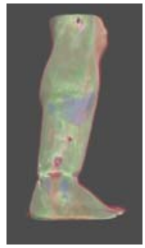
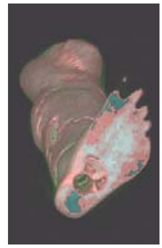
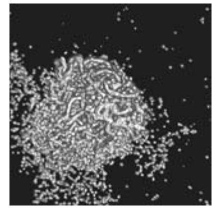
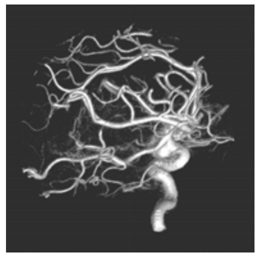
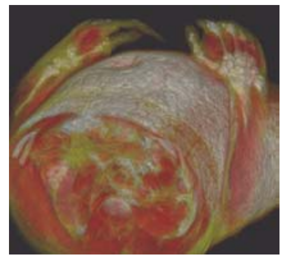
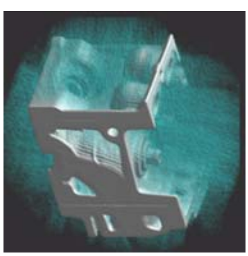
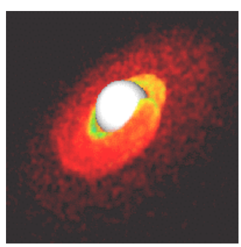
5. Volume Visualization

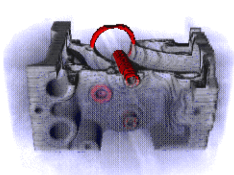
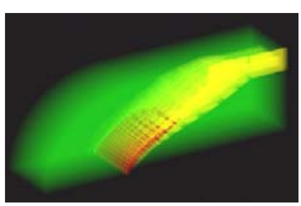
- Scalar volume data
- Medical Applications:
  CT, MRI, confocal micros-
  copy, ultrasound, etc.

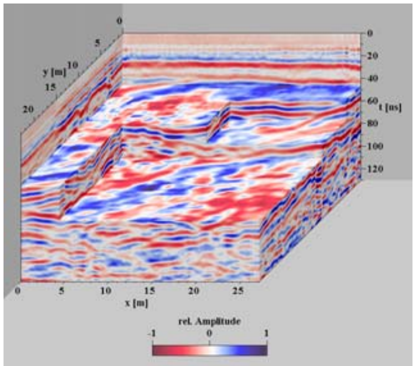



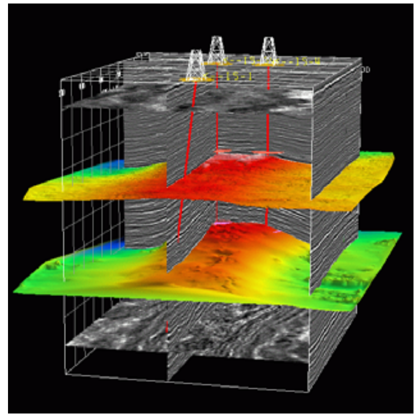
5. Volume Visualization

5. Volume Visualization





5. Volume Visualization

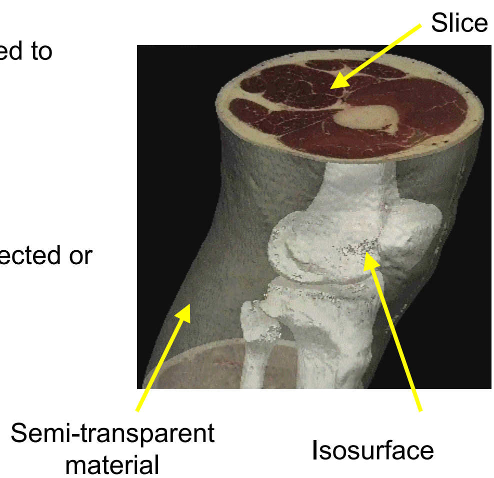
- Some possible characteristics of volume data
  - Essential information in the interior
  - Can not be described by geometric representation (fire, clouds, gaseous phenomena)
  - Distinguish between shape (given by the geometry of the grid) and appearance (given by the scalar values)
  - Even if the data could be described geometrically, there are, in general, too many primitives to be represented

5. Volume Visualization

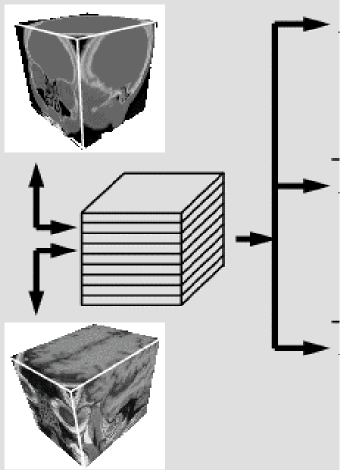
- Volume rendering techniques
  - Techniques for 2D scalar fields
  - Indirect volume rendering techniques (e.g. surface fitting)
    - Convert/reduce volume data to an intermediate representation (surface representation), which can be rendered with traditional techniques
  - Direct volume rendering
    - Consider the data as a semi-transparent gel with physical properties and directly get a 3D representation of it

5. Volume Visualization

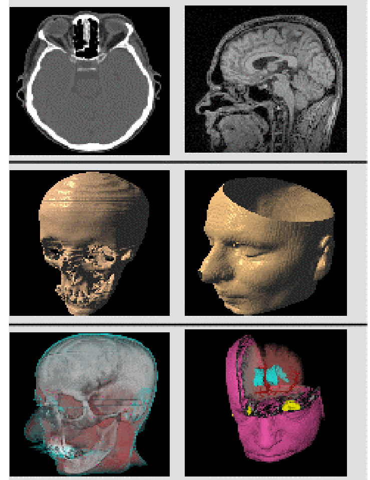
- Slicing:
  Display the volume data, mapped to colors, on a slice plane
- Isosurfacing:
  Generate opaque/semi-opaque surfaces
- Transparency effects:
  Volume material attenuates reflected or emitted light



5. Volume Visualization



- 2D visualization slice images (MPR)
- *Indirect* 3D visualization isosurfaces (SSD)
- *Direct* 3D visualization volume rendering (DVR)

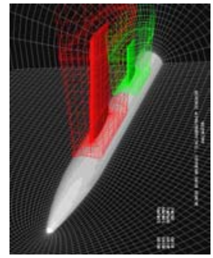


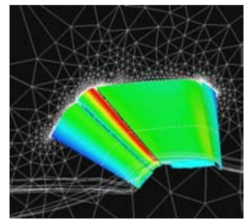
5. Volume Visualization

- Direct volume rendering techniques
  - Direct volume rendering allows for the "global" representation integrating physical characteristics
  - But prohibits interactive display due to its numerical complexity, in general
- Indirect volume rendering techniques
  - Often result in complex representations
  - Pre-processing the surface representation might help
  - Use graphics hardware for interactive display
- Goal
  - Integrate different techniques in order to represent the data as "good" as possible
  - But, keep in mind that the most correct method in terms of physical realism must not be the most optimal one in terms of understanding the data

5. Volume Visualization

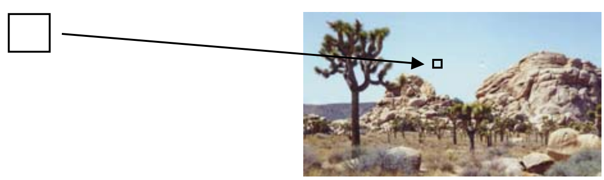
- Different grid structures:
  - Structured: uniform, rectilinear, curvilinear
  - Unstructured
  - Scattered data



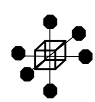


5. Volume Visualization

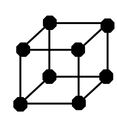
- Pixel (picture element)



- Voxel (volume element)
  - Values are constant within a region around a grid point



- Cell
  - Values between grid points are resampled by interpolation

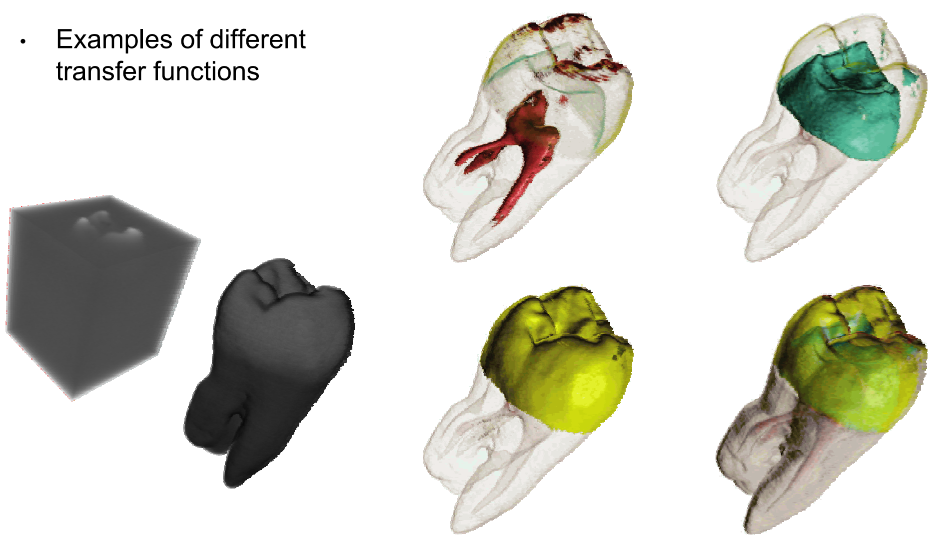


5.1. Classification

- Color table for volume visualization
- Maps raw voxel value into presentable entities:
  color, intensity, opacity, etc.
- Transfer function
- Goals and issues:
  - Empowers user to select “structures”
  - Extract important features of the data set
  - Classification is non trivial
  - Histogram can be a useful hint
  - Often interactive manipulation of transfer functions needed

5.1. Classification

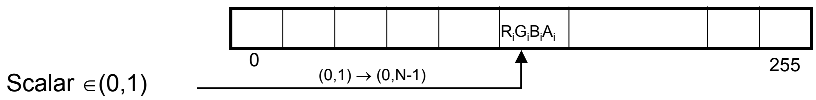
- Examples of different transfer functions



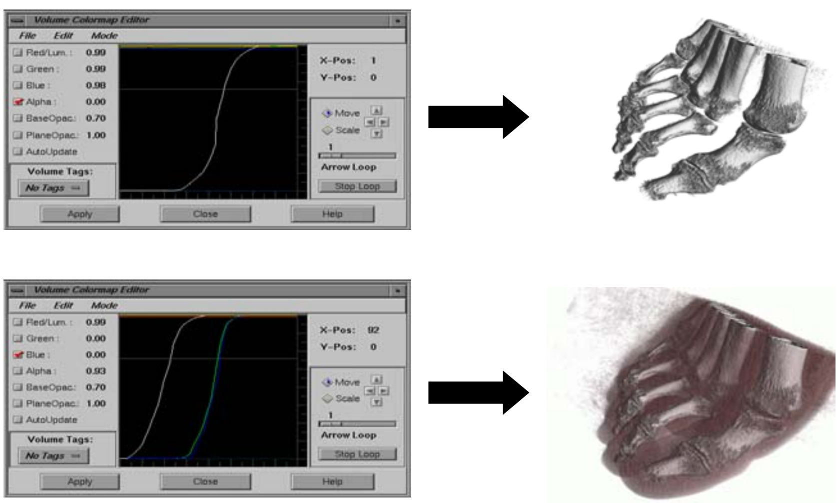
5.1. Classification

- Most widely used approach for transfer functions:
  - Assign to each scalar value a different color value
  - Assignment via transfer function *T*

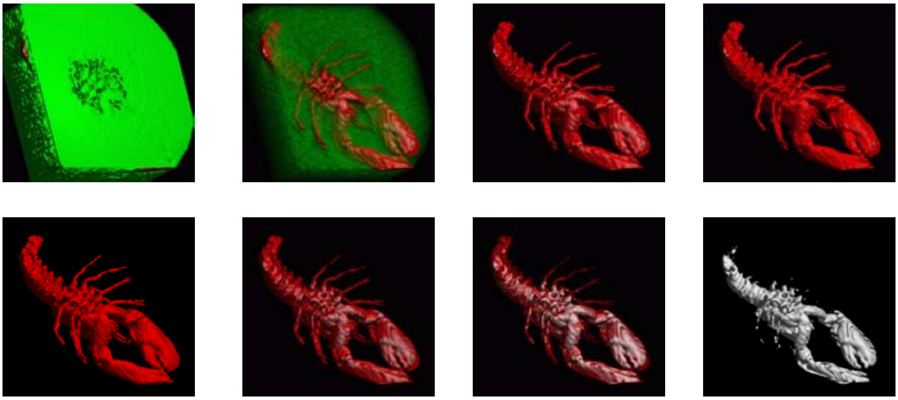
    $T : scalarvalue \rightarrow colorvalue$
  - Common choice for color representation: RGBA
  - Alpha value is very important, describes opacity
  - Code color values into a color lookup table
  - On-the-fly update of color LUT



5.1. Classification

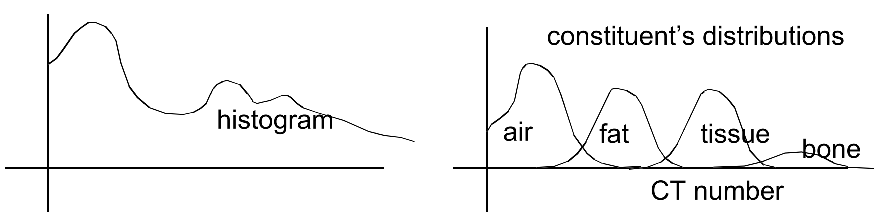


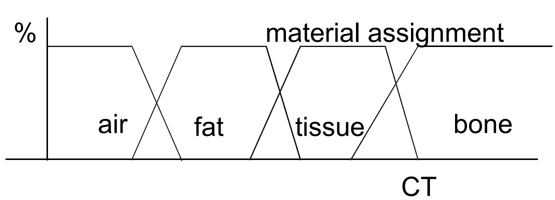
5.1. Classification



5.1. Classification

- Heuristic approach, based on measurements of many data sets





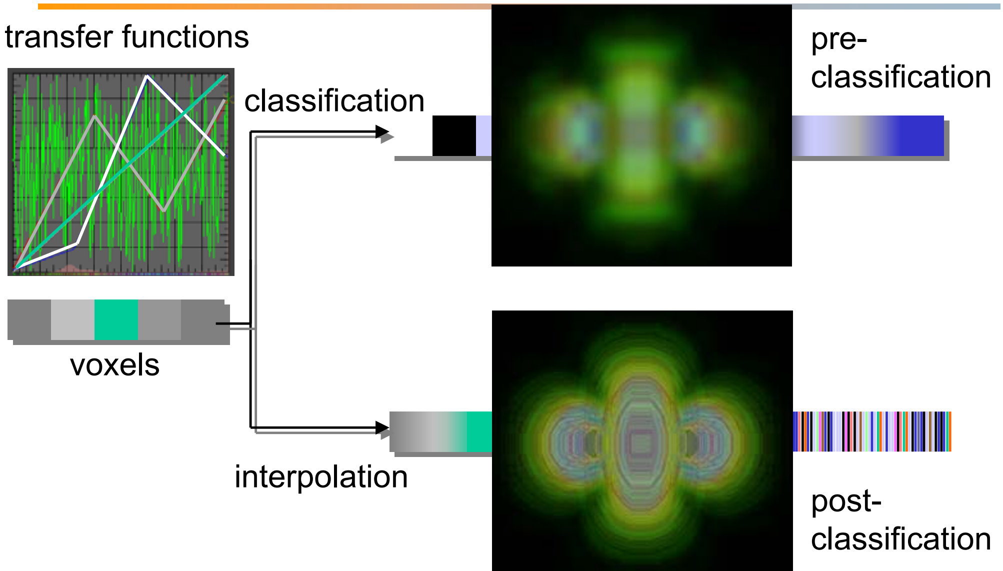
5.1. Classification

- Hounsfield units (HU) for CT data sets
  - Describes x-ray attenuation, i.e., density of material
  - 12 bit CT-measurements
  - Range of values from -1024 to +3071 HU
  - Typical values:
    - Air: -1024
    - Fat: -100 to -20
    - Water: 0
    - Soft tissue such as muscle: +20 to +80
    - Bone: > +500
  - For visualization 12 bit are reduced to 8 bit by windowing (loss of dynamic range)

5.1. Classification

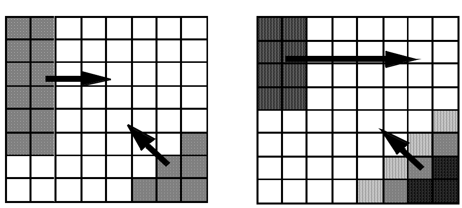
- Pre-shading
  - Assign color values to original function values
  - Interpolate between color values
- Post-shading
  - Interpolate between scalar values
  - Assign color values to interpolated scalar values

5.1. Classification



5.1. Classification

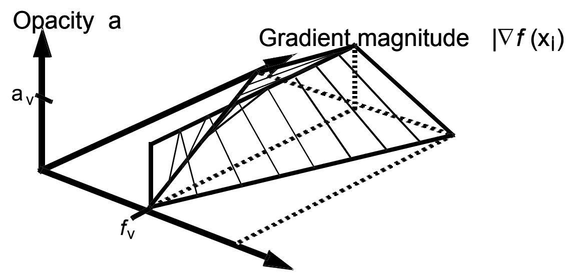
- Usually not only interested in a particular isosurface but also in regions of "change"
- Feature extraction - High value of opacity in regions of change
  - Homogeneous regions less interesting - transparent
- Surface "strength" depends on gradient
- Gradient of the scalar field is taken into account



5.1. Classification

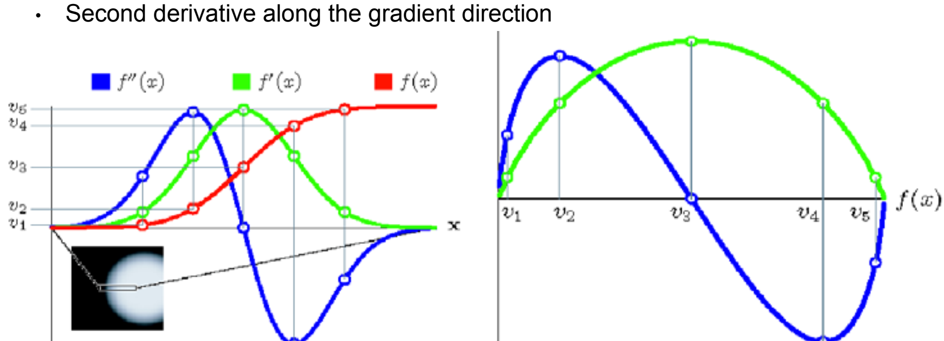
- Scalar value and gradient of the scalar field in a transfer function to emphasize isosurfaces [Levoy 1988]

$$\alpha(x_i) = \alpha_v \left(1 - \frac{1}{r}\left|\frac{f_v - f(x_i)}{|f'(x_i)|}\right|\right)$$



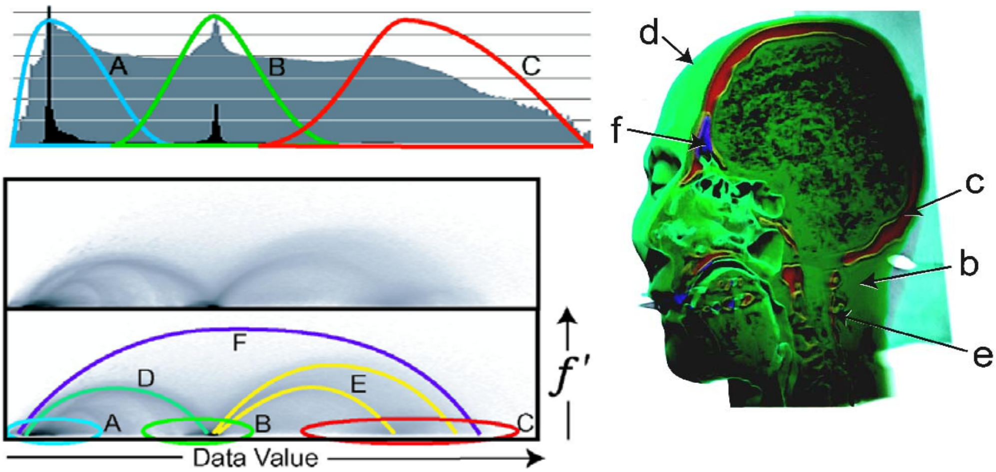
5.1. Classification

- Multidimensional transfer functions
  [Kindlmann & Durkin 98, Kniss, Kindlmann, Hansen 01]
- Problem: How to identify boundary regions/surfaces
- Approach: 2D/3D transfer functions, depending on
  - Scalar value, Magnitude of the gradient
  - Second derivative along the gradient direction



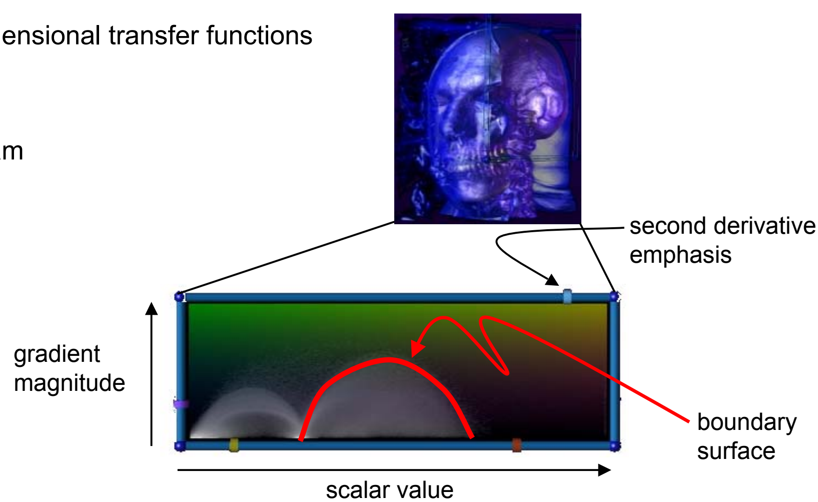
5.1. Classification

- Multidimensional transfer functions



5.1. Classification

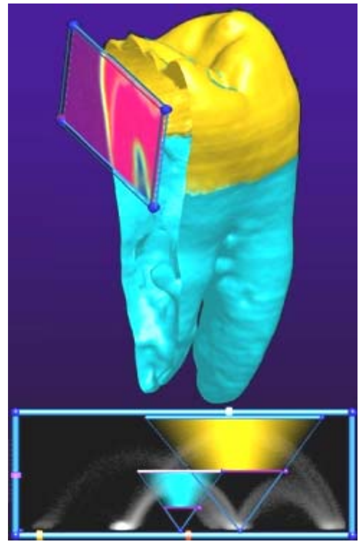
- Multidimensional transfer functions
- Histogram



5.1. Classification

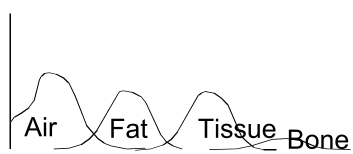
- Multidimensional transfer functions

- Extraction of two boundaries
- Triangle function in histogram

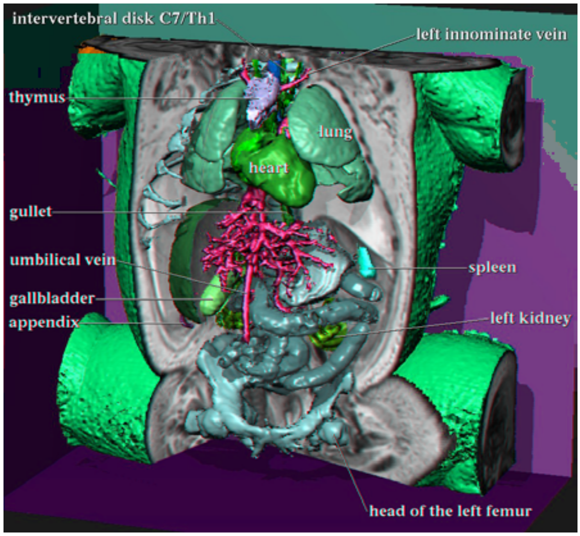


5.2. Segmentation

- Different features with same value
  - Example CT: different organs have similar X-ray absorption
  - Classification can not be distinguished
- Label voxels indicating a type
- Segmentation = pre-processing
- Semi-automatic process!!!



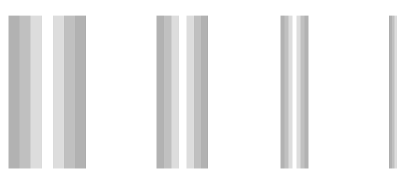
5.2. Segmentation



Anatomic atlas

5.3. Volumetric Shading

- Shading:
  - Simulate reflection of light
  - Simulate effect on color
- We want to make use of the human visual system’s ability to efficiently deal with shaded objects
- Interpretation of intensity gradient

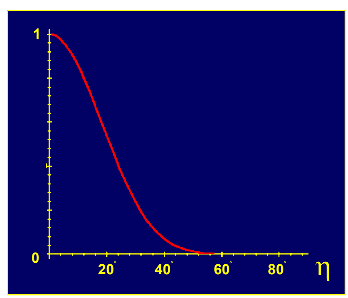


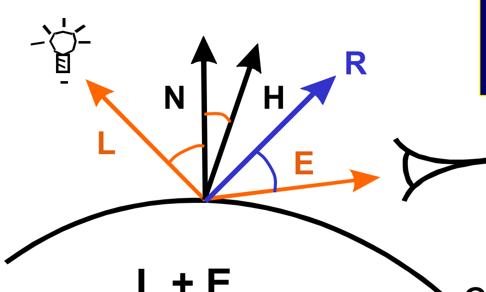
5.3. Volumetric Shading

- Review of the Phong illumination model
  - Ambient light + diffuse light + specular light
- Ambient light: $C = k_a C_a O_d$
  - k_a is ambient contribution
  - C_a is color of ambient light
  - O_d is diffuse color of object
- Diffuse light: $C = k_d C_p O_d \cos(\theta)$
  - k_d is diffuse contribution
  - C_p is color of point light
  - O_d is diffuse color of object
  - $\cos(\theta)$ is angle of incoming light
- Specular light: $C = k_s C_p O_s \cos^n(\sigma)$
  - k_s is specular contribution
  - C_p is color of point light
  - $\cos(\sigma)$ is angle of reflected light and eye
  - n is the specular exponent

5.3. Volumetric Shading

- $\cos(\theta) = \mathbf{N} * \mathbf{L}$
- $\cos(\sigma) = \mathbf{N} * \mathbf{H}$ (Blinn-Phong)

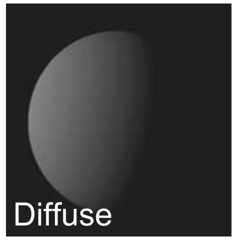


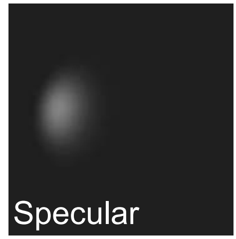


$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{E}}{\|\mathbf{L} + \mathbf{E}\|}$$

$$\cos(\sigma)^{10} = (\mathbf{N}\ \ \mathbf{H})^{10}$$

5.3. Volumetric Shading

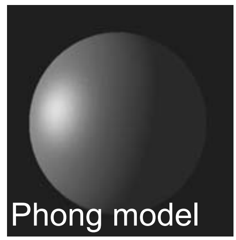




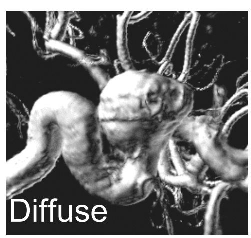
$K_a = 0.1$

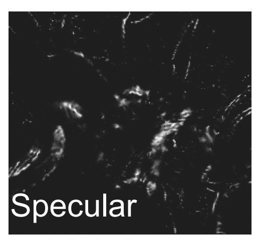
$K_d = 0.5$

$K_s = 0.4$



5.3. Volumetric Shading

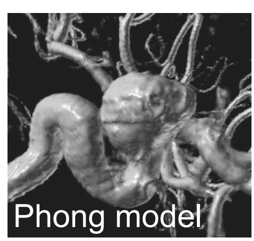




$K_a = 0.1$

$K_d = 0.5$

$K_s = 0.4$



5.3. Volumetric Shading

- What is the normal vector in a scalar field?
- Use the gradient!
- Gradient is perpendicular to isosurface
  (direction of largest change)

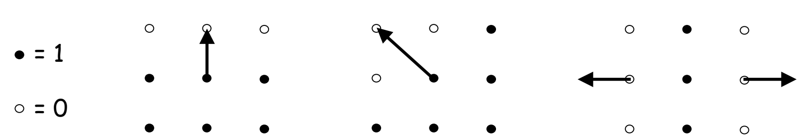
- Numerical computation of the gradient:
  - Central difference
  - Intermediate difference (forward/backward difference)
  - Sobel Operator

5.3. Volumetric Shading

- Central difference
  - Computation

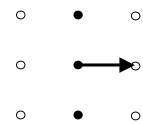
$$G_x = V_{x+1,y,z} - V_{x-1,y,z}$$
$$G_y = V_{x,y+1,z} - V_{x,y-1,z}$$
$$G_z = V_{x,y,z+1} - V_{x,y,z-1}$$

  - Convolution kernel: [-1 0 1]
  - High-pass filter
  - Not isotropic; length is 1 to sqrt(3)
  - Needs normalization



5.3. Volumetric Shading

- Intermediate difference (forward / backward)
  - Convolution kernel: [-1 1]
  - Very cheap
  - Detects high frequencies
  - Noisy data --> less good
  - Also not isotropic
- Sobel operator
  - Nearly isotropic
  - Very expensive (multiple multiplications and summations)

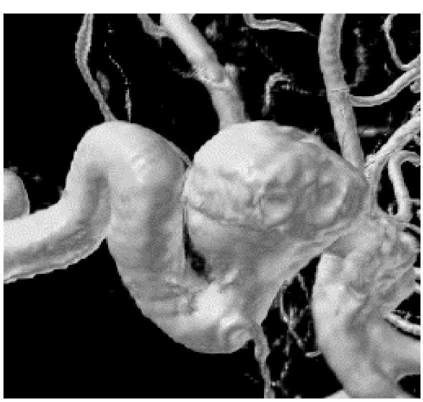


$$\begin{array}{ccc} \text{prev. slice} & \text{this z-slice} & \text{next slice} \\ \begin{bmatrix} -1 & 0 & 1 \\ -3 & 0 & 3 \\ -1 & 0 & 1 \end{bmatrix} & \begin{bmatrix} -3 & 0 & 3 \\ -6 & 0 & 6 \\ -3 & 0 & 3 \end{bmatrix} & \begin{bmatrix} -1 & 0 & 1 \\ -3 & 0 & 3 \\ -1 & 0 & 1 \end{bmatrix} \end{array}$$

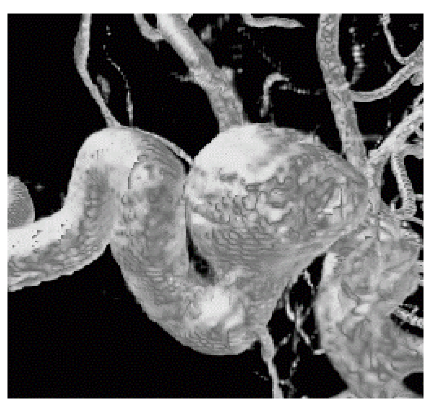
partial derivative along the *z*-axis

other axes by rotation

5.3. Volumetric Shading

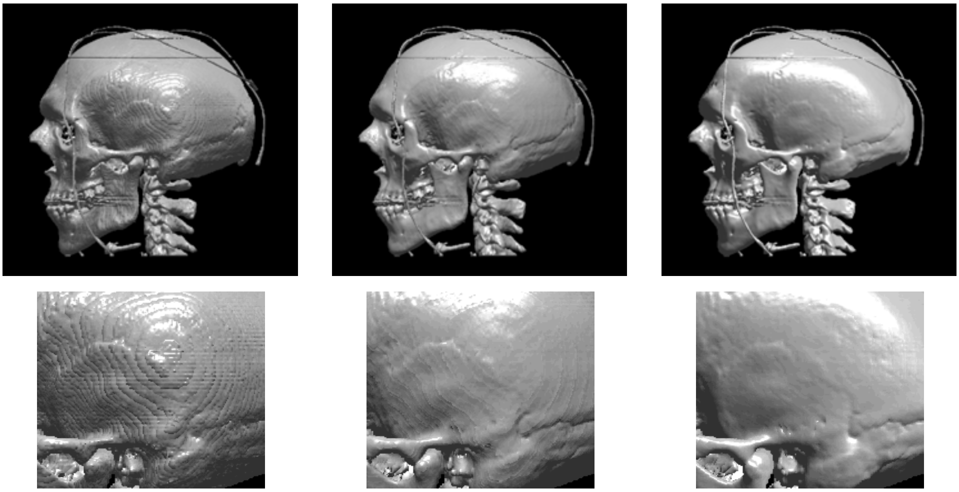


Central differences



Intermediate differences

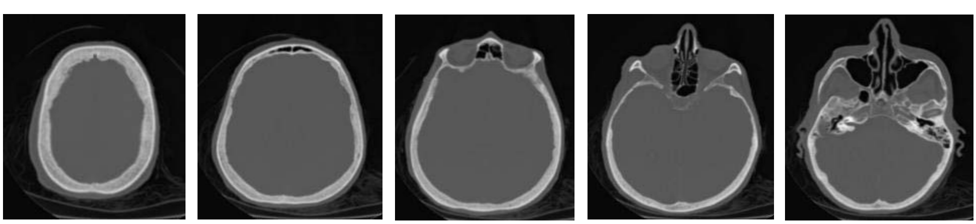
5.3. Volumetric Shading



Intermediate differences | Central differences | Sobel operator

5.4. Slicing

- Orthogonal slicing
  - Interactively resample the data on slices perpendicular to the x-,y-,z-axis
  - Use visualization techniques for 2D scalar fields
    - Color coding
    - Isolines
    - Height fields

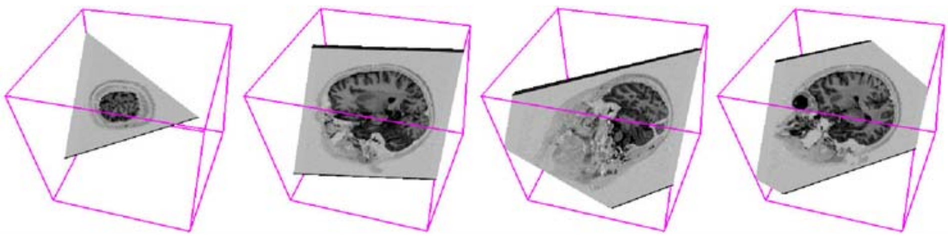


Slice 20 30 40 50 60

CT data set

5.4. Slicing

- Oblique slicing (MPR multiplanar reformating)
  - Resample the data on arbitrarily oriented slices
    - Resampling in software or hardware (trilinear interpolation)
    - Exploit 3D texture mapping functionality
      - Store volume in 3D texture
      - Compute sectional polygon (clip plane with volume bounding box)
      - Render textured polygon



5.5. Indirect Volume Rendering

- Indirect volume rendering
  - If *f(x,y,z)* is differentiable in every point then the level-sets *f(x,y,z)* = *c* are isosurfaces to the isovalue *c*
  - Techniques to determine and to reconstruct isosurfaces from volume data
    - Contour tracing
    - Cuberille, opaque cubes
    - Marching cubes/tetrahedra

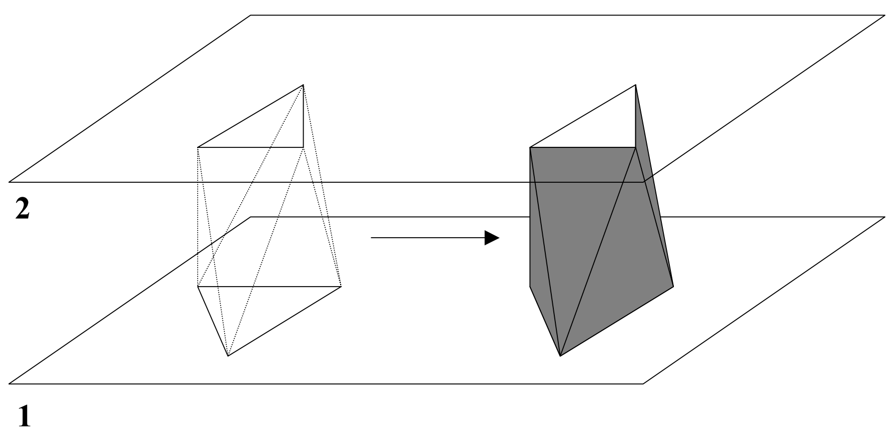
5.5. Indirect Volume Rendering

- Contour tracing
  - Find isosurfaces from 2D contours
    - *Segmentation:* find closed contours in 2D slices and represent them as polylines
    - *Labeling:* identify different structures by means of the isovalue of higher order characteristics
    - *Tracing:* connect contours representing the same object from adjacent slices and form triangles
    - *Rendering:* display triangles
  - Choose topological or geometrical reconstruction
  - Problems:
    - Sometimes there are many contours in each slice or there is a high variation between slices

    → Tracing (assignment) becomes very difficult

5.5. Indirect Volume Rendering

Contour tracing



5.5. Indirect Volume Rendering

- Generic surface fitting techniques
  - Choose an isovalue (arbitrarily or from segmentation)
  - Detect all cells the surface is passing through by checking the vertices
    - Mark vertices with respect to $f(x,y,z) \geq / < c$ (+/-)
    - Consider all cells with different signs at vertices
  - Place graphical primitives in each marked cell and render the surface

5.5. Indirect Volume Rendering

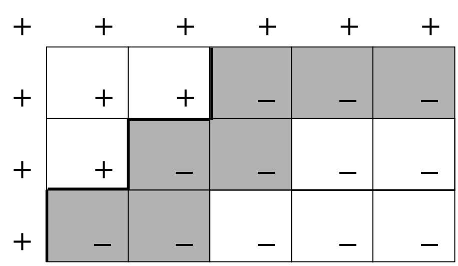
- Cuberille (opaque cubes) approach [Herman 1979]

  (A) Binarization of the volume with respect to the isovalue

  (B) Find all boundary front-faces

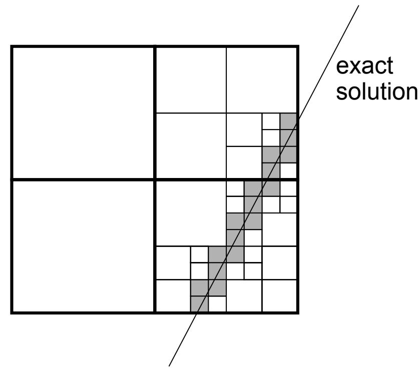
  if the normal of each face points outward the cell, find all faces where the normal points towards the viewpoint (N•V>0)

  (C) Render these faces as shaded polygons
- “Voxel” point of view: NO interpolation within cells



5.5. Indirect Volume Rendering

- Cuberille approach yields blocky surfaces
- Improve results by adaptive subdivision
  - Subdivide each marked cube into 8 smaller cubes
  - Use trilinear interpolation in order to reconstruct data values at new cell corners
  - Repeat cuberille approach for each new cube until pixel size



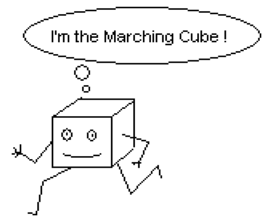
5.6. Marching Cubes

- In order to get a better approximation of the „real" isosurface the Marching-Cubes (MC) algorithm was developed [Lorensen, Cline 1987]
  - Works on the original data
  - Approximates the surface by a triangle mesh
  - Surface is found by linear interpolation along cell edges
  - Uses gradients as the normals of the isosurface
  - Efficient computation by means of lookup tables
- THE standard geometry-based isosurface extraction algorithm

5.6. Marching Cubes

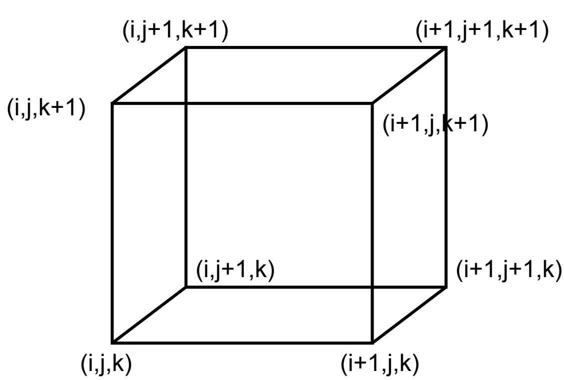
- The core MC algorithm
  - Cell consists of 4(8) pixel (voxel) values:
    (i+[01], j+[01], k+[01])

1. Consider a cell
2. Classify each vertex as inside or outside
3. Build an index
4. Get edge list from table[index]
5. Interpolate the edge location
6. Compute gradients
7. Consider ambiguous cases
8. Go to next cell



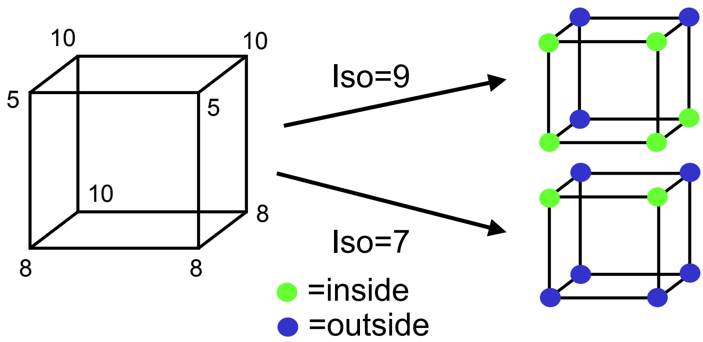
5.6. Marching Cubes

- Step 1: Consider a cell defined by eight data values



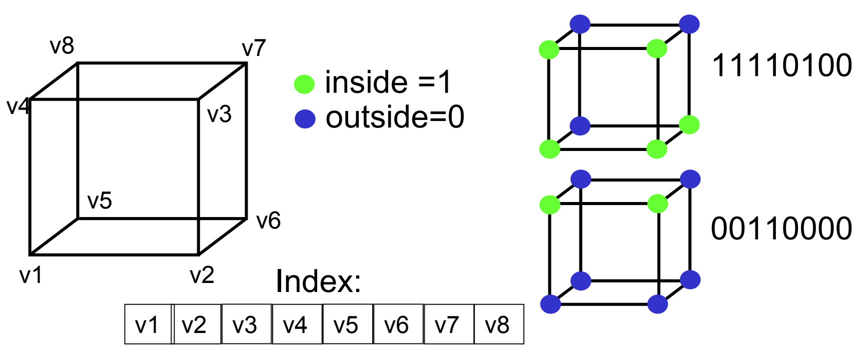
5.6. Marching Cubes

- Step 2: Classify each voxel according to whether it lies
  - outside the surface (value > isosurface value)
  - inside the surface (value <= isosurface value



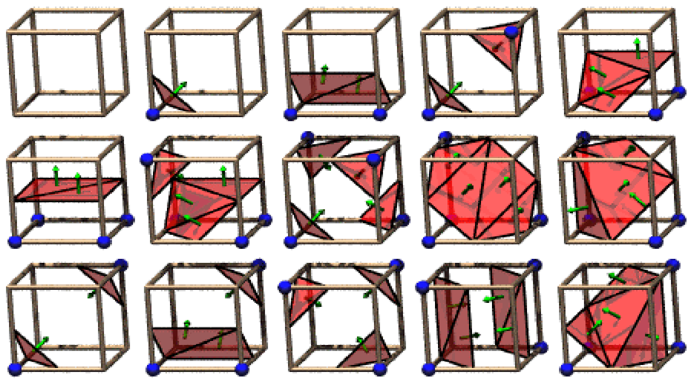
5.6. Marching Cubes

- Step 3: Use the binary labeling of each voxel to create an index



5.6. Marching Cubes

- Step 4: For a given index, access an array storing a list of edges
  - All 256 cases can be derived from 15 base cases due to symmetries

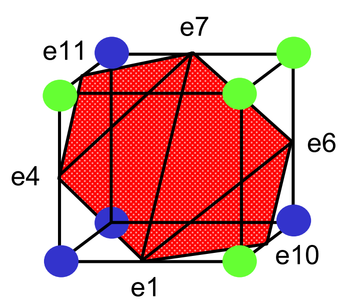


The 15 Cube Combinations

5.6. Marching Cubes

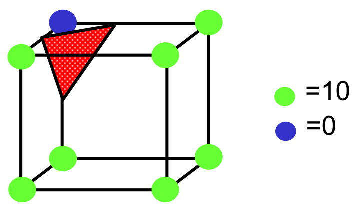
- Step 4 *cont*.: Get edge list from table
  - Example for

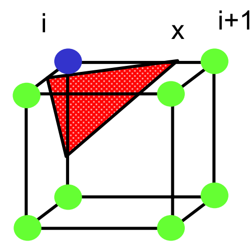
    Index = 10110001
    triangle 1 = e4,e7,e11
    triangle 2 = e1, e7, e4
    triangle 3 = e1, e6, e7
    triangle 4 = e1, e10, e6



5.6. Marching Cubes

- Step 5: For each triangle edge, find the vertex location along the edge using linear interpolation of the voxel values





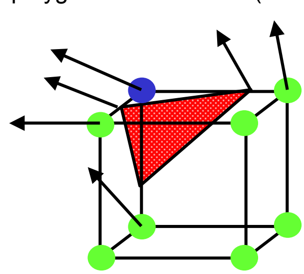
T=5

$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$

T=8

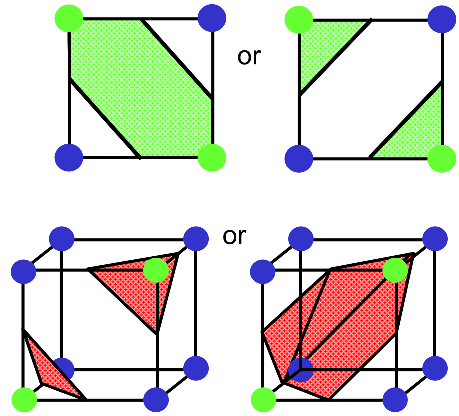
5.6. Marching Cubes

- Step 6: Calculate the normal at each cube vertex
  - $G_x = V_{x+1,y,z} - V_{x-1,y,z}$
    $G_y = V_{x,y+1,z} - V_{x,y-1,z}$
    $G_z = V_{x,y,z+1} - V_{x,y,z-1}$
  - Normalize
  - Use linear interpolation to compute the polygon vertex normal (of the isosurface)



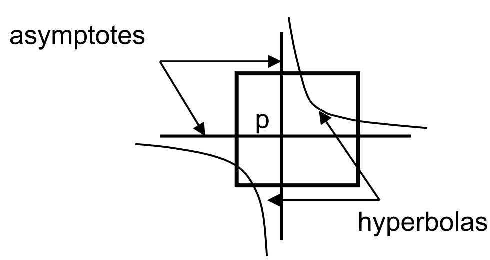
5.6. Marching Cubes

- Step 7: Consider ambiguous cases
  - Ambiguous cases: 3, 6, 7, 10, 12, 13
  - Adjacent vertices: different states
  - Diagonal vertices: same state
  - Resolution: decide for one case



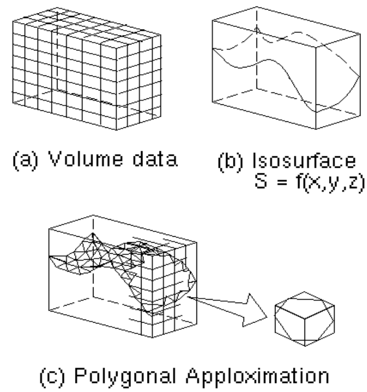
5.6. Marching Cubes

- Step 7 *cont.*: Consider ambiguous cases
- Asymptotic Decider [Nielson, Hamann 1991]
  - Assume bilinear interpolation within a face
  - Hence isosurface is a hyperbola
  - Compute the point p where the asymptotes meet
  - Sign of S(p) decides the connectivity



5.6. Marching Cubes

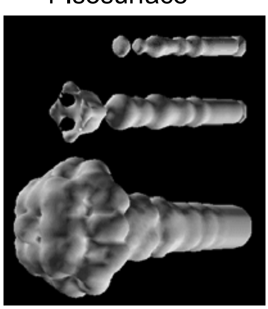
- Summary
  - 256 Cases
  - Reduce to 15 cases by symmetry
  - Ambiguity resides in cases 3, 6, 7, 10, 12, 13
  - Causes holes if arbitrary choices are made
- Up to 5 triangles per cube
- Dataset of 512^3 voxels can result in several million triangles (many Mbytes!!!)
- Semi-transparent representation --> sorting
- Optimization:
  - Reuse intermediate results
  - Prevent vertex replication
  - Mesh simplification

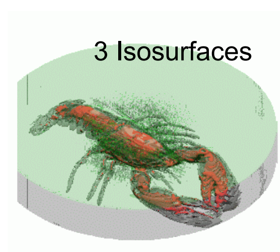


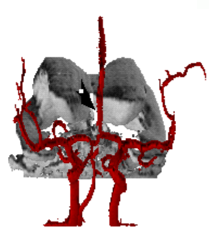
5.6. Marching Cubes

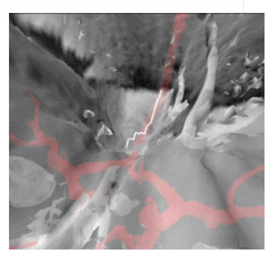
- Examples

1 Isosurface





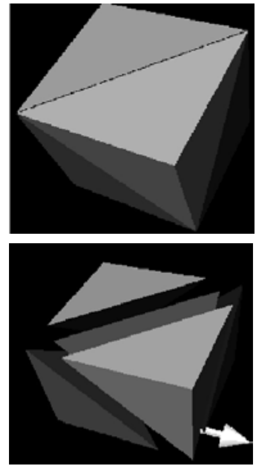




2 Isosurfaces

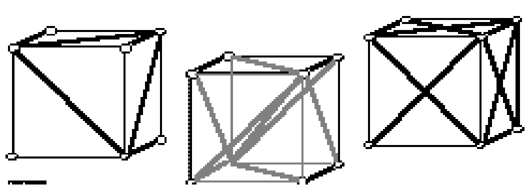
5.6. Marching Tetrahedra

- Marching Tetrahedra [Shirley et al. 1990]
- Primarily used for unstructured grids
  - Split cells into tetrahedra
- Process each tetrahedron similarly to the MC-algorithm
  - Two different cases:
  - A) one – and three + (or vice versa)
    - The surface is defined by one triangle
  - B) two – and two +
  - Sectional surface given by a quadrilateral – split it into two triangles using the shorter diagonal



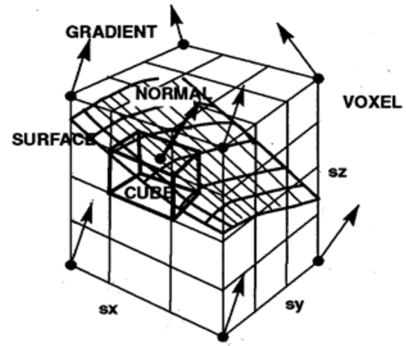
5.6. Marching Tetrahedra

- Properties
  - Fewer cases, i.e. 3 instead of 15
    - no problems with consistency between adjacent cells
  - Number of generated triangles might increase considerably compared to the MC-algorithm due to splitting into tetrahedra
  - Huge amount of geometric primitives
  - But, several improvements exist:
    - Hierarchical surface reconstruction
    - View-dependent surface reconstruction
    - Mesh decimation



5.7. Dividing Cubes

- Dividing cubes [Cline, Lorensen 1988]
- Uniform grids
- Basic idea
  - Create “surface points" instead of triangles
  - Associate surface normal with each surface
  - Surface points (when rendered) are pixel size
  - Subdivide cells as necessary



5.7. Dividing Cubes

- Algorithm:
  - Choose a cube
  - Classify, whether an isosurface is passing through it or not
  - If (surface is passing through)
    - Recursively subdivide cube until pixel size
  - Compute normal vectors at each corner
  - Render shaded points with averaged normal

5.7. Dividing Cubes

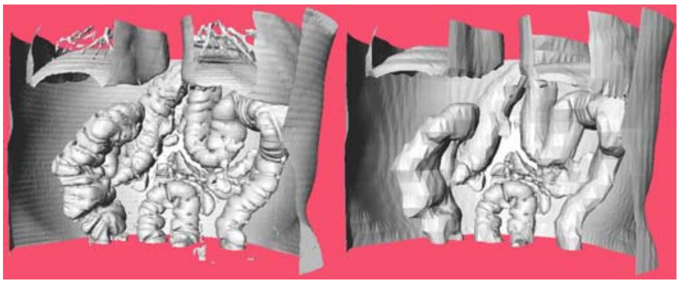
- Properties
  - View dependent load balancing
  - Better surface approximation due to interpolation within cells
  - Only good for rendering, but since no surface representation is generated it does not allow for further computations on the surface
  - Eliminates scan conversion step
  - Point cloud rendering randomly ordered points
  - No topology

5.8. Optimization of Fitted Surfaces

- All surface fitting techniques produce a huge amount of geometric primitives
- Several improvements exist:
  - Hierarchical surface reconstruction
  - View-dependent surface reconstruction
  - Mesh decimation

5.8. Optimization of Fitted Surfaces

- Hierarchical surface reconstruction
  - Generate copies of the data set at different resolutions
  - Select level-of-detail based on error criterion
    - Distance of coarse approximation to "original" surface



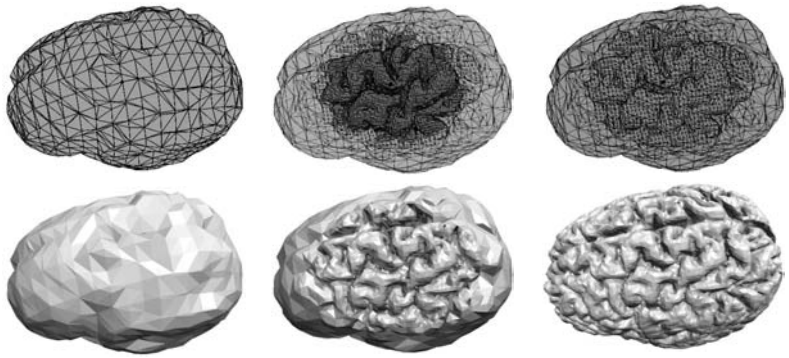
Full reconstruction (6M∆) LOD reconstruction (123K∆)

5.8. Optimization of Fitted Surfaces

- View-dependent surface reconstruction
  - User defined level-of-detail (focus point oracle, like a lens)
  - View frustum culling
    - Avoid reconstructing in regions that are outside the viewing pyramid
  - Occlusion culling
    - Avoid reconstruction in regions that are already occluded by the surface (implies front-to-back traversal)
  - Avoid reconstruction in cells that are below the pixel size

5.8. Optimization of Fitted Surfaces

- View-dependent surface reconstruction



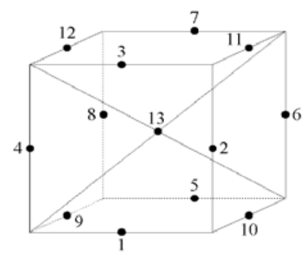
5.8. Optimization of Fitted Surfaces

- Mesh decimation
  - Remove triangles and re-triangulate with less triangles
  - Consider deviation between mesh before and after decimation
  - Generate hierarchical mesh structure as a post-process and switch to appropriate resolution during display

5.9. Discretized Marching Cubes

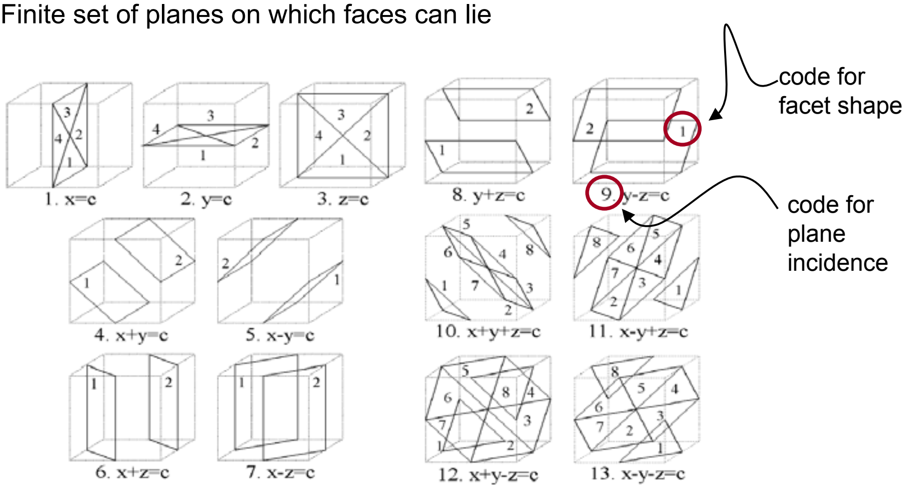
- Discretized Marching Cubes (DiscMC) [Montani et al. 1994]
- Accelerate standard MC
- Mixture in-between:
  - Cuberille approach (constant scalar value on each voxel)
  - Marching Cubes (trilinear interpolation in cells)

- Approximation of MC: discrete positions for vertices of isosurface
  - 13 different vertex positions
  - 12 edge-midpoints + 1 centroid



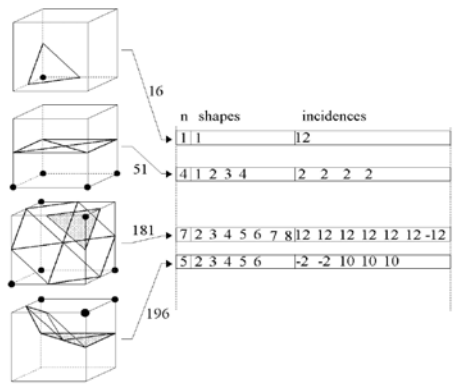
5.9. Discretized Marching Cubes

- Finite set of planes on which faces can lie



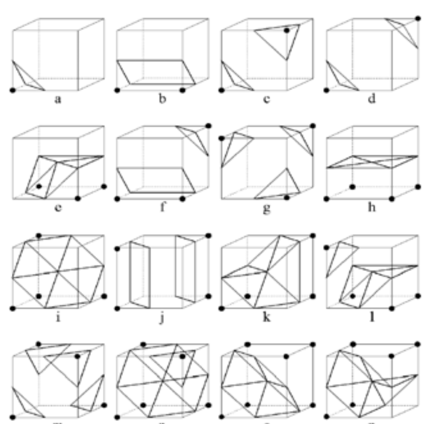
5.9. Discretized Marching Cubes

- Classification of a facet by
  - Plane incidence and
  - Shape
- Sign of incidence determines orientation of facet
- Classification of isosurface fragment (facet set)
  - Indices to incidences and shapes



5.9. Discretized Marching Cubes

- Lookup table
  - Based on MC LUT
  - Simple reorganization
  - Indices as above
- Vertex positions of facet determined by vertex configuration of cell
- No linear interpolation needed

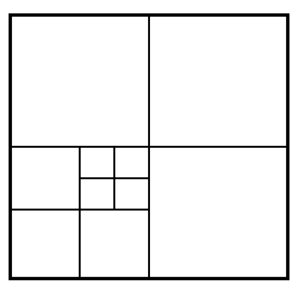


5.9. Discretized Marching Cubes

- Algorithm:
  - Analogously to MC: traversing the grid
  - Normal vectors based on gradients (same as MC)
  - Postprocessing: merging facets and edges
- Advantages:
  - Simple classification of facet sets
  - Many coplanar facets due to small number of plane incidences -> significantly reduces number of triangles after merge
  - No interpolation needed, i.e., only integer arithmetic
  - Still quite good results

5.10. Octree-Based Isosurface Extraction

- Acceleration of MC
- Domain search
- Octree-based approach [Wilhelms, van Gelder 1992]
  - Spatial hierarchy on grid (tree)
  - Store minimum and maximum scalar values for all children with each node
  - While traversing the octree, skip parts of the tree which cannot contain the specified isovalue



5.10. Octree-Based Isosurface Extraction

- What data structure for octree?
- Advantages of full octree:
  - Simple array-like structure and organization
  - No pointers needed
- Number of nodes in full octree:

$$n_{\text{nodes}} = \sum_{i=0}^{(\log_2 s)-1} 8^i = \frac{s^3 - 1}{7} \approx 0.14\, n_{\text{data points}}$$

-> optimal ratio is $n_{\text{nodes}} \,/\, n_{\text{data points}} \approx 0.14$

5.10. Octree-Based Isosurface Extraction

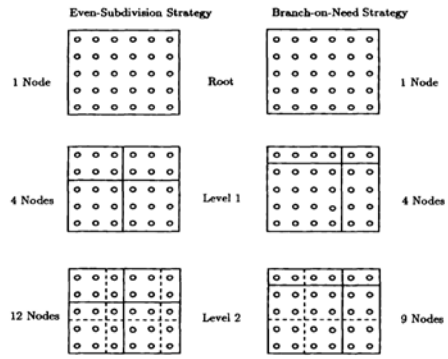
- Problem with memory consumption of complete octree:
  - Ideal: grid size of $2^n \bullet 2^n \bullet 2^n$
  - Normally different resolutions that are not powers of two
- Example:
  - Data set: 320•320•40
  - 4M data points
  - Full octree:
    $1+2^3+4^3+ \ldots +256^3$ = 20M elements (nodes)
  - 2 values per element: minimum and maximum values

5.10. Octree-Based Isosurface Extraction

- Solution: Branch on Need Octree (BONO)
  - Consider octree as conceptionally full
  - Avoid allocating memory for empty subspaces
  - Delay subdivision until needed
  - Allocate only dimensions of powers of two
  - Aspects of a bottom-up approach

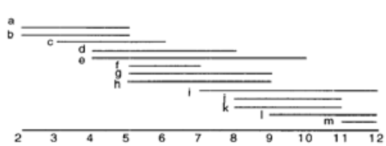
  - For above example: approx. 585k nodes (opposed to 20M nodes)

  - Ratio almost optimal: $n_{\text{nodes}} / n_{\text{data points}} \approx 0.1428$
  - Ratio never exceeds 0.162

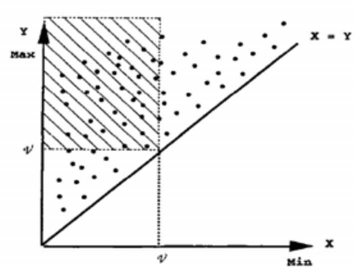


5.11. Range Query for Isosurface Extraction

- Acceleration of MC
- Data structures based on scalar values (not on domain decomposition)
- Store minimum and maximum values for each cell in special data structures



interval structure for min/max

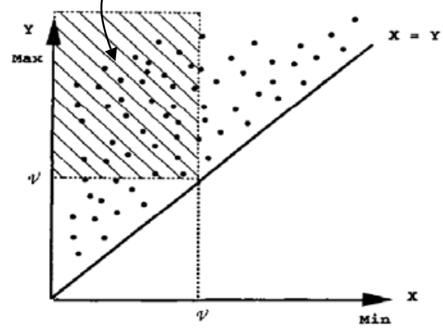


in span space

5.11. Range Query for Isosurface Extraction

- Each point in span space represents one cell with respective min/max values
- Relevant cells lie in rectangular region in span space

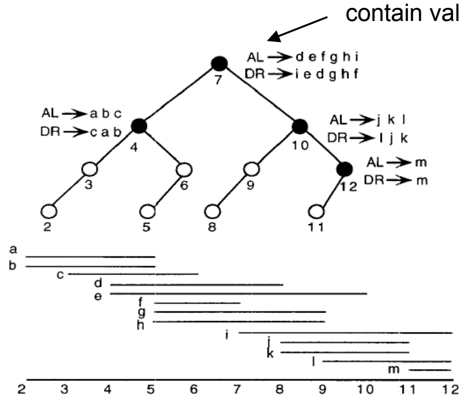
- Problem:
  How can all these cells
  be efficiently found?



5.11. Range Query for Isosurface Extraction

- "Optimal isosurface extraction from irregular volume data"
  [Cignoni et al. 1996]
- Interval tree
  - *h* different extreme scalar values
  - Balanced tree: height log *h*
  - Bisecting the discriminant scalar value
  - Node contains
    - Scalar values
    - Sorted intervals *AL* (ascending left)
    - Sorted (same) intervals *DR* (descending right)

contain value 7



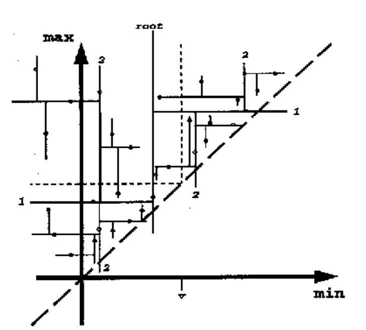
5.11. Range Query for Isosurface Extraction

- “Optimal isosurface extraction from irregular volume data”
- Running time: $O(k + \log h)$ due to
  - Traversal of interval tree: log *h* (height of the tree)
  - *k* intervals in the node = number of relevant cells (i.e., output sensitive)

5.11. Range Query for Isosurface Extraction

- Variations of the above range query based on interval trees:
  - Near optimal isosurface extraction (NOISE) [Livnat et al. 1996]
  - Isosurfacing in span space with utmost efficiency (ISSUE) [Shen et al. 1996]

- NOISE
  - Based on span space
  - Kd-tree for span space
  - Worst caser running time: $O(k + \text{sqrt}(n))$ with
  - *k* = number of relevant cells (with isosurface)
  - *n* = total number of grid cells

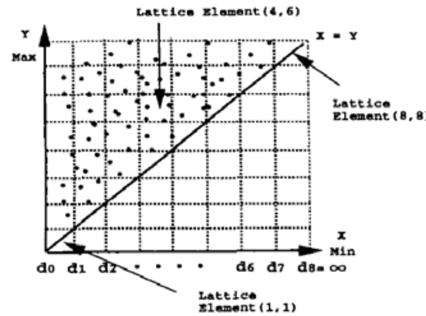


5.11. Range Query for Isosurface Extraction

- ISSUE: Isosurfacing in span space with utmost efficiency
  - Based on span space
  - Lattice subdivision on span space
  - Average running time:

$$O\left(k + \log\left(\frac{n}{L}\right) + \frac{\sqrt{n}}{L}\right)$$

  - L = dimension of grid in x and y



5.11. Range Query for Isosurface Extraction

- All range-query algorithms suitable
  for structured and unstructered grids

5.12. Contour Propagation

- Acceleration of cell traversal
- Algorithm:
  - Trace isosurface starting at a seed cell
  - Breadth-first traversal along adjacent faces
  - Finally, cycles are removed, based on marks at already traversed cells

- Similar to 2D approach
- Same problem:
  - Find ALL disconnected isosurfaces
  - Issue of optimal seed set