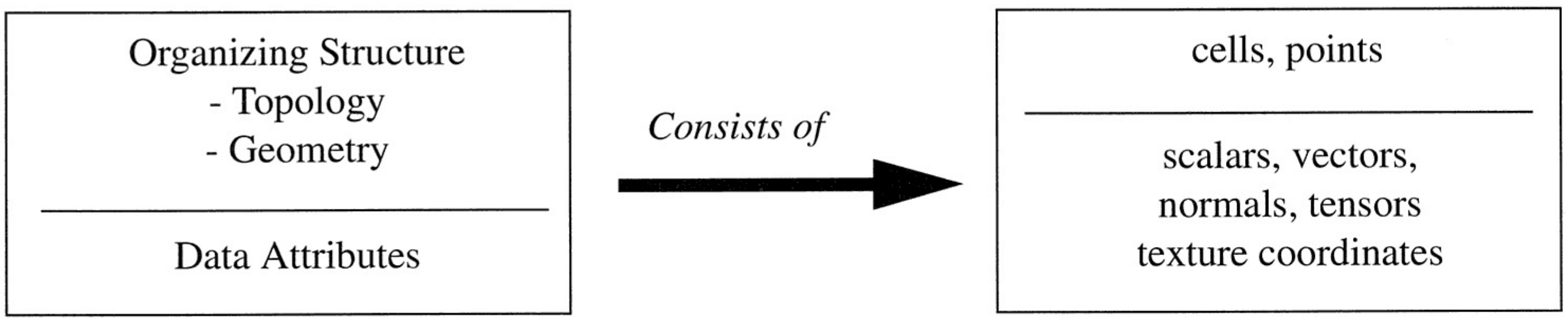


# Data Representations

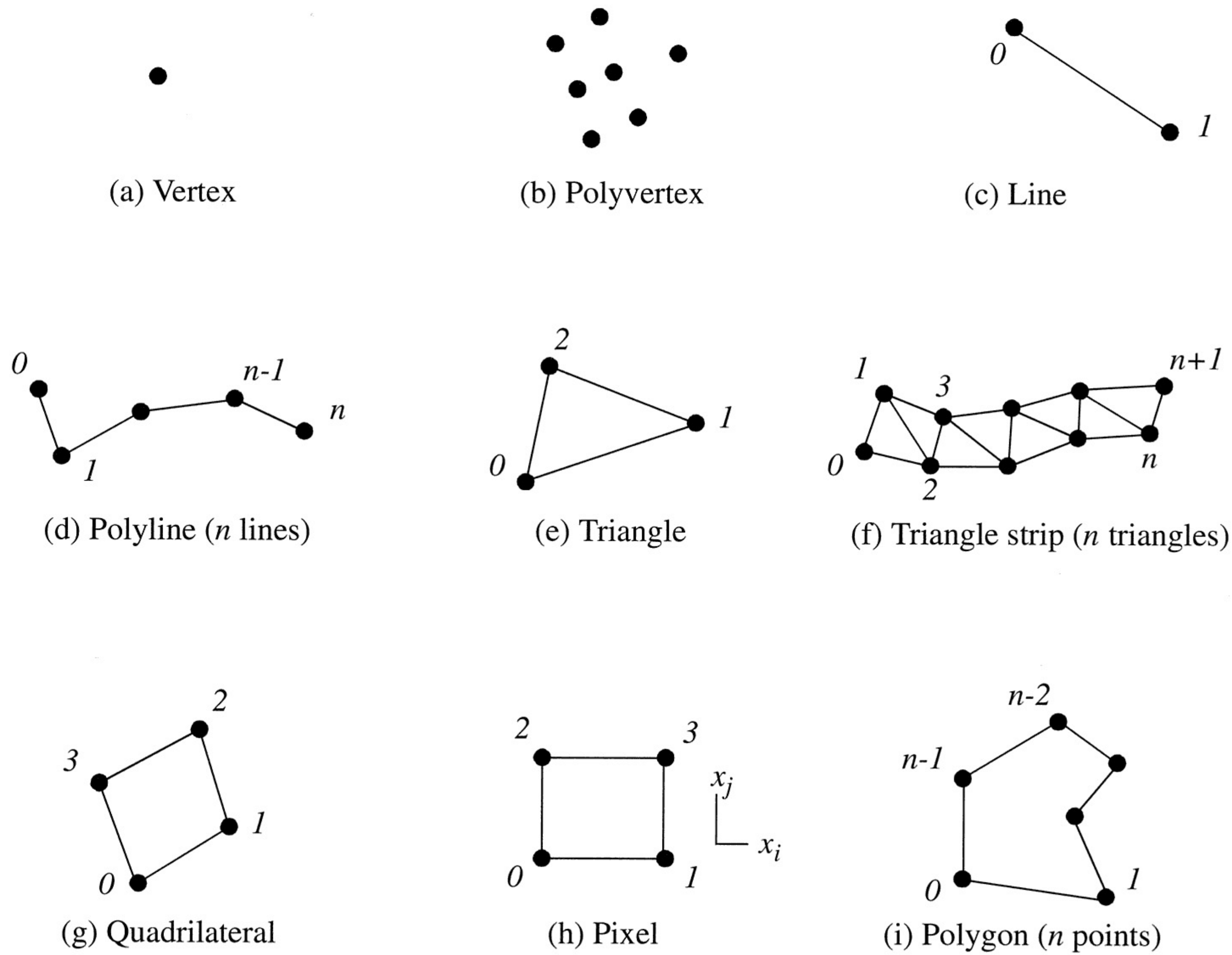
- Many ways to represent data
- Points (e.g., 3D raster, point cloud)
- Lines
- Vectors
- These are all **discrete** data representations
- Data can be **regular** or **irregular**
- Regular = relationship exists between data points
- Compare: 3D raster vs. point cloud
- Data also has **dimension**: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




- **cells** define topology, **points** define geometry
- Linear cell types and non-linear cell types

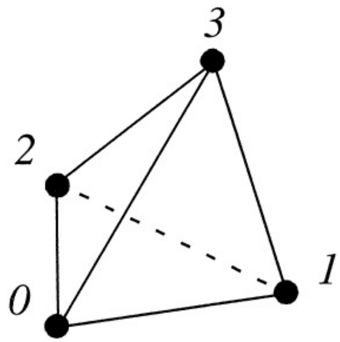


(a) Vertex (b) Polyvertex (c) Line

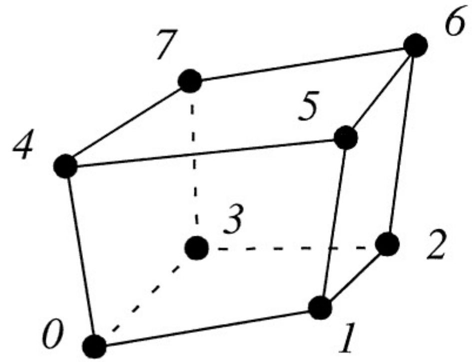
(d) Polyline (*n* lines) (e) Triangle (f) Triangle strip (*n* triangles)

(g) Quadrilateral (h) Pixel (i) Polygon (*n* points)

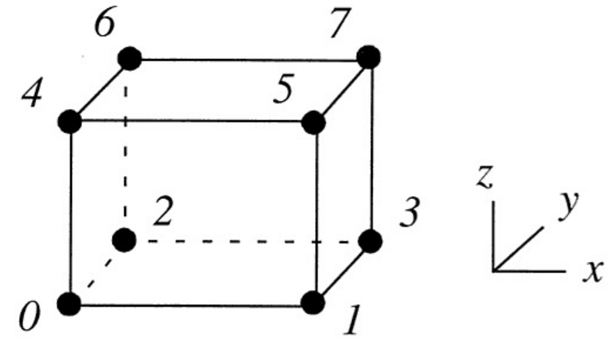
- Cell topology defined by **connectivity** of vertices



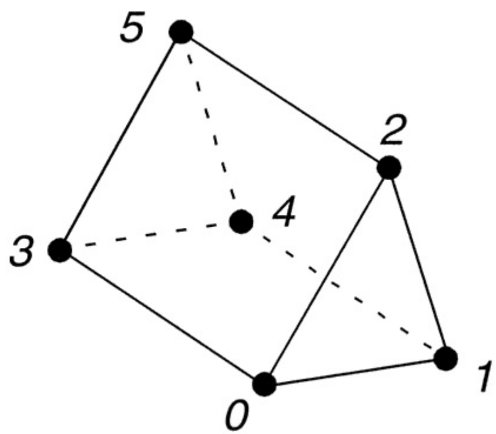

(j) Tetrahedron



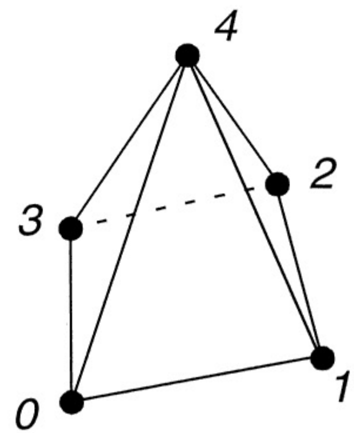

(k) Hexahedron



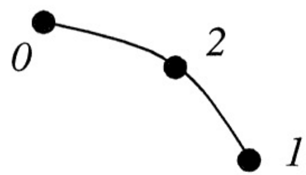

(l) Voxel



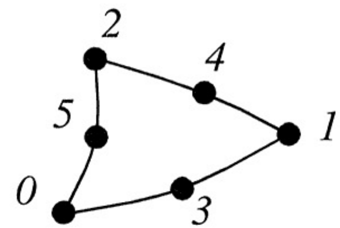

(m) Wedge



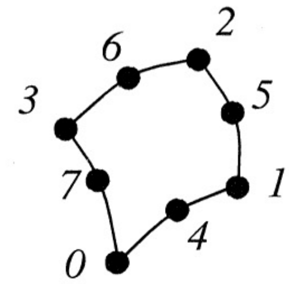

(n) Pyramid



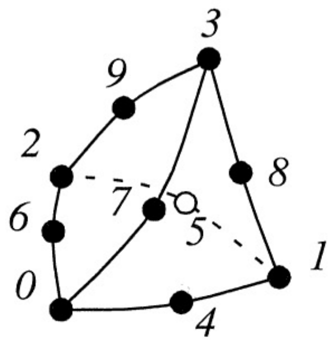

(a) Quadratic Edge



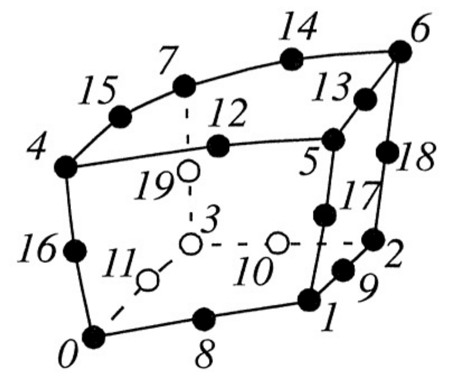

(b) Quadratic Triangle




(c) Quadratic Quadrilateral



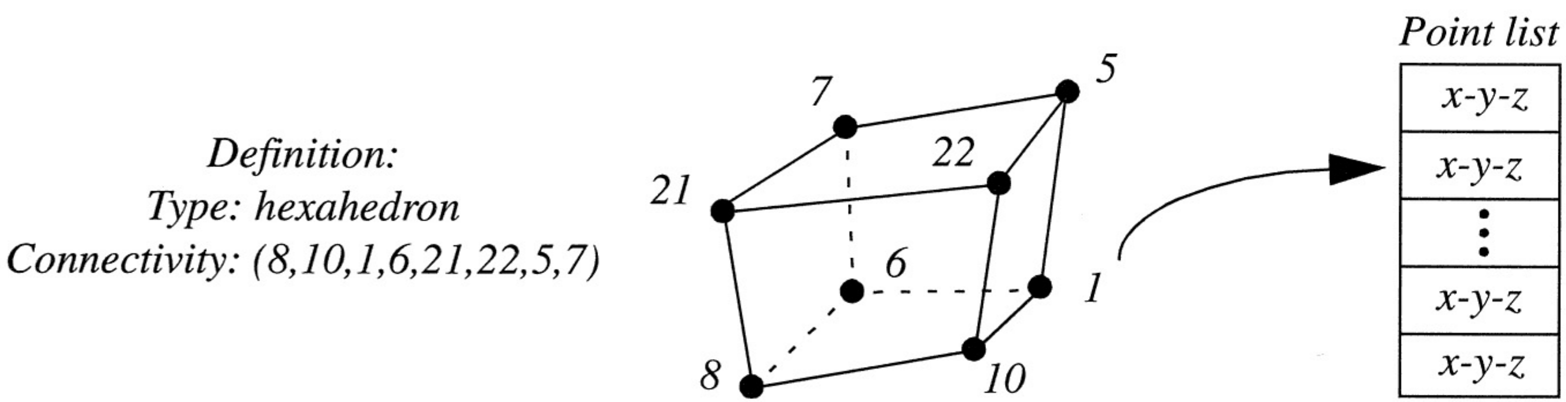

(d) Quadratic Tetrahedron




(e) Quadratic Hexahedron

# Cell Example: Hexahedron

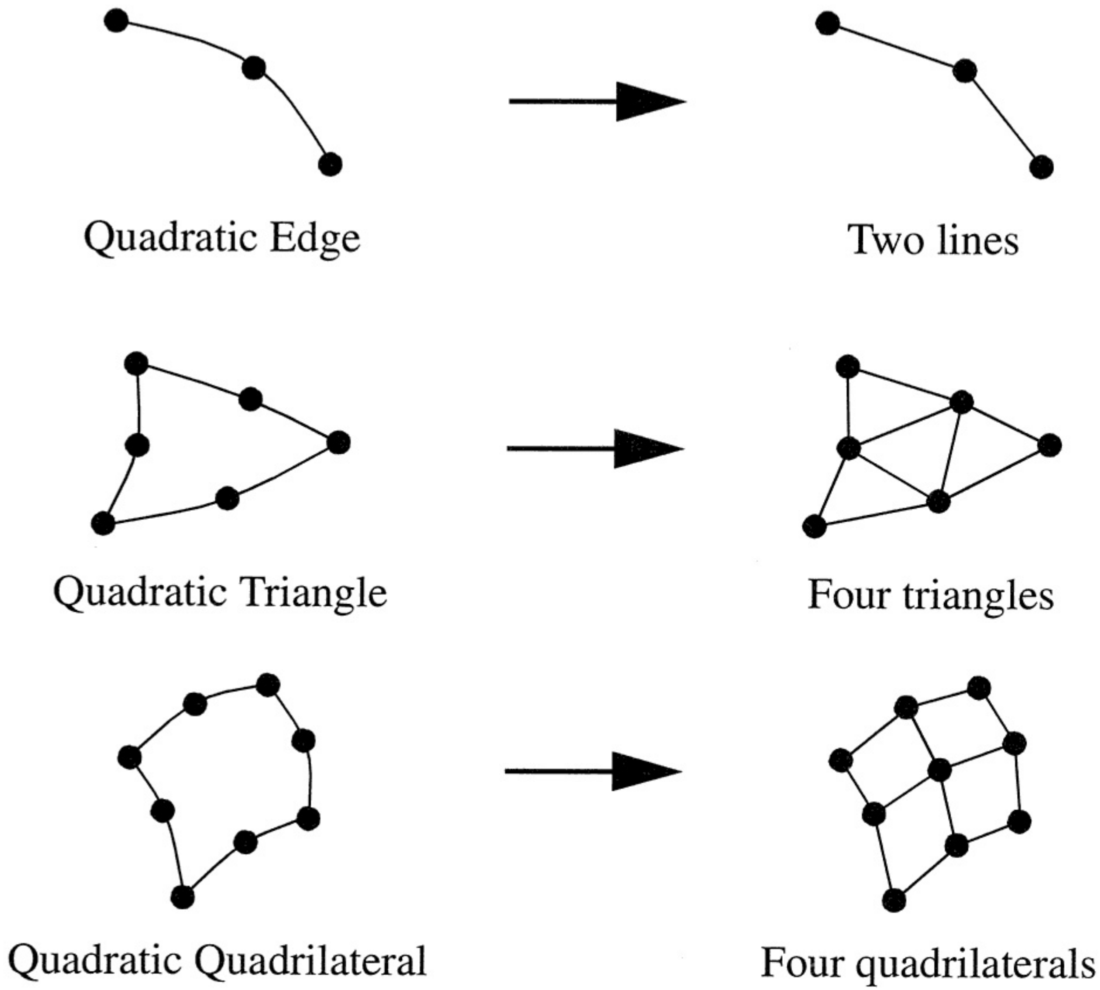
- Vertices listed in special order define topology



# 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

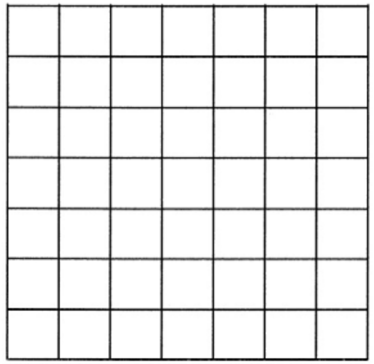
- **Scalar** data is data that is single-valued at all locations in a data-set
- Examples: temperature, stock price, elevation
- **Vector** data is data with magnitude and direction
- Examples: position, velocity, acceleration
- **Normals** (direction vectors) are vectors of magnitude 1
- **Texture coordinates** 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

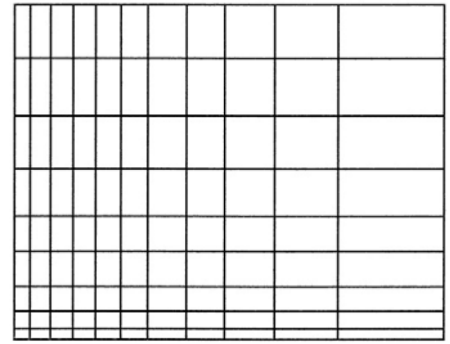
- **Tensors** are mathematical generalizations of vectors and scalars
- Usually written as matrices
- Tensor visualization is extremely difficult

# Types of Data-sets

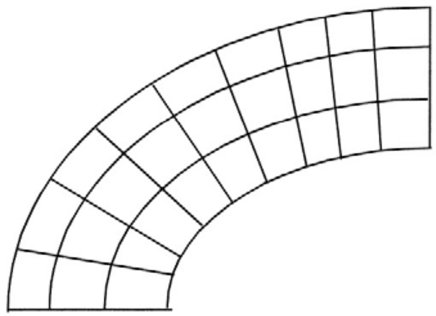
- Regular vs. irregular structure – refers to topology of data-set
- Data-sets with regular topology, we do not need to store connectivity information
- Points themselves can be regular or irregular
- If irregular, we need to store the positions
- Unstructured data must be explicitly represented
- High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



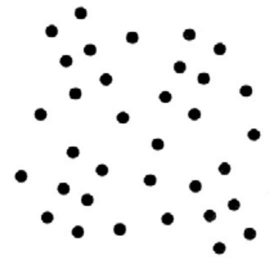
(a) Image Data



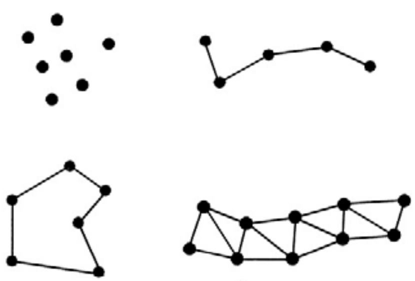
(b) Rectilinear Grid



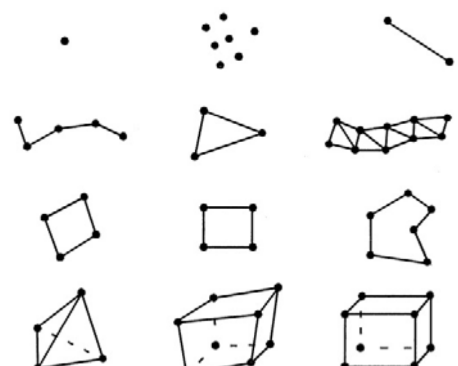
(c) Structured Grid



(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

# Fundamental Visualization Algorithms

# Visualization Algorithms

- “Algorithms that transform data are the heart of visualization”
- Algorithms classified according to **structure** and **type** of data
- **Geometric transformations** change geometry but not topology
- Examples: translation, rotation, scaling
- **Topological transformations** change topology but not geometry
- Example: convert from regular to irregular grid

# Visualization Algorithms

- **Attribute transformations** convert or create attributes in data
- Example: convert vector to scalar
- **Combined transformations** change data structure and attributes
- Algorithms that change data type include **scalar algorithms, vector algorithms, tensor algorithms,** and **modeling algorithms**
- **Volume visualization** and **vector visualization** have their own special algorithms

# Scalar Algorithms

- **Color mapping** – map scalar data to colors
- Why scalars?
- How would you map a vector to a color?
- **Color lookup table (LUT)** – attributes inside particular range are mapped to color

$$s_i < min\ ,\ i = 0$$

$$s_i > max\ ,\ i = n - 1$$

$$i = n\left(\frac{s_i - min}{max - min}\right)$$

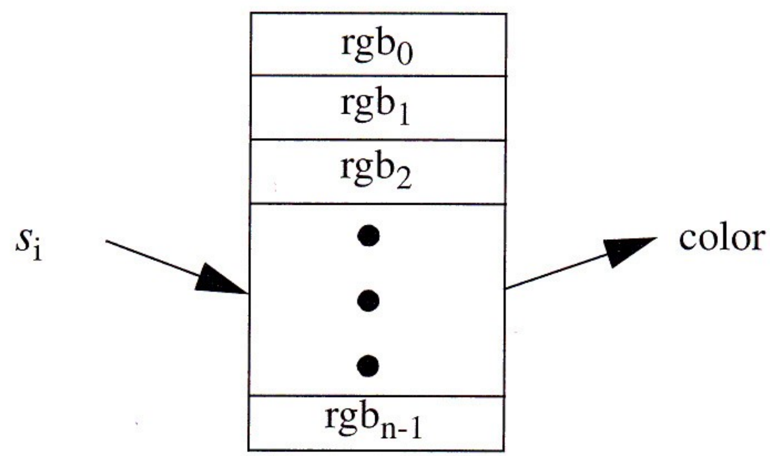



**Figure 6–1** Mapping scalars to colors via a lookup table.

# Transfer Functions

- More general form of lookup table
- Can map data to color as well as transparency
- Usually expressed as actual functions

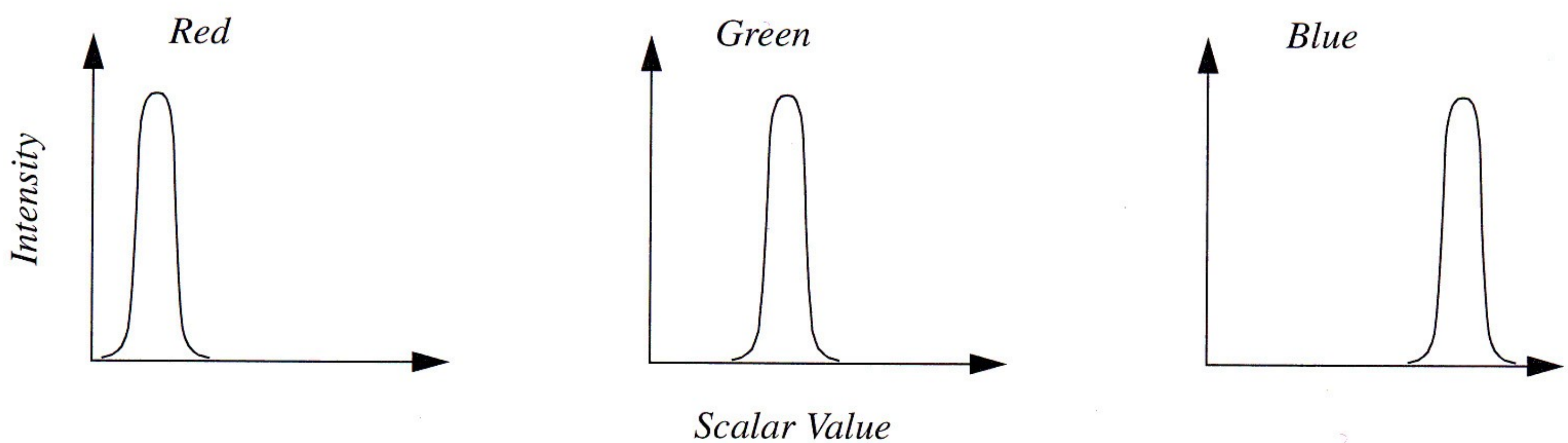
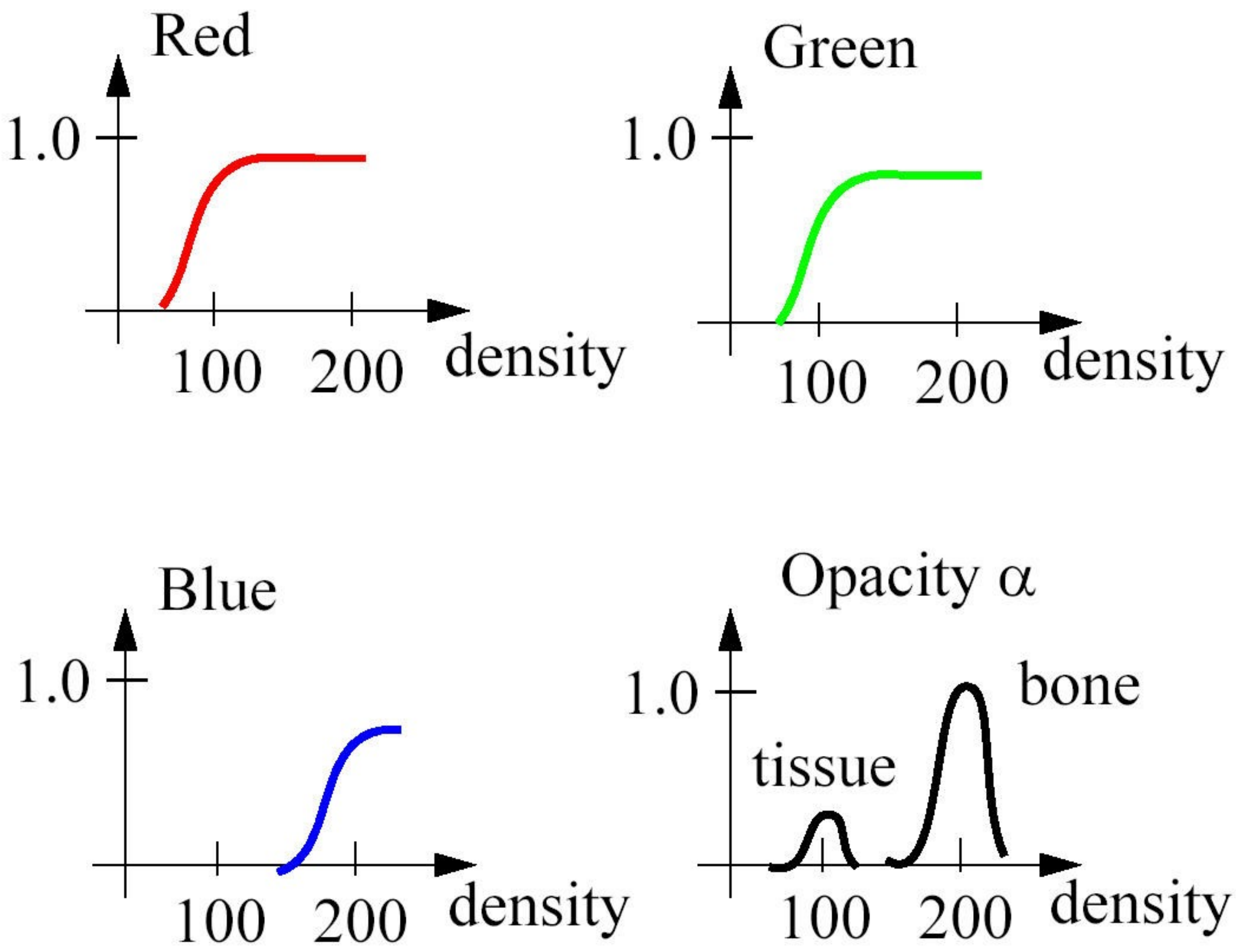



**Figure 6–2** Transfer function for color components red, green, and blue as a function of scalar value.

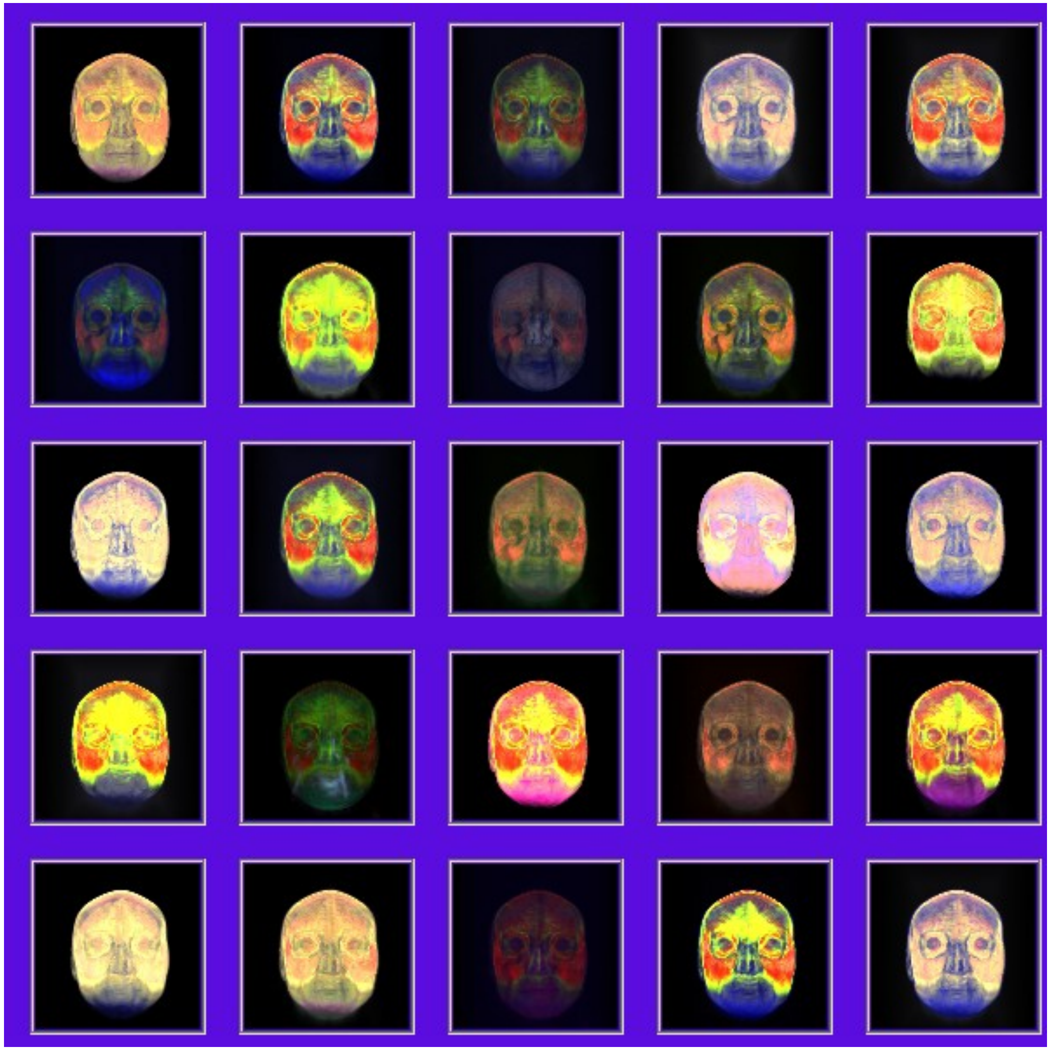
# Transfer Functions




# Transfer Functions

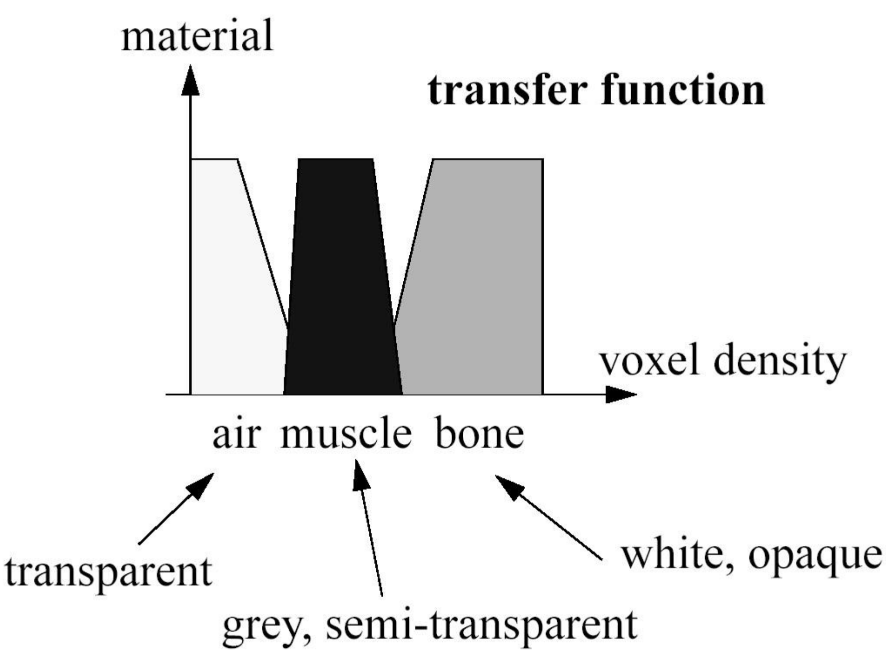
- Difficult to design
- Semi-automatic systems exist: **transfer function design galleries**
- Idea: generate random transfer functions, user selects ones he likes, system *mutates* them using a genetic algorithm to create new ones

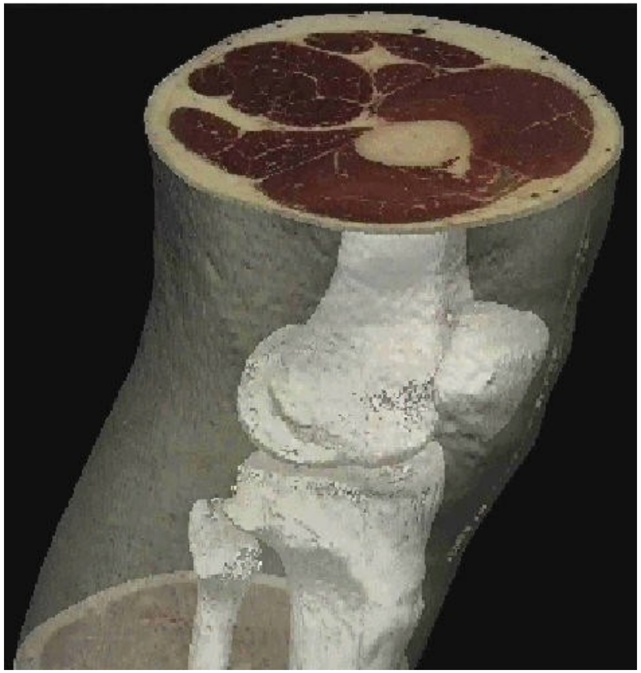
# Transfer Function Design Galleries



# Transfer Functions

- The assignment of color and transparency to density is also called **classification**



# Transfer Functions

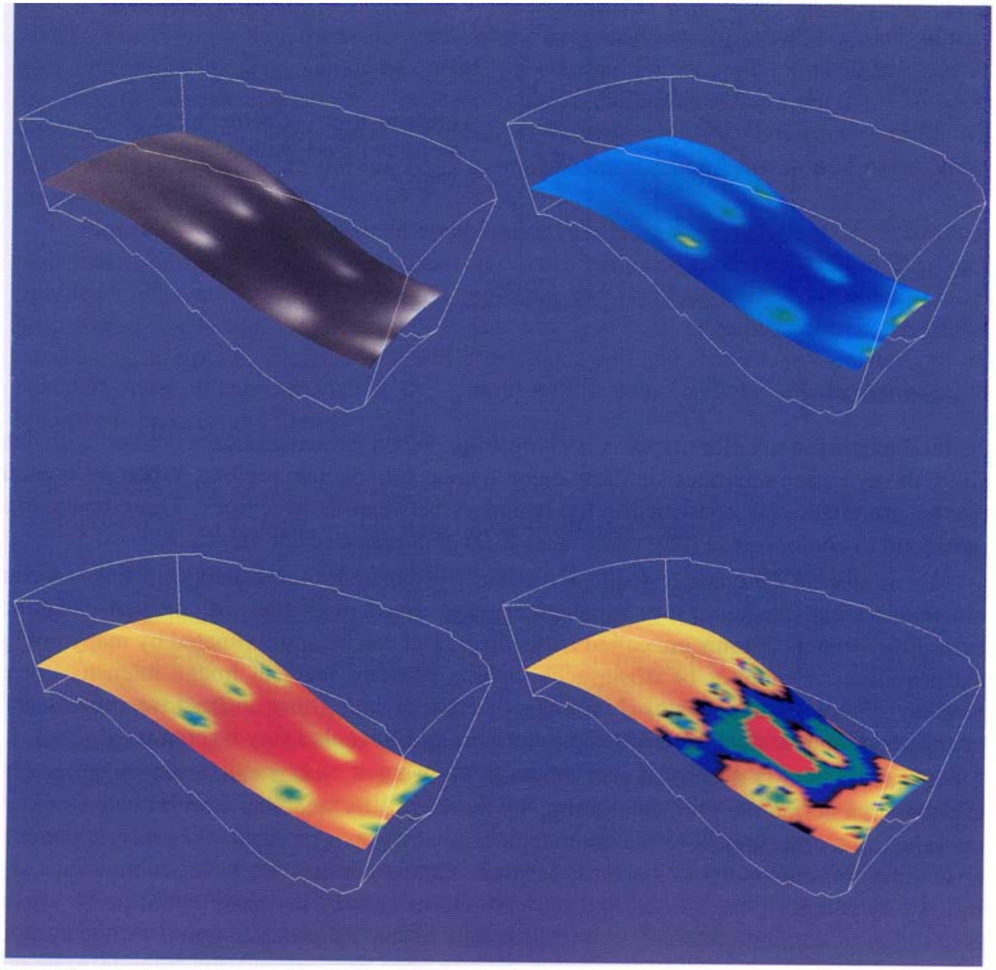


**Figure 6–3** Flow density colored with different lookup tables. Top-left: grayscale; Top-right rainbow (blue to red); lower-left rainbow (red to blue); lower-right large contrast (`rainbow.tcl`).

# Contouring

- **Isocontour** and **isosurface extraction** can reveal structure of data (e.g., isobars on weather maps)
- Separate data into regions
- Isocontours: connected line segments
- Isosurfaces: triangular meshes

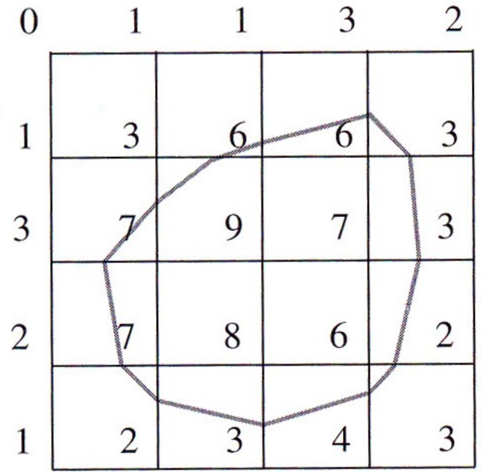



**Figure 6–4** Contouring a 2D structured grid with contour line value = 5.

# Contouring

- Isolines cross cell boundaries
- Use **interpolation** to compute crossing point
- **Marching squares** algorithm processes each quadrilateral cell independently
- Each vertex may be inside or outside (or on) contour
- How many cases must we consider?
- Ambiguous cases

# Marching Squares Cases

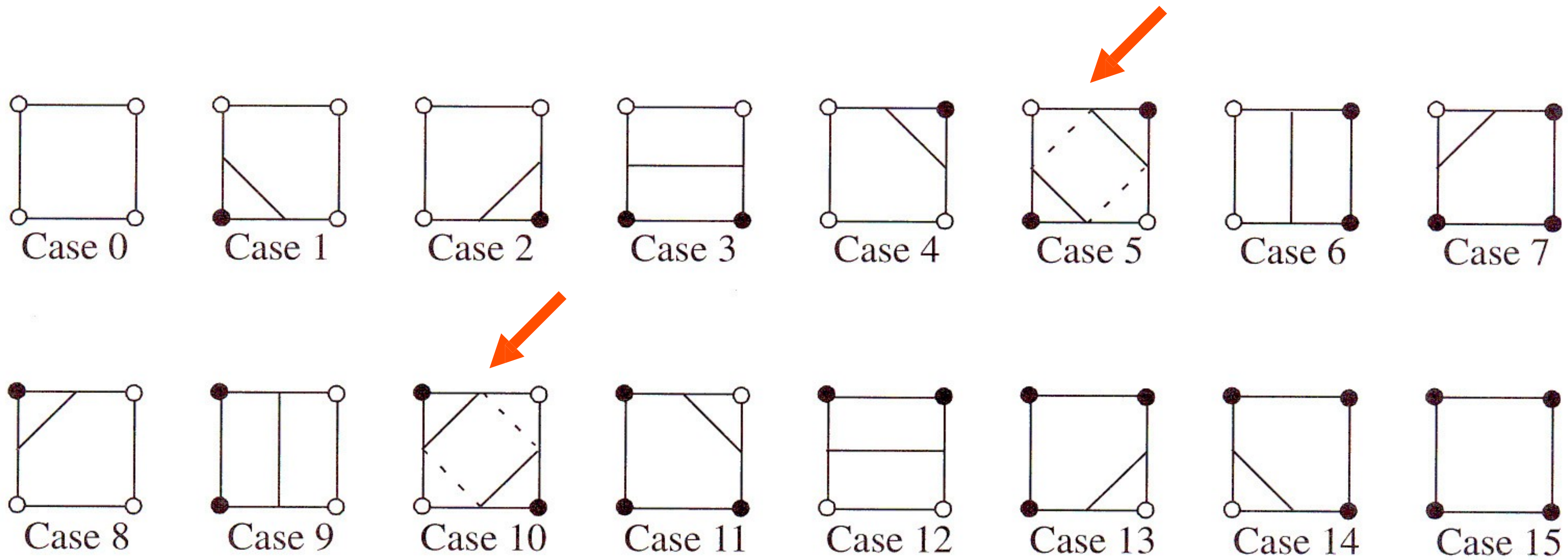



**Figure 6–5** Sixteen different marching squares cases. Dark vertices indicate scalar value is above contour value. Cases 5 and 10 are ambiguous.

# Marching Squares Ambiguous Case

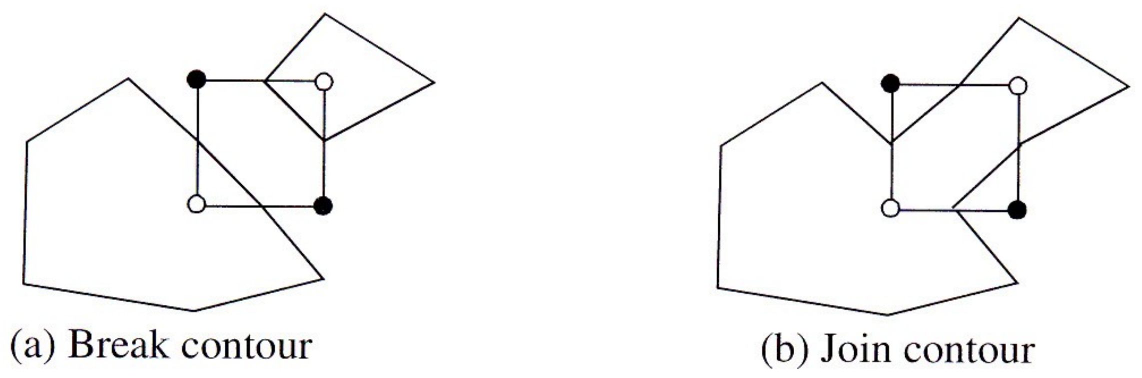


Figure 6–8 Choosing a particular contour case will break (a) or join (b) the current contour. Case shown is marching squares case 10.

# Marching Cubes

- **Marching cubes** algorithm extracts isosurfaces from 3D rasters
- Very famous algorithm
- How many cases of hexahedral cells must we consider?
- Each of 8 vertices may be inside or outside
- $2^8 = 256$
- Lots of symmetry ⌨ really only 15 cases to consider

# Marching Cubes Cases

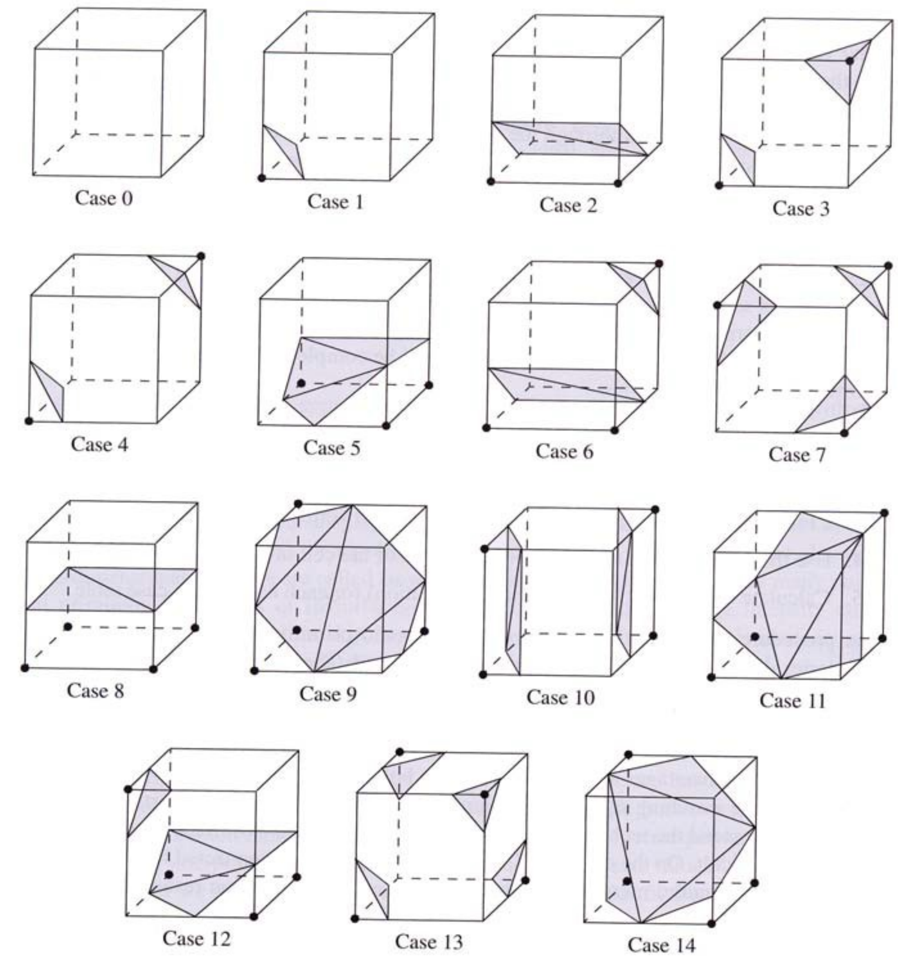



**Figure 6–6** Marching cubes cases for 3D isosurface generation. The 256 possible cases have been reduced to 15 cases using symmetry. Dark vertices are greater than the selected isosurface value.

# Marching Cubes Ambiguous Cases

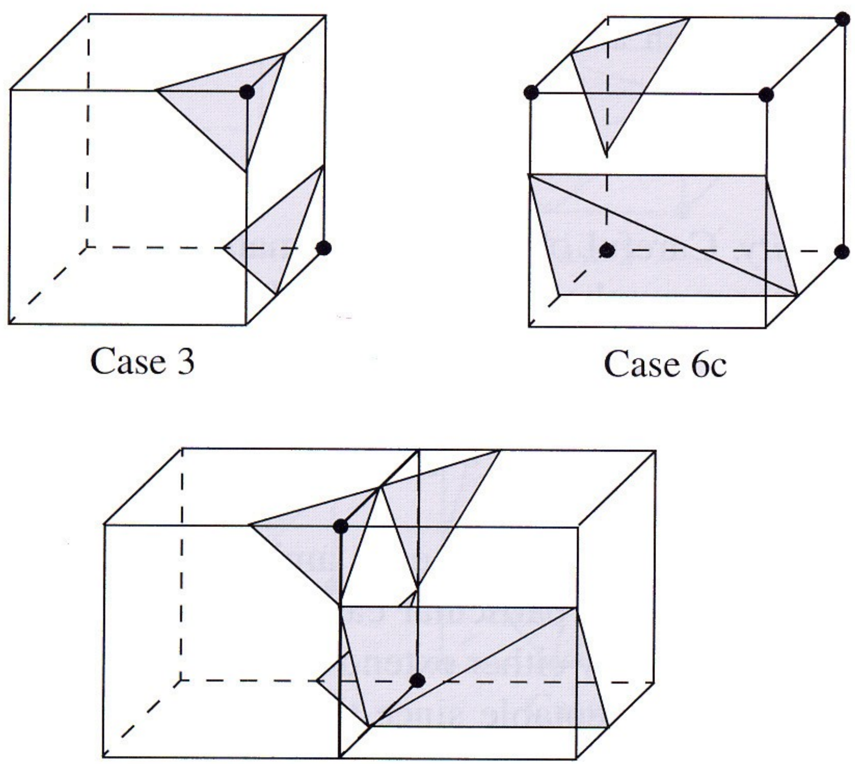



**Figure 6–9** Arbitrarily choosing marching cubes cases leads to holes in the isosurface.

# Marching Cubes Complementary Cases Used to Avoid Holes

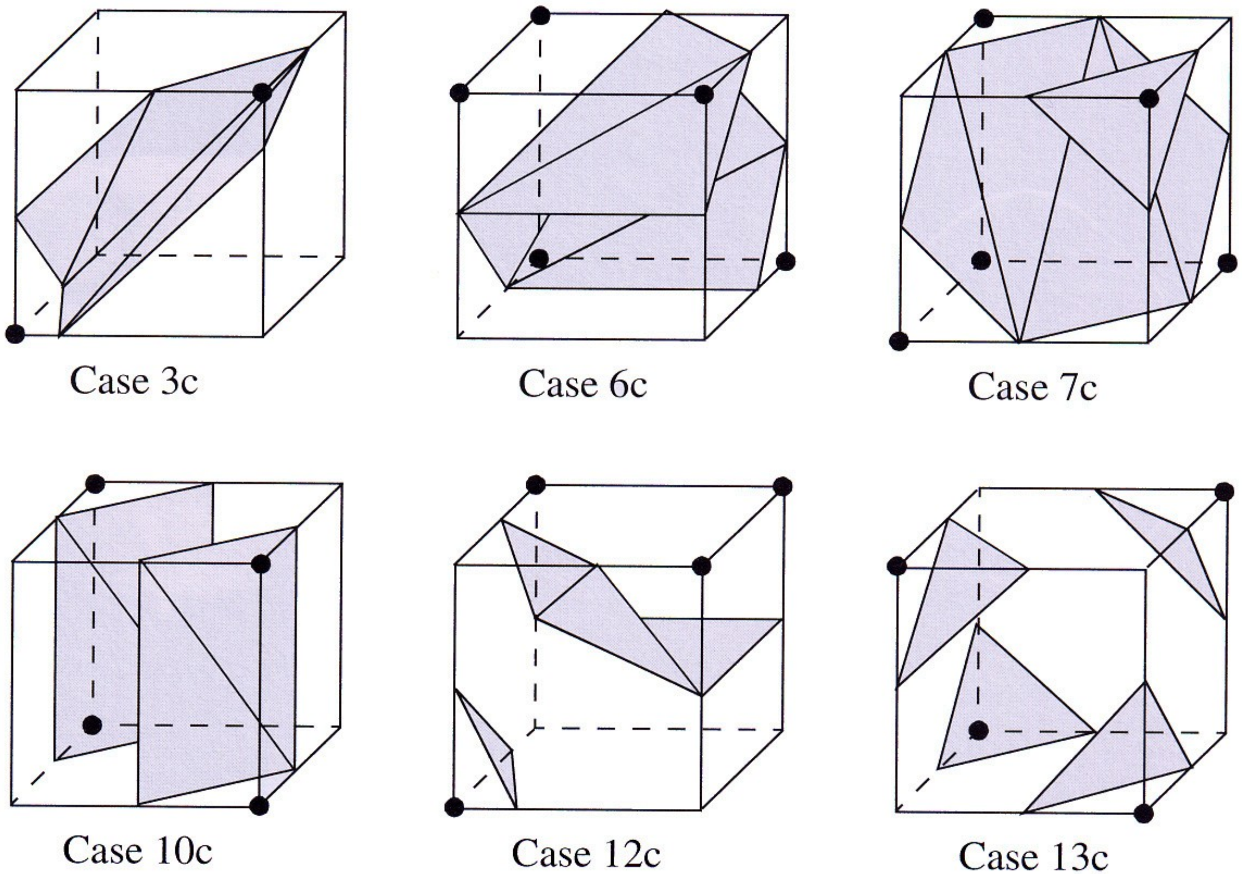



**Figure 6–10** Marching cubes complementary cases.

# Marching Triangles & Tetrahedra

- Can extend/simplify marching squares to *marching triangles*, and marching cubes to *marching tetrahedra*
- Divide squares into triangles, cubes into tetrahedra (how?) and then run different algorithms
- Tradeoff for both algorithms: simplicity vs. memory usage

# Contouring Examples

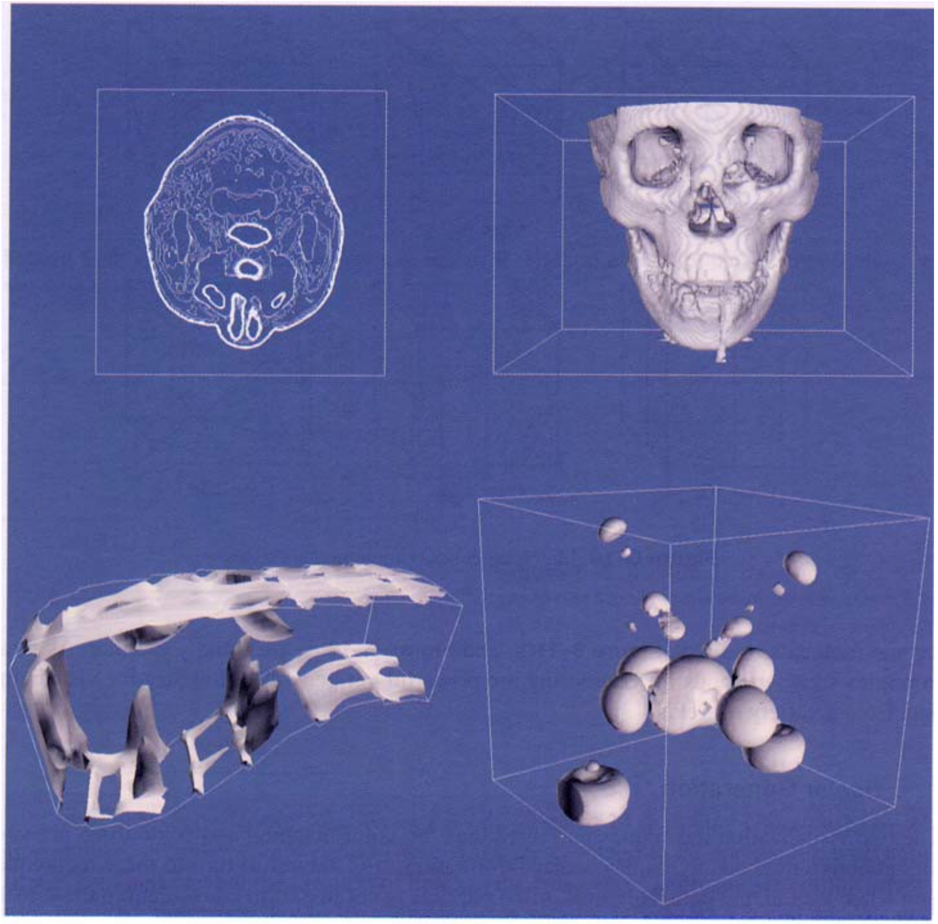


**Figure 6–11** Contouring examples. (a) Marching squares used to generate contour lines (headSlic.tcl); (b) Marching cubes surface of human bone (headBone.tcl); (c) Marching cubes surface of flow density (combIso.tcl); (d) Marching cubes surface of iron-protein (ironPIso.tcl).

# Scalar Generation

- Vectors and other n-D quantities can be turned into scalars
- Example: taking magnitude of vector
- Example: Hawaii terrain visualization created by projecting vector onto vertical
- Normalize vectors to give maximum magnitude of 1.0
- Steepest slope mapped to brightest color

# Scalar Generation

$$s_i = \frac{(p_i - p_l) \cdot (p_h - p_l)}{\left|p_h - p_l\right|^2}$$

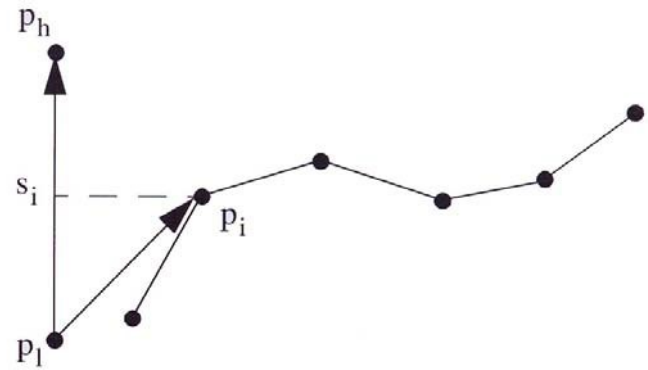
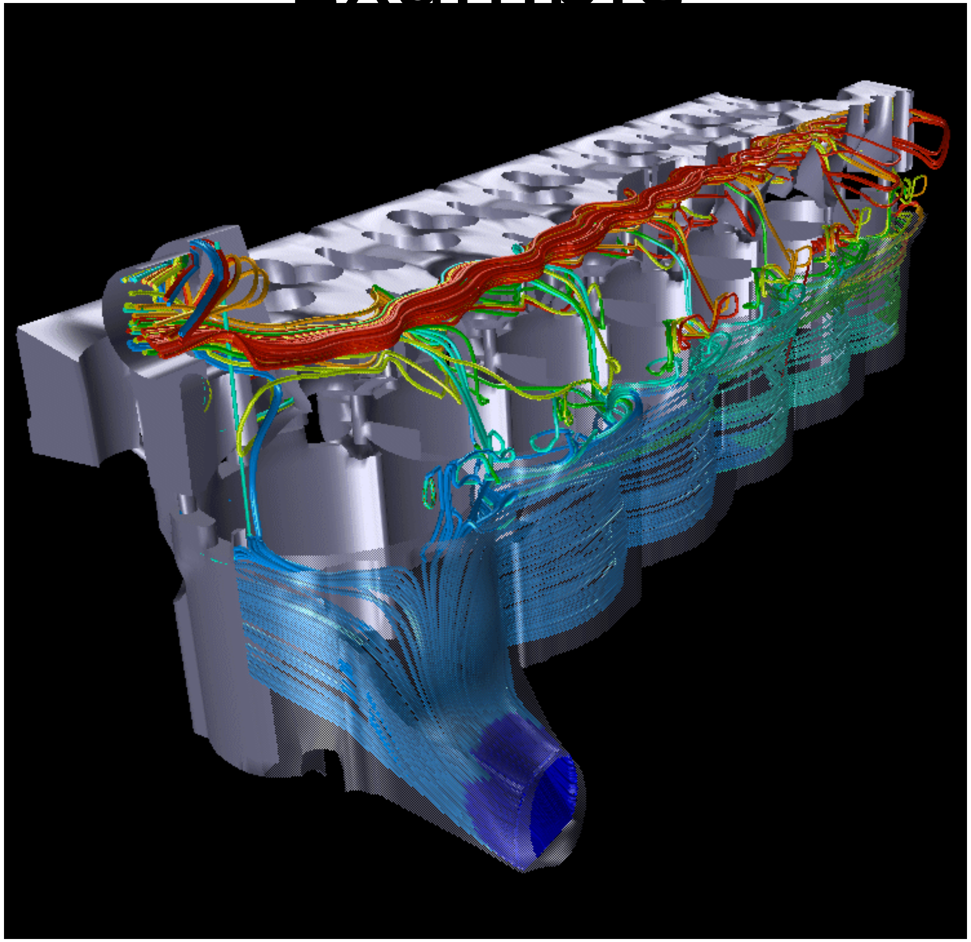



**Figure 6–12** Computing scalars using normalized dot product. Bottom half of figure illustrates technique applied to terrain data from Honolulu, Hawaii (`hawaii.tcl`).

# Vector Field Visualization

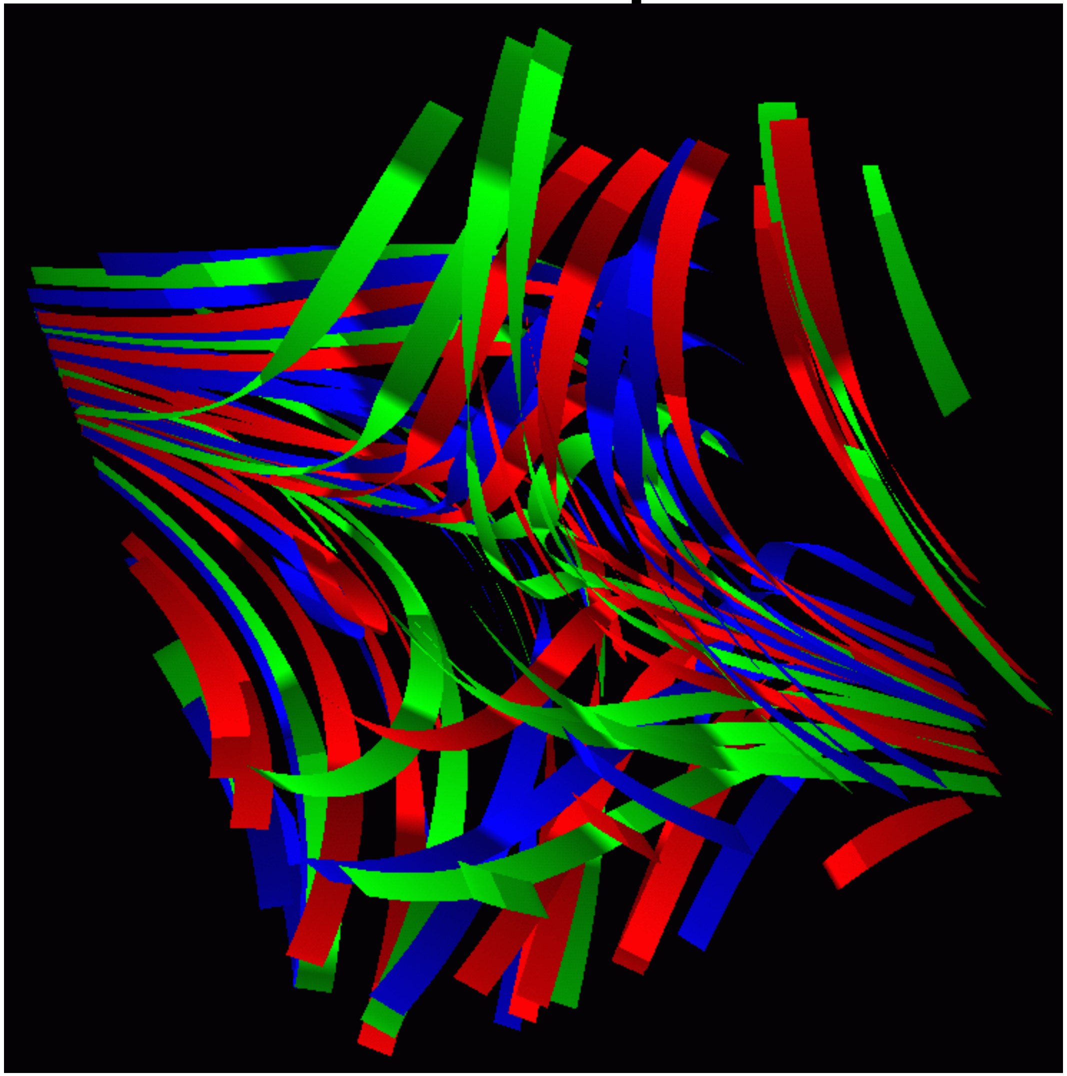
- Streamlines
  - Integration through vector field
- Stream ribbons
  - Connect two streamlines
- Streamtubes
  - Connect three or more streamlines
- Stream surfaces
  - Sweep line segment through vector field

# Streamlines Example

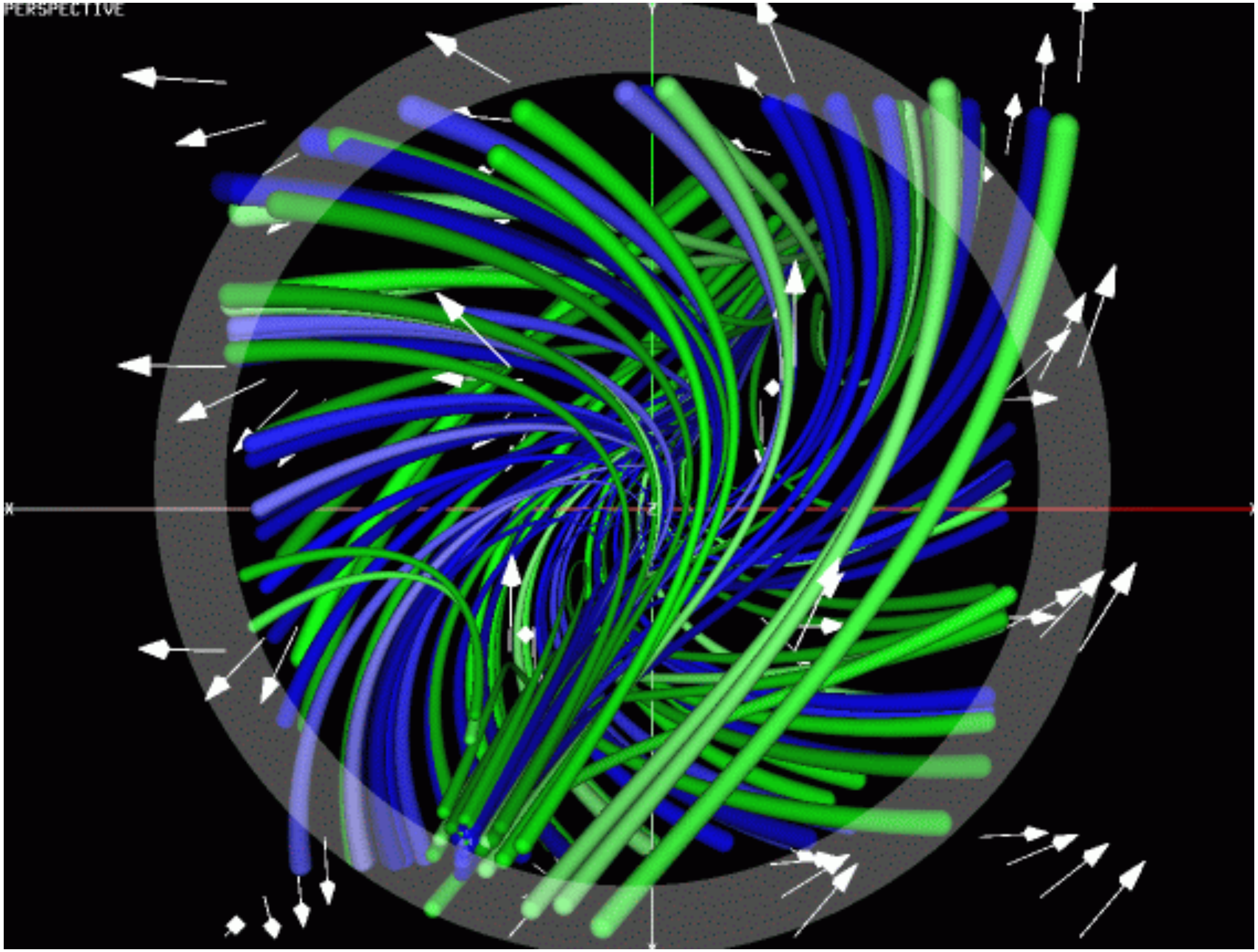


Color indicates temperature of air flowing through engine

# Streamribbons Example



# Streamtubes Example



# Streamesurfaces Example