Volume Visualization

Overview: Volume Visualization (1)

- Introduction to volume visualization
  - On volume data
  - Voxels vs. cells
  - Interpolation
  - Gradient
  - Classification
  - Transfer Functions (TF)
  - Slice vs surface vs. volume rendering
  - Overview: techniques

Overview: Volume Visualization (2)

- Simple methods
  - Slicing, multi-planar reconstruction (MPR)
- Direct volume visualization
  - Image-order vs. object-order
  - Raycasting
  - α-compositing
  - Hardware volume visualization
- Indirect volume visualization
  - Marching cubes

Volume Visualization

- Introduction:
  - VolVis = visualization of volume data
    - Mapping 3D$\rightarrow$2D
    - Projection (e.g., MIP), slicing, vol. rendering, …
  - Volume data =
    - 3D$\times$1D data
    - Scalar data, 3D data space, space filling
  - User goals:
    - Gain insight in 3D data
    - Structures of special interest + context

Volume Data

- Where do the data come from?
  - Medical Application
    - Computed Tomographie (CT)
    - Magnetic Resonance Imaging (MR)
  - Materials testing
    - Industrial-CT
  - Simulation
    - Finite element methods (FEM)
    - Computational fluid dynamics (CFD)
  - etc.

3D Data Space

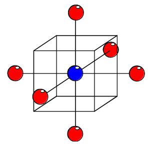
- How are volume data organized?
  - Cartesian resp. regular grid:
    - CT/MR: often dx=dy<dz, e.g. 135 slices (z) á 512^2 values (as x & y pixels in a slice)
    - **Data enhancement**: iso-stack-calculation = Interpolation of additional slices, so that dx=dy=dz $\Rightarrow$ 512^3 Voxel
    - Data: **Cells** (cuboid), Corner: **Voxel**
  - Curvi-linear grid resp. unstructured:
    - Data organized as tetrahedra or hexahedra
    - Often: conversion to tetrahedra

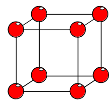
VolVis – Challenges

- Rendering projection,
  so much information and so few pixels!
- Large data sizes, e.g.
  512×512×1024 voxel á 16 bit = 512 Mbytes
- Speed,
  Interaction is very important, >10 fps!

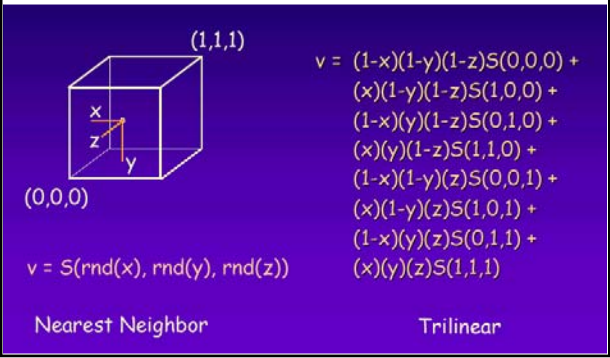
Voxels vs. Cells

- Two ways to interpret the data:
  - Data: set of voxel
    - **Voxel** = abbreviation for volume element (cf. pixel = "picture elem.)
    - Voxel = point sample in 3D
    - Not necessarily interpolated
  - Data: set of cells
    - Cell = cube primitive (3D)
    - Corners: 8 voxel (see above)
    - Values in cell: interpolation used

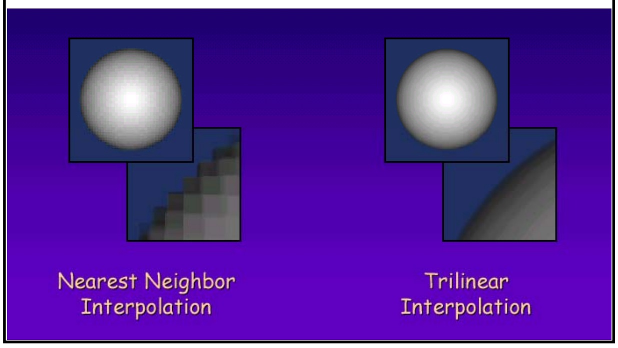




Interpolation

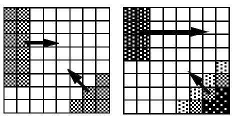


Interpolation – Results



Gradients in Volume Data

- Volume data: $f(\mathbf{x}) \in R^1$, $\mathbf{x} \in R^3$
- Gradient ∇f: 3D vector points in direction of largest function change
- Gradient magnitude: length of gradient
- Emphasis of changes:
  - Special interest often in transitional areas
  - Gradients: measure degree of change (like surface normal)
  - Larger gradient magnitude $\Rightarrow$ larger opacity



Gradients as Normal Vector Replacement

- Gradient $\nabla f = (\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)$
- $\nabla f|_{x_0}$ normal vector to iso-surface $f(x_0)=f_0$
- Central difference in x-, y- & z-direction (in voxel):

$$\nabla f\,(x,y,z) \;=\; 1/2 \begin{pmatrix} f(x+1)-f(x-1) \\ f(y+1)-f(y-1) \\ f(z+1)-f(z-1) \end{pmatrix}$$

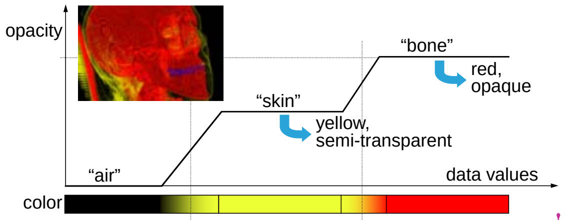
- Then tri-linear interpolation within a cell
- Alternatives:
  - Forward differencing: $\nabla f\,(x)=f(x+1)-f(x)$
  - Backwards differencing: $\nabla f\,(x)=f(x)-f(x-1)$
  - Intermediate differencing: $\nabla f\,(x+0.5)=f(x+1)-f(x)$

Classification

- Assignment data ⇒ semantics:
  - Assignment to objects, e.g., bone, skin, muscle, etc.
  - Usage of data values, gradient, curvature
  - Goal: segmentation
  - Often: semi-automatic resp. manual
  - Automatic approximation: transfer functions (TF)

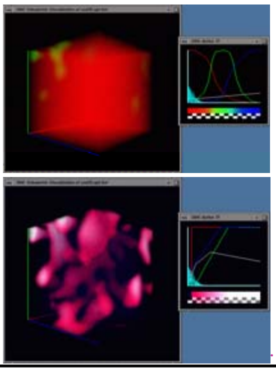
Transfer Functions (TF)

- Mapping data → "renderable quantities":
  - 1.) data→color (f(i)→C(i))
  - 2.) data→opacity (non-transparency) (f(i)→α(i))



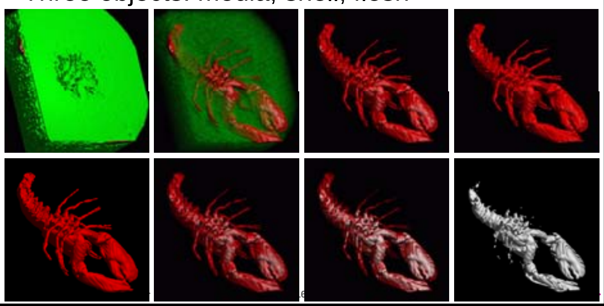
Different Transfer Functions

- Image results:
  - Strong dependence on transfer functions
  - Non-trivial specification
  - Limited segmentation possibilities

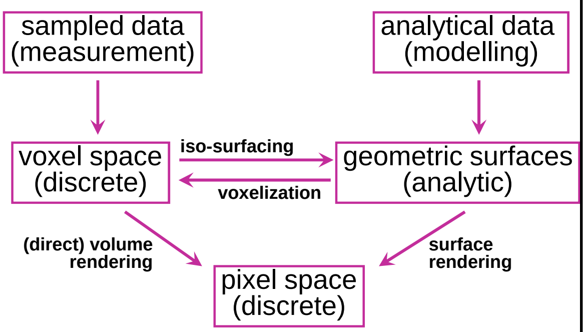


Lobster – Different Transfer Functions

- Three objects: media, shell, flesh

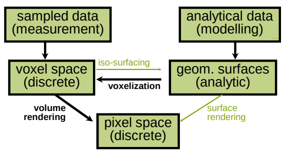


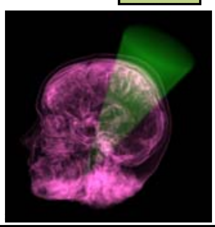
Concepts and Terms



Concepts and Terms

- Example
  - X-Ray Modelling
  - Surface-definition
  - Sampling (voxelization), combination
  - Direct volume rendering



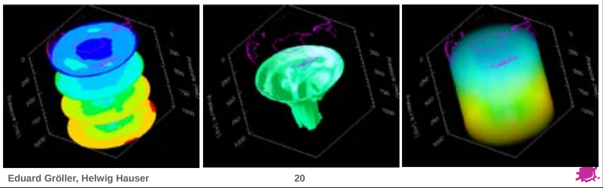


Slice vs. Surface vs. Volume Rendering

- Slice rendering
  - 2D cross-section from 3D volume data
- Surface rendering:
  - **Indirect** volume visualization
  - Intermediate representation: iso-surface, “3D”
  - Pros: Shading→Shape!, HW-rendering
- Volume rendering:
  - **Direct** volume visualization
  - Usage of transfer functions
  - Pros: illustrate the interior, semi-transparency

Slices **vs.** Iso-Surfaces. **vs.** Volume Rendering

- Comparison ozon-data over Antarctica:
  - Slices: selective (z), 2D, color coding
  - Iso-surface: selective (f_0), covers 3D
  - Vol. rendering: transfer function dependent, “(too) sparse – (too) dense”



VolVis-Techniques – Overview

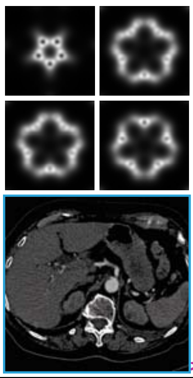
- Simple methods:
  - Slicing, MPR (multi-planar reconstruction)
- Direct volume visualization:
  - Ray casting
  - Shear-warp factorization
  - Splatting
  - 3D texture mapping
  - Fourier volume rendering
- Surface-fitting methods:
  - Marching cubes (marching tetrahedra)

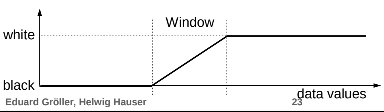
Simple Methods

Slicing, etc.

Slicing

- Slicing:
  - Axes-parallel slices
  - Regular grids: simple
  - Without transfer function no color
  - Windowing: adjust contrast
- General grid, arbitrary slicing direction





Direct Volume Visualization

Image-Order vs. Object-Order

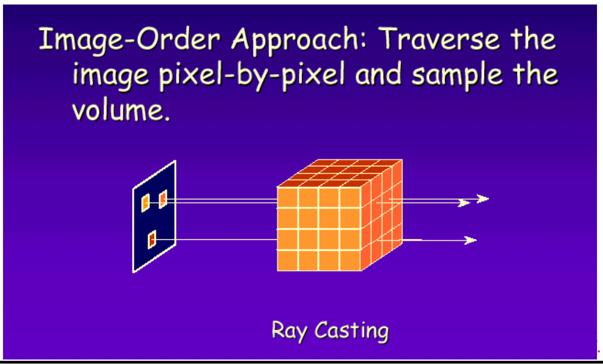
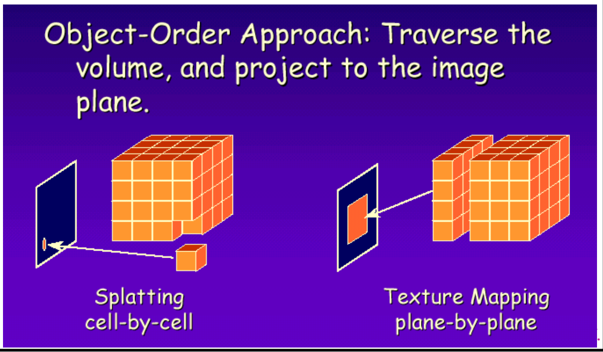


Image-Order vs. Object-Order



Ray Casting

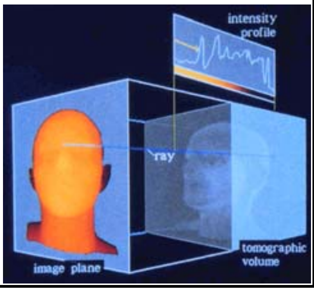
Image-Order Method

Ray Tracing vs. Ray Casting

- Ray Tracing: method from image generation
- In volume rendering: only viewing rays
  ⇒ therefore Ray Casting
- Classical image-order method
- Ray Tracing: ray – object intersection
  Ray Casting: no objects, density values in 3D
- In theory: take all density values into account!
  In practice: traverse volume step by step
- Interpolation necessary for each step!

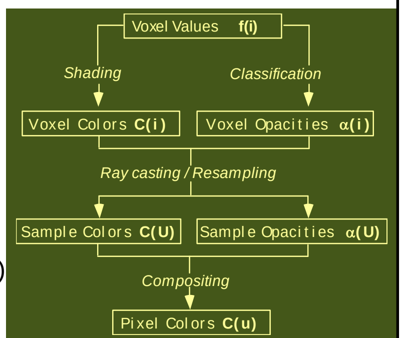
Ray Traversal through Volume Data

- Context:
  - Volume data: 1D value defined in 3D – $f(\mathbf{x}) \in R^1$, $\mathbf{x} \in R^3$
  - Ray defined as half-line: $\mathbf{r}(t) \in R^3$, $t \in R^1 > 0$
  - Values along Ray: $f(\mathbf{r}(t)) \in R^1$, $t \in R^1 > 0$ (intensity profile)



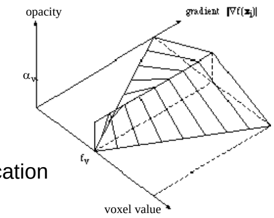
Standard Ray Casting

- Levoy '88:
- 1. $C(i)$, $\alpha(i)$ (from TF)
- 2. Ray casting, interpolation
- 3. Compositing (or combinations)

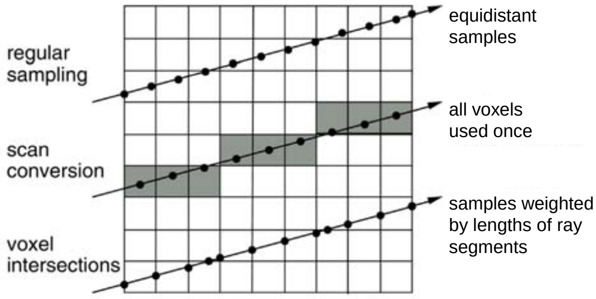


1. Shading, Classification

- 1. Step:
  - Shading, f(i)→C(i):
    - Apply transfer function
    - diffuse illumination (Phong), gradient ≈ normal
  - Classification, f(i)→α(i):
    - Levoy '88, gradient enhanced
    - Emphasizes transitions
  - Nowadays: shading/classification after ray-casting/resampling

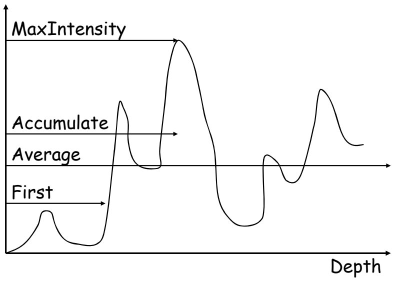


2. Ray Traversal – Three Approaches



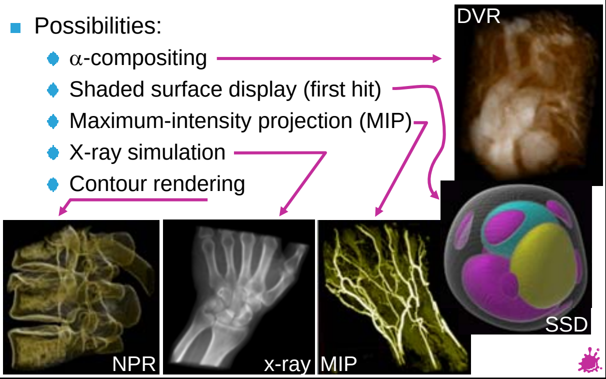
3. Types of Combinations

- Overview:
  - MIP ⇒
  - Compositing ⇒
  - X-Ray ⇒
  - First hit ⇒



Types of Combinations

- Possibilities:
  - α-compositing
  - Shaded surface display (first hit)
  - Maximum-intensity projection (MIP)
  - X-ray simulation
  - Contour rendering



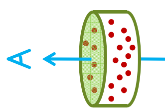
α-Compositing –

a Specific Optical Model for Volume Rendering

Display of
Semi-Transparent Media

Modelling of Natural Phenomena

- Various models (Examples):
  - Emission only (light particles)
  - Absorption only (dark fog)
  - Emission & absorption (clouds)
  - Single scattering, w/o shadows
  - Multiple scattering
- Two approaches:
  - Analytical model (via differentials)
  - Numerical approximation (via differences)



Emission and Absorption

- Continuous model (no scattering):
  - At each position is given:
    - Emission $g(t)$
    - Extinction coefficient $\tau(t)$
  - Differential $d\mathbf{I}/dt = g(t) - \tau(t)\mathbf{I}(t)$
  - Emission $g(t)$ attenuated by $T(t,s)$
  - Only Emission: $\mathbf{I}_0 \quad + \int_{t\in[0,s]} g(t)\,dt$
  - With Absorption: $\mathbf{I}_0 \cdot T(0,s) + \int_{t\in[0,s]} g(t)\cdot T(t,s)\,dt$
  - Emission und Absorption:

$$\mathbf{I}_0 \cdot \exp(-\int_{u\in[0,s]} \tau(u)du) + \int_{t\in[0,s]} g(t)\cdot \exp(-\int_{u\in[t,s]} \tau(u)du)\,dt$$

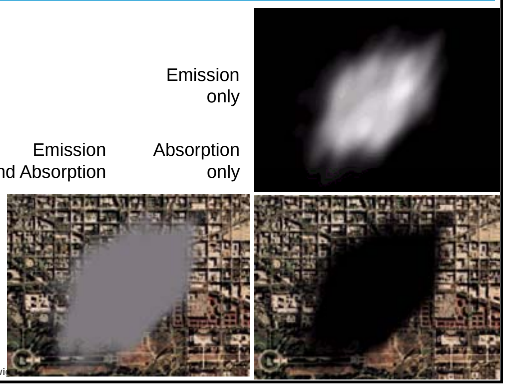
Numerical Approximation

- Discrete model (compositing):
  - For each volume extent i:
    - Contribution C_i
    - Opacity α_i, transparency $1-\alpha_i$
  - $Out_i = In_i \cdot (1-\alpha_i) + C_i \cdot \alpha_i$ (Std.-compositing)
  - Convex combination from background and own contribution
  - $Out_s = In_0 \cdot \Pi_{s \geq k \in N}(1-\alpha_k) + \Sigma_{s \geq k \in N} C_k \cdot \alpha_k \cdot \Pi_{s \geq l > k}(1-\alpha_l)$
  - Opacity-weighted colors: $C_i \cdot \alpha_i$ instead of C_i

Emission and Absorption

- Differential model:
  - $\mathbf{I}(s) = \mathbf{I}_0 \cdot T(0,s) + \int_{t\in[0,s]} g(t)\cdot T(t,s)\,dt$
  - $\mathbf{I}(s) = \mathbf{I}_0 \cdot \exp(-\int_{u\in[0,s]}\tau(u)du) + \int_{t\in[0,s]} g(t)\cdot \exp(-\int_{u\in[t,s]}\tau(u)du)\,dt$
- Discrete Approximation:
  - $Out_i = In_i \cdot (1-\alpha_i) + C_i \cdot \alpha_i$ (Compositing)
  - $Out_s = In_0 \cdot \Pi_{s \geq k \in N}(1-\alpha_k) + \Sigma_{s \geq k \in N} C_k \cdot \alpha_k \cdot \Pi_{s \geq l > k}(1-\alpha_l)$

Emission or/and Absorption

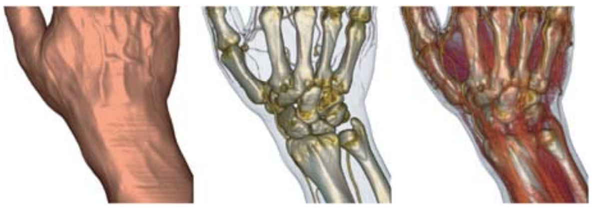


Compositing: F2B vs. B2F

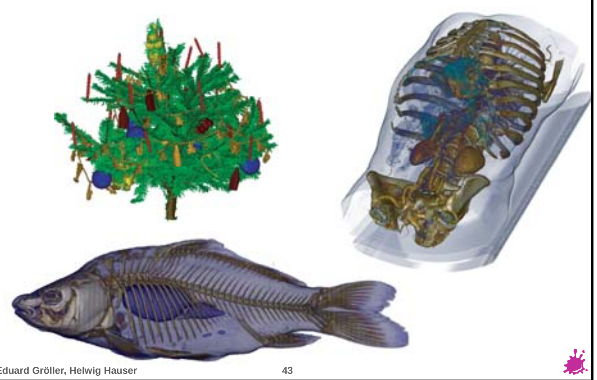
- Back-to-Front (B2F):
  - $Out_i = In_i \cdot (1-\alpha_i) + C_i \cdot \alpha_i$, $In_{i+1} = Out_i$ …
  - Depending on local transparency $(1-\alpha_i) \Rightarrow$ convex combination of old In_i & new C_i
  - Example:
    - Voxel i: C_i = red, α_i=30%; so far: In_i = white
    - Result of compositing: 70% white + 30% red
- Front-to-Back (F2B):
  - $Col = Col + (1-\alpha_{akk}) \cdot C_i \cdot \alpha_i$ … accumulated color
  - $\alpha_{akk} = \alpha_{akk} + (1-\alpha_{akk}) \cdot \alpha_i$ … accumulated opacity

Ray Casting – Examples

- CT scan of human hand (244x124x257, 16 bit)

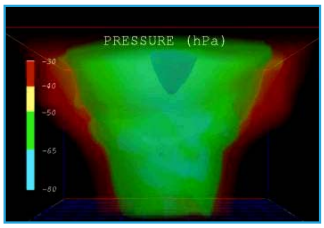


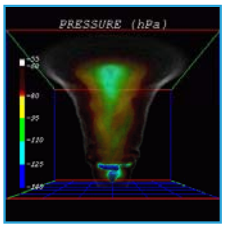
Ray Casting – Examples



Ray Casting – Further Examples

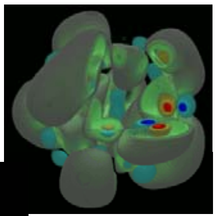
- Tornado Visualization:

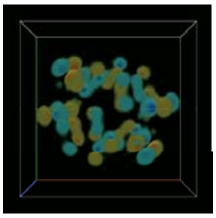


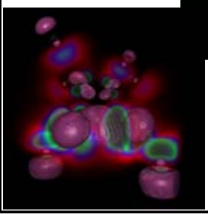


Ray Casting – Further Examples

- Molecular data:







Hardware-Volume Visualization

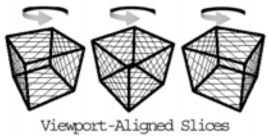
Faster with Hardware?!

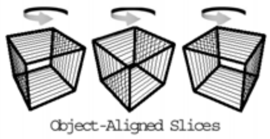
Two Approaches

- 3D-textures:
  - Volume data stored in 3D-texture
  - Proxy geometry (slices) parallel to image plane, are interpolated tri-linearly
  - Back-to-front compositing
- 2D-textures:
  - 3 stacks of slices (x-, y- & z-axis), slices are interpolated bi-linearly
  - Select stack (most “parallel” to image plane)
  - Back-to-front compositing

Variation of View Point

- 3D-textures:
  - Number of slices arbitrary
- 2D-textures:
  - Stack change: discontinuity



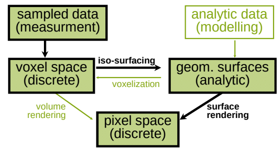


Indirect Volume Visualization

Iso-Surface-Display

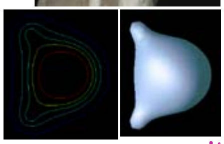
Concepts and Terms

- Example
  - CT measurement
  - Iso-stack-conversion
  - Iso-surface-calculation (marching cubes)
  - Surface rendering (OpenGL)



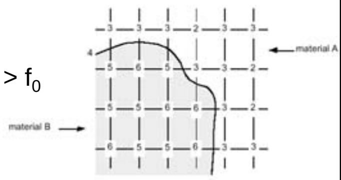
Iso-Surfaces

- Intermediate representation
- Aspects:
  - Preconditions:
    - expressive Iso-value, Iso-value separates materials
    - Interest: in transitions
  - Very selective (binary selection / omission)
  - Uses traditional hardware
  - Shading ⇒ 3D-impression!



Volume Data ⇔ Iso-Surfaces

- Iso-Surface:
  - Iso-value f_0
  - Separates values > f_0 from values ≤ f_0
  - Often not known →
  - Can only be approximated from samples!
  - Shape / position dependent on type of reconstruction

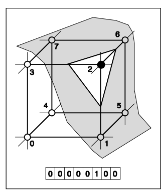


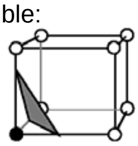
Marching Cubes (MC)

Iso-Surface-Display

Approximation of Iso-Surface

- Approach:
  - Iso-Surface intersects data volume = set of all cells
- Idea:
  - Parts of iso-surface represented on a(n intersected) cell basis
  - As simple as possible: Usage of triangles





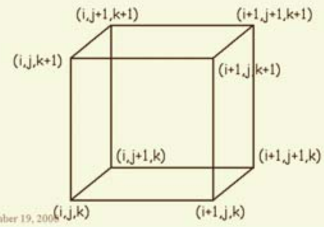
Marching Cubes

- ✓ 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. Go to next cell

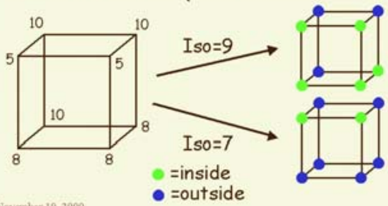
MC 1: Create a Cube

- ✓ Consider a Cube defined by eight data values:



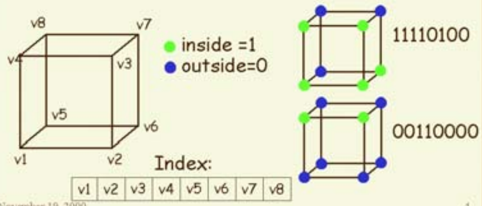
MC 2: Classify Each Voxel

- ✓ Classify each voxel according to whether it lies outside the surface (value > iso-surface value) inside the surface (value <= iso-surface value)



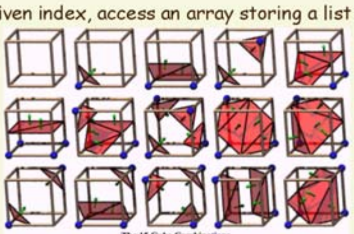
MC 3: Build An Index

- ✓ Use the binary labeling of each voxel to create an index



MC 4: Lookup Edge List

- ✓ For a given index, access an array storing a list of edges

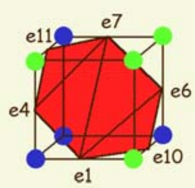


The 15 Cube Combinations

- ✓ all 256 cases can be derived from 15 base cases

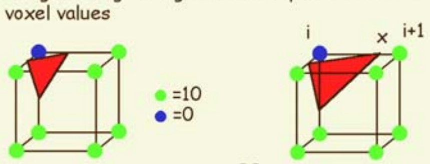
MC 5: Example

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



MC 6: Interp. Triangle Vertex

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

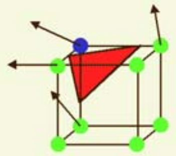


$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$

MC 7: Compute Normals

- Calculate the normal at each cube vertex

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

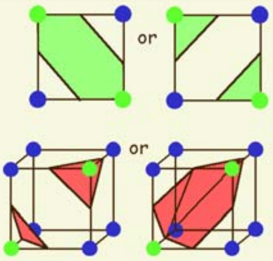


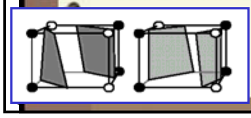
$\vec{N} = \frac{\vec{G}}{|\vec{G}|}$

- Use linear interpolation to compute the polygon vertex normal

MC 8: Ambiguous Cases

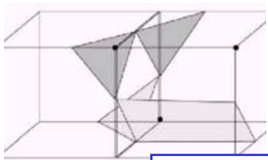
- Ambiguous cases: 3, 6, 7, 10, 12, 13
- Adjacent vertices: different states
- Diagonal vertices: same state
- Resolution: decide for one case

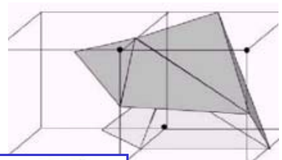




Danger: Holes!

- Wrong vs. correct classification!





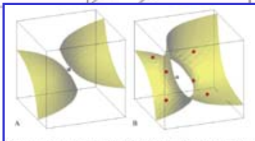
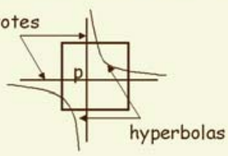


Figure 4: Two internal configurations for the Marching Cubes configuration 5

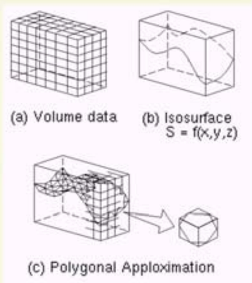
MC 9: Asymptotic Decider

- Assume bilinear interpolation within a face
- hence iso-surface is a hyperbola
- compute the point p where the asymptotes meet
- sign of S(p) decides the connectedness



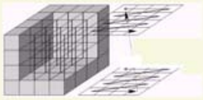
Marching Cubes - Summary 1

- 256 Cases
- reduce to 15 cases by symmetry
- Complementary cases - (swap in- and outside)
- Ambiguity resides in cases 3, 6, 7, 10, 12, 13
- Causes holes if arbitrary choices are made.



Marching Cubes - Summary 2

- Up to 4 triangles per cube
- Dataset of 512^3 voxels can result in several million triangles (many Mbytes!!!)
- Iso-surface does not represent an object!!!
- No depth information
- Semi-transparent representation --> sorting
- Optimization:
  - Reuse intermediate results
  - Prevent vertex replication
  - Mesh simplification



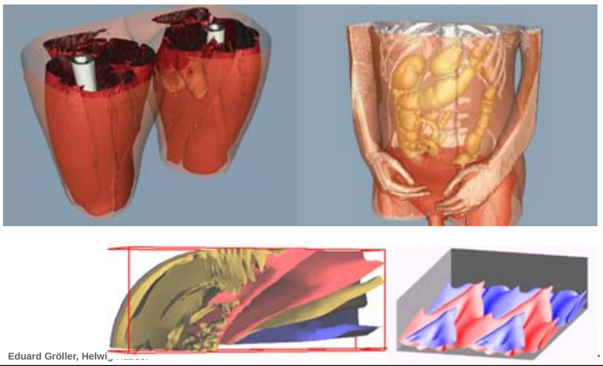
MC Examples

1 Iso-surface

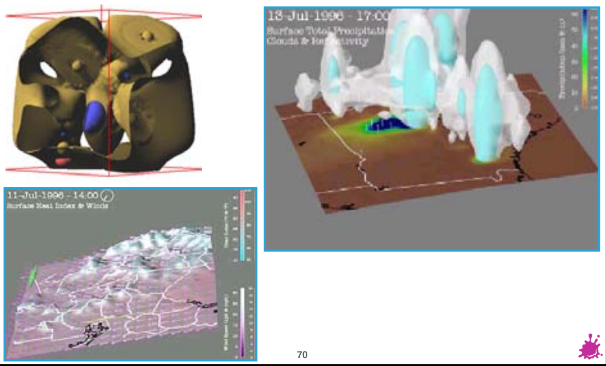
3 Iso-surfaces

2 Iso-surfaces

Further Examples



Even Further Examples



Conclusion
Volume Visualization

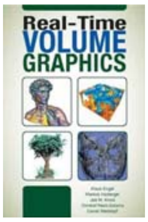
General Remarks

Surface vs. Volume Rendering

- Surface Rendering:
  - Indirect representation / display
  - Conveys surface impression
  - Hardware supported rendering (fast?!)
  - Iso-value-definition
- Volume Rendering:
  - Direct representation / display
  - Conveys volume impression
  - Often realized in software (slow?!)
  - Transfer functions

Literature, References

- Marc Levoy: "**Display of Surfaces from Volume Data**" in *IEEE Computer Graphics & Applications*, Vol. 8, No. 3, June 1988
  - Nelson Max: "**Optical Models for Direct Volume Rendering**" in *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 2, June 1995
  - W. Lorensen & H. Cline: "**Marching Cubes: A High Resolution 3D Surface Construction Algorithm**" in *Proceedings of ACM SIGRRAPH '87 = Computer Graphics*, Vol. 21, No. 24, July 1987
- K. Engel, M. Hadwiger et al. "**Real-Time Volume Graphics**" http://www.real-time-volume-graphics.org/



Acknowledgments

- For material for this lecture unit
  - Roberto Scopigno, Claudio Montani (CNR, Pisa)
  - Hans-Georg Pagendarm (DLR, Göttingen)
  - Michael Meißner (GRIS, Tübingen)
  - Torsten Möller
  - Gordon Kindlmann
  - Joe Kniss
  - Nelson Max (LLNL), Marc Levoy (Stanford)
  - Lloyd Treinish (IBM)
  - Torsten Möller
  - Roger Crawfis (Ohio State Univ.)
  - Hanspeter Pfister (MERL)
  - Dirk Bartz
  - Markus Hadwiger