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

- Introduction:
  - VolVis = visualization of volume data
    - Mapping 3D→2D
    - Projection (e.g., MIP), slicing, vol. rendering, …
  - Volume data =
    - 3D×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

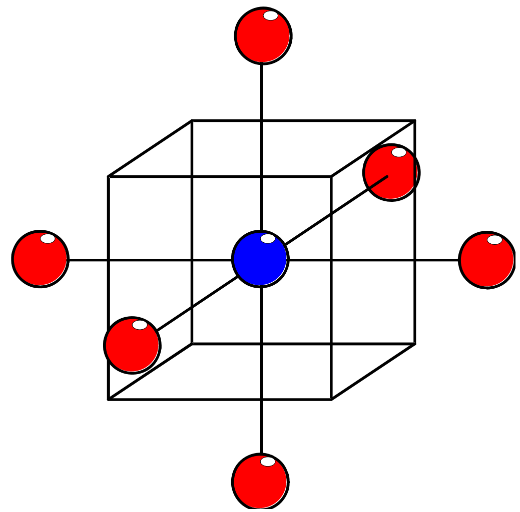
- How are volume data organized?
  - Cartesian resp. regular grid:
    - CT/MR: often dx=dy<dz, e.g. 135 slices (z) á 512^2 values (as x & y pixels in a slice)
    - **Data enhancement**: iso-stack-calculation = Interpolation of additional slices, so that dx=dy=dz $\Rightarrow$ 512^3 Voxel
    - Data: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

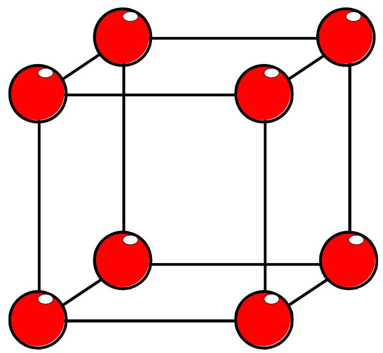
VolVis – Challenges

- 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!

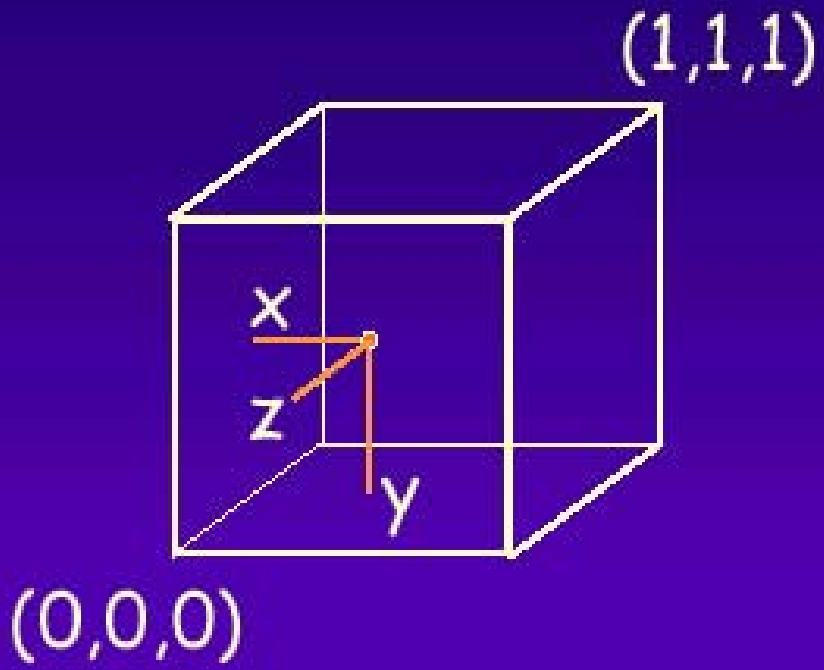
Voxels vs. Cells

- 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



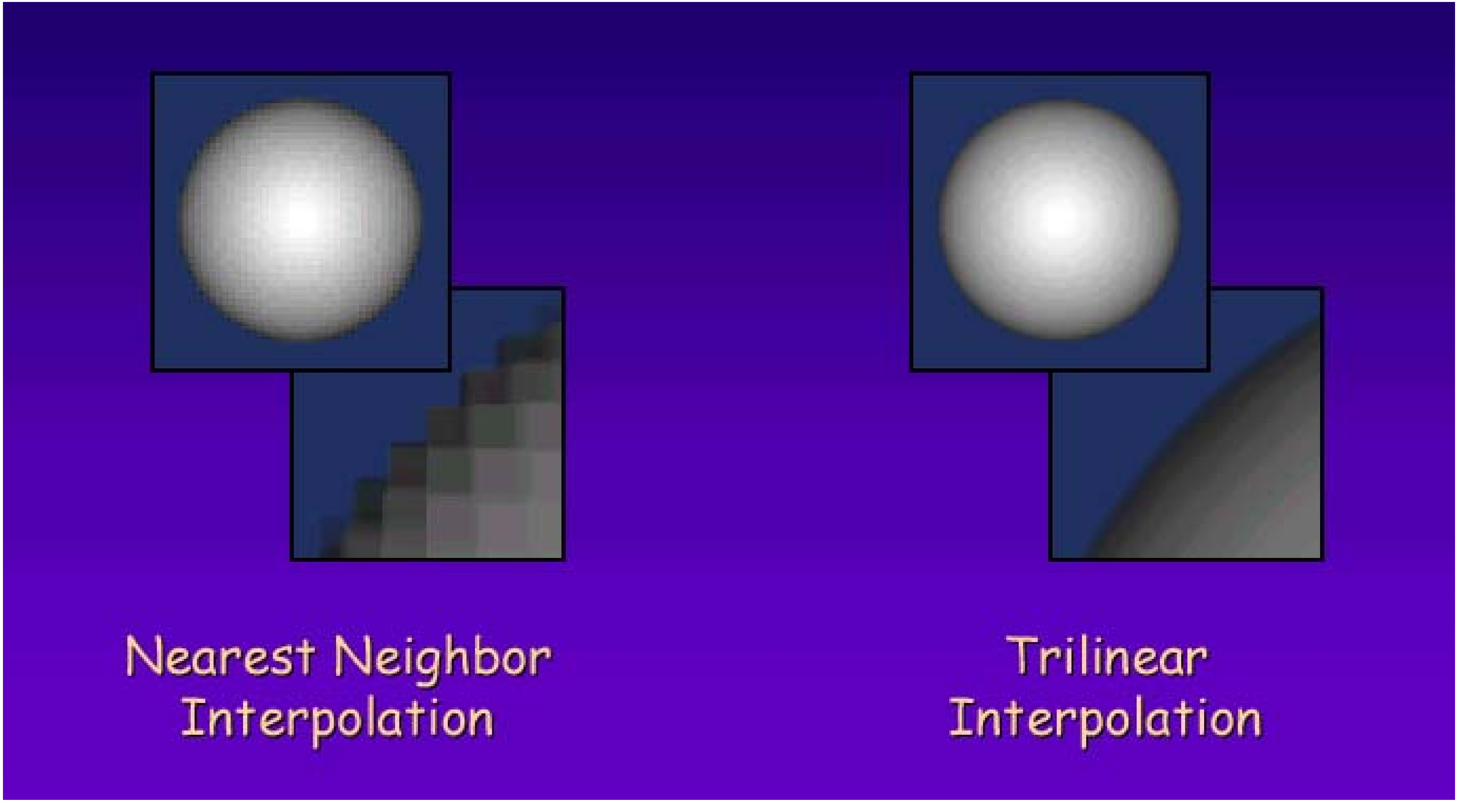
$$v = S(\mathrm{rnd}(x), \mathrm{rnd}(y), \mathrm{rnd}(z))$$

Nearest Neighbor

$$\begin{aligned} v = \; & (1-x)(1-y)(1-z)S(0,0,0) + \\ & (x)(1-y)(1-z)S(1,0,0) + \\ & (1-x)(y)(1-z)S(0,1,0) + \\ & (x)(y)(1-z)S(1,1,0) + \\ & (1-x)(1-y)(z)S(0,0,1) + \\ & (x)(1-y)(z)S(1,0,1) + \\ & (1-x)(y)(z)S(0,1,1) + \\ & (x)(y)(z)S(1,1,1) \end{aligned}$$

Trilinear

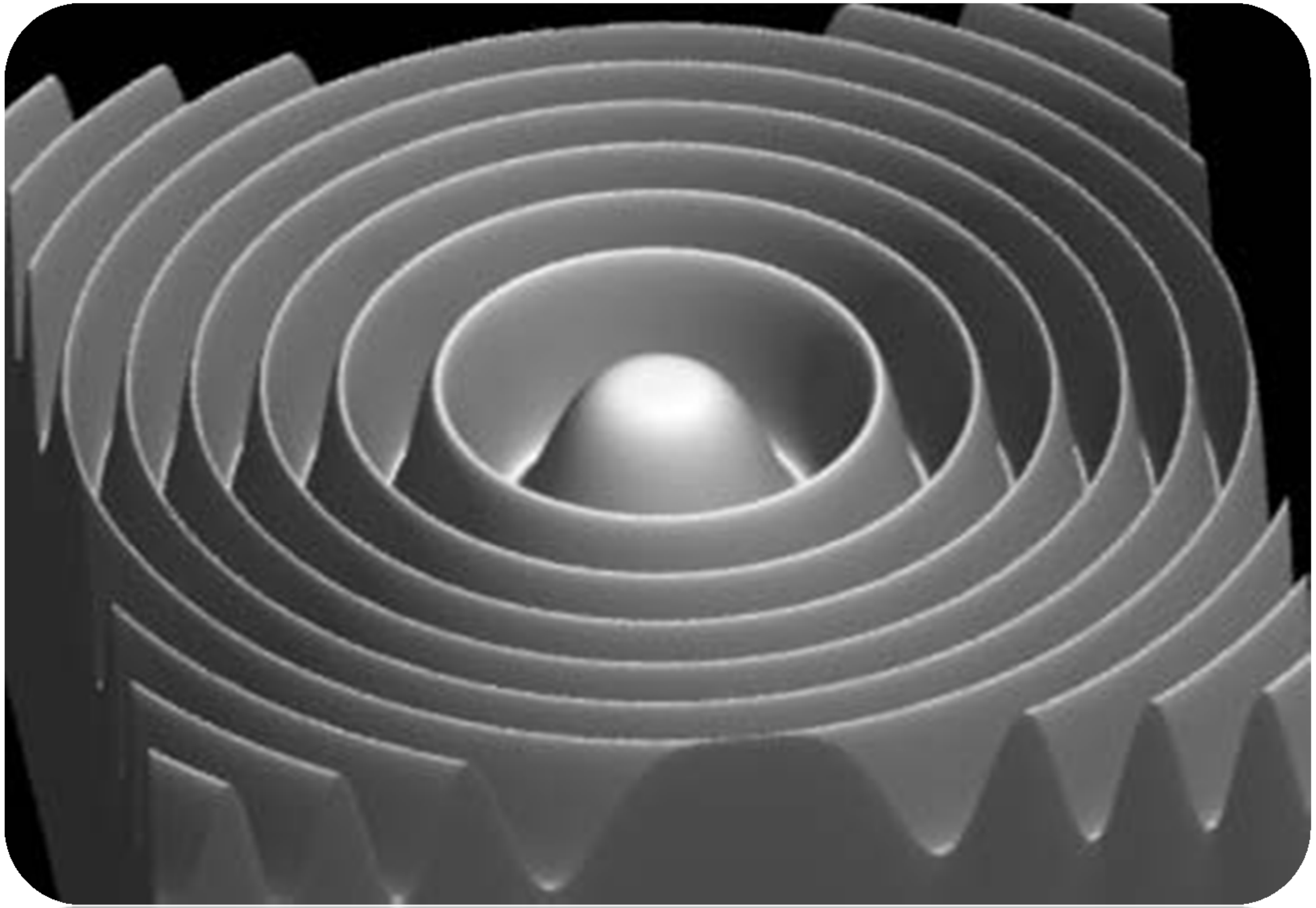
Interpolation – Results



Higher-Order Reconstruction (1)

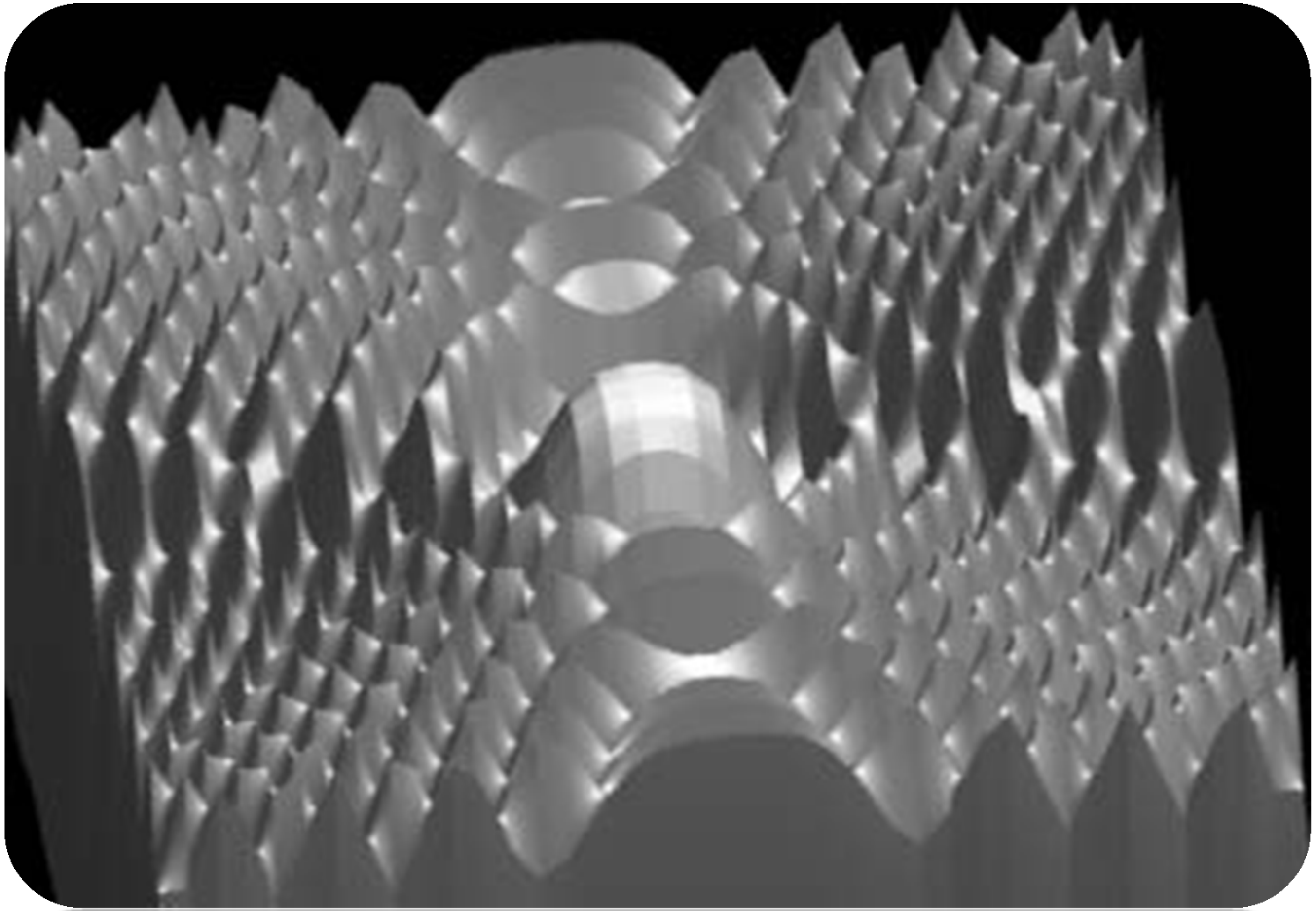
- If very high quality is needed, more complex reconstruction filters may be required
  - Marschner-Lobb function is a common test signal to evaluate the quality of reconstruction filters [Marschner and Lobb 1994]
  - The signal has a high amount of its energy near its Nyquist frequency
  - Makes it a very demanding test for accurate reconstruction

- **Analytical evaluation** of the Marschner-Lobb test signal



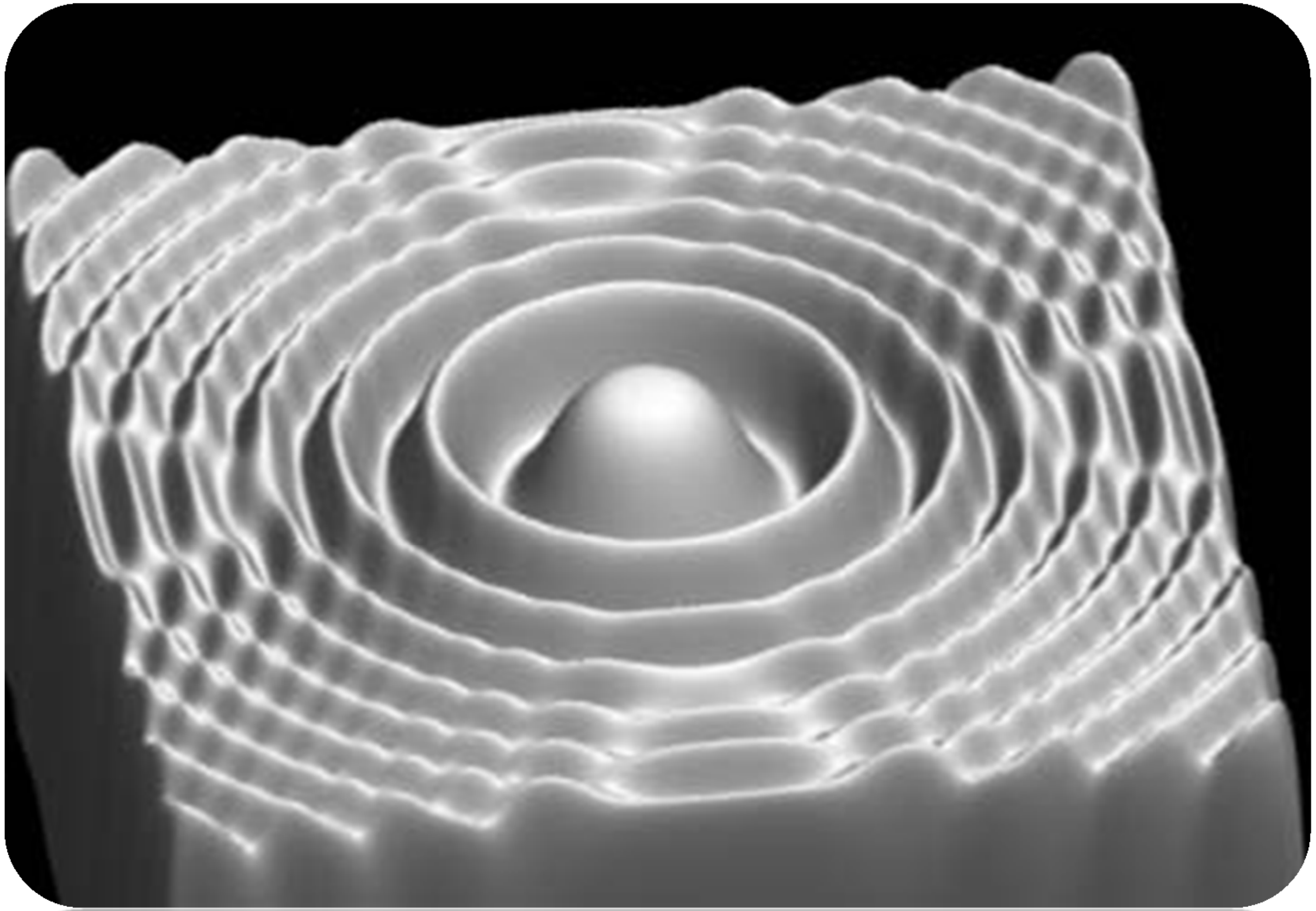
Higher-Order Reconstruction (3)

- **Trilinear** reconstruction of Marschner-Lobb test signal

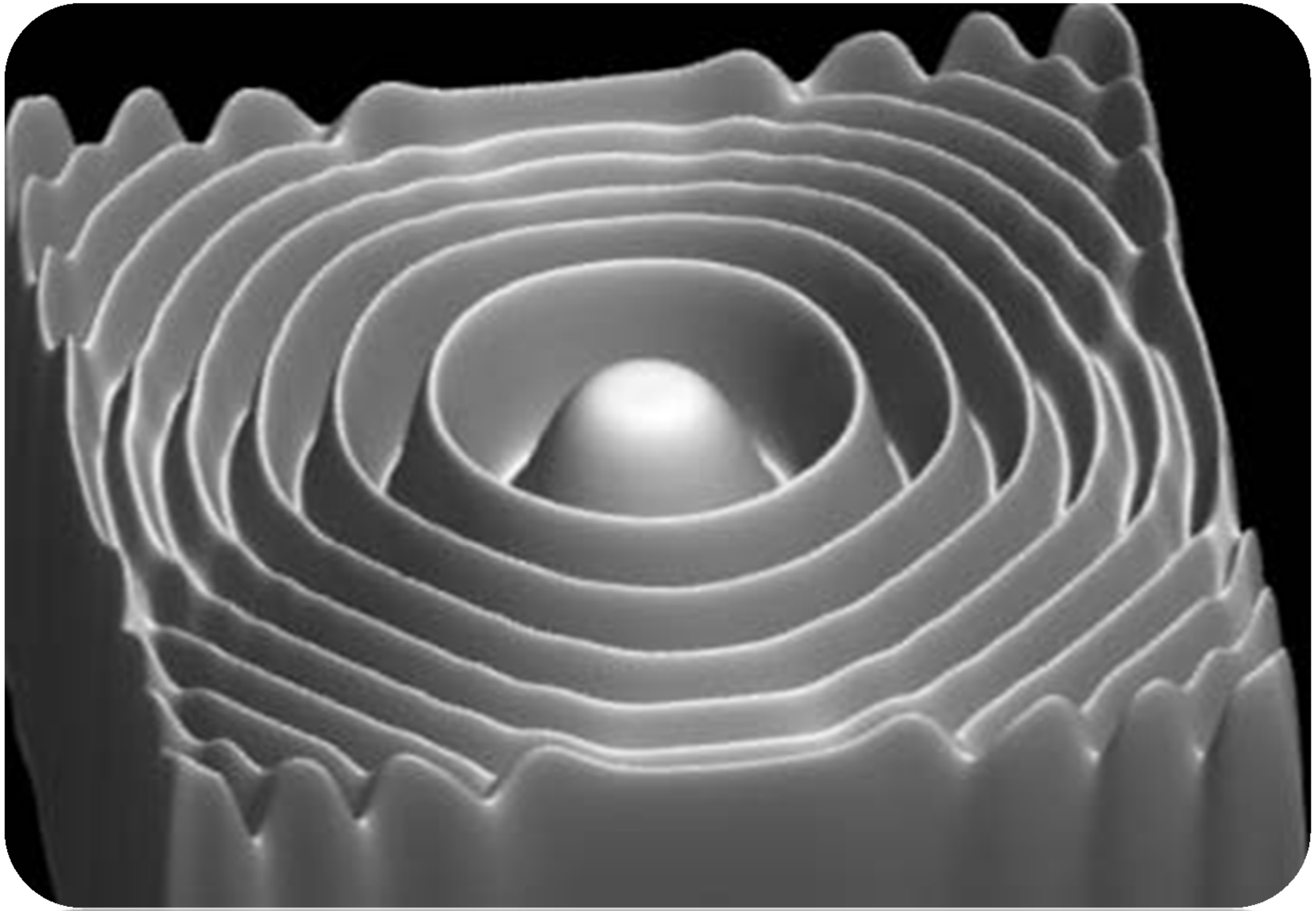


Higher-Order Reconstruction (4)

- **B-Spline** reconstruction of Marschner-Lobb test signal

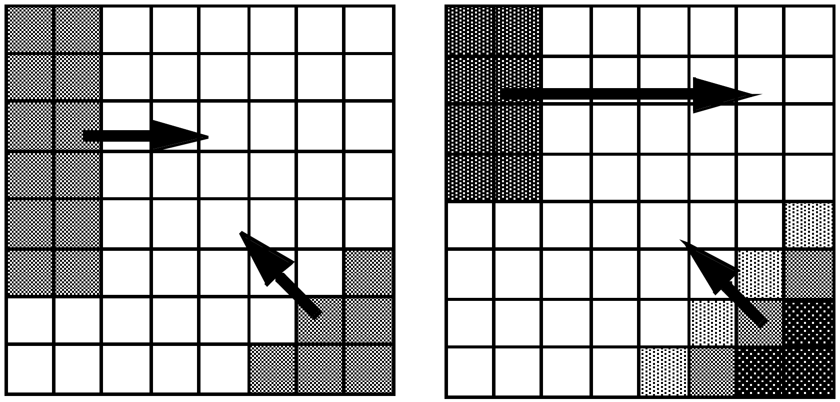


- **Windowed sinc** reconstruction of Marschner-Lobb test signal



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

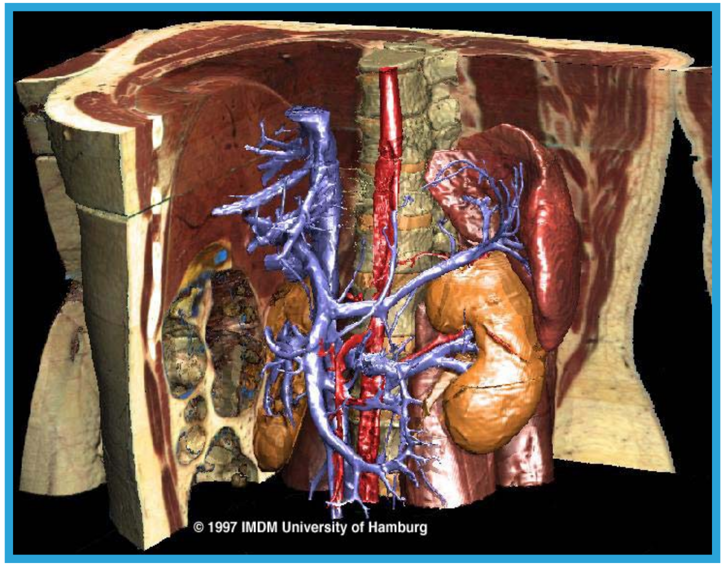
- Gradient $\nabla f = (\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)$
- $\nabla f|_{x_0}$ normal vector to iso-surface $f(x_0)=f_0$
- Central difference in x-, y- & z-direction (in voxel):

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

- 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: $\nabla f\,(x+0.5)=f(x+1)-f(x)$

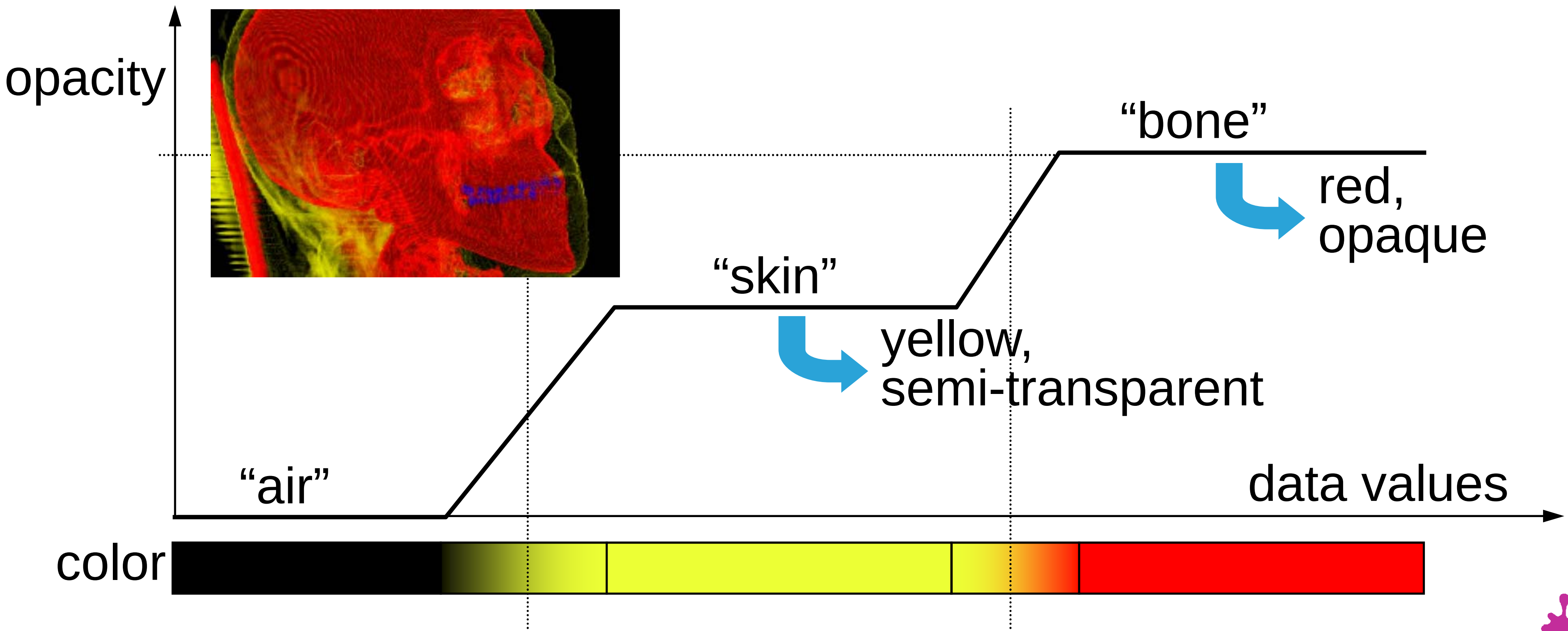
Classification

- 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)



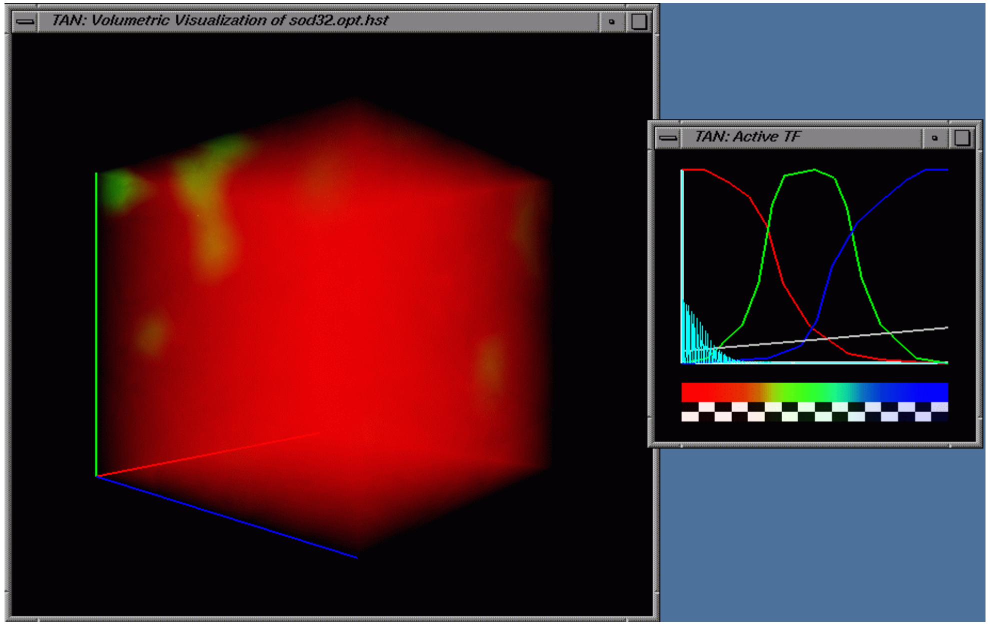
Transfer Functions (TF)

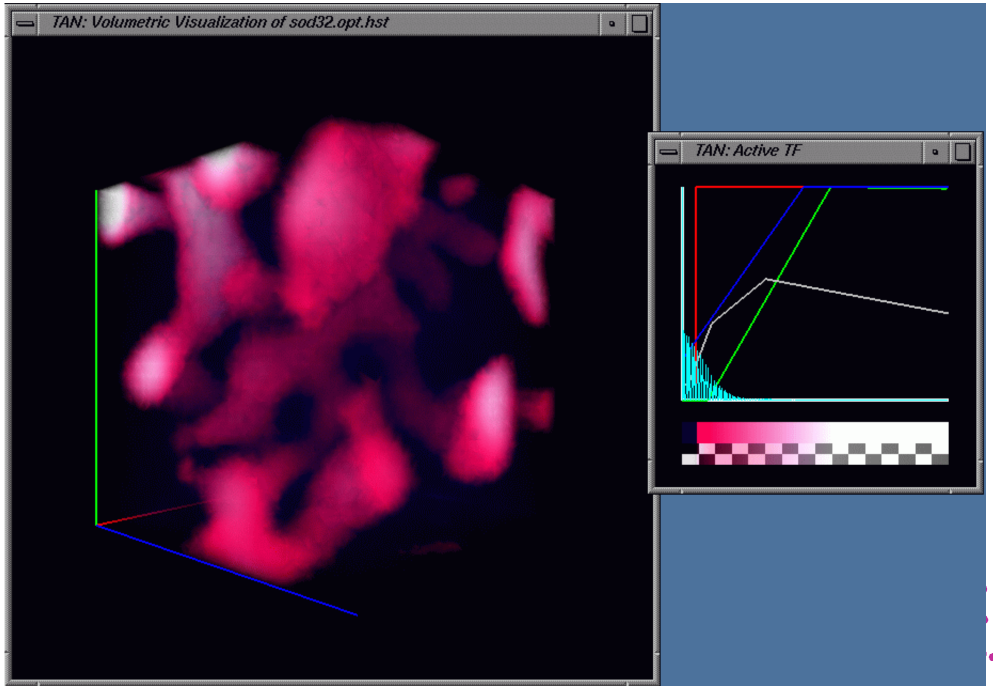
- Mapping data → ”renderable quantities”:
  - 1.) data→color (f(i)→C(i))
  - 2.) data→opacity (non-transparency) (f(i)→α(i))



Different Transfer Functions

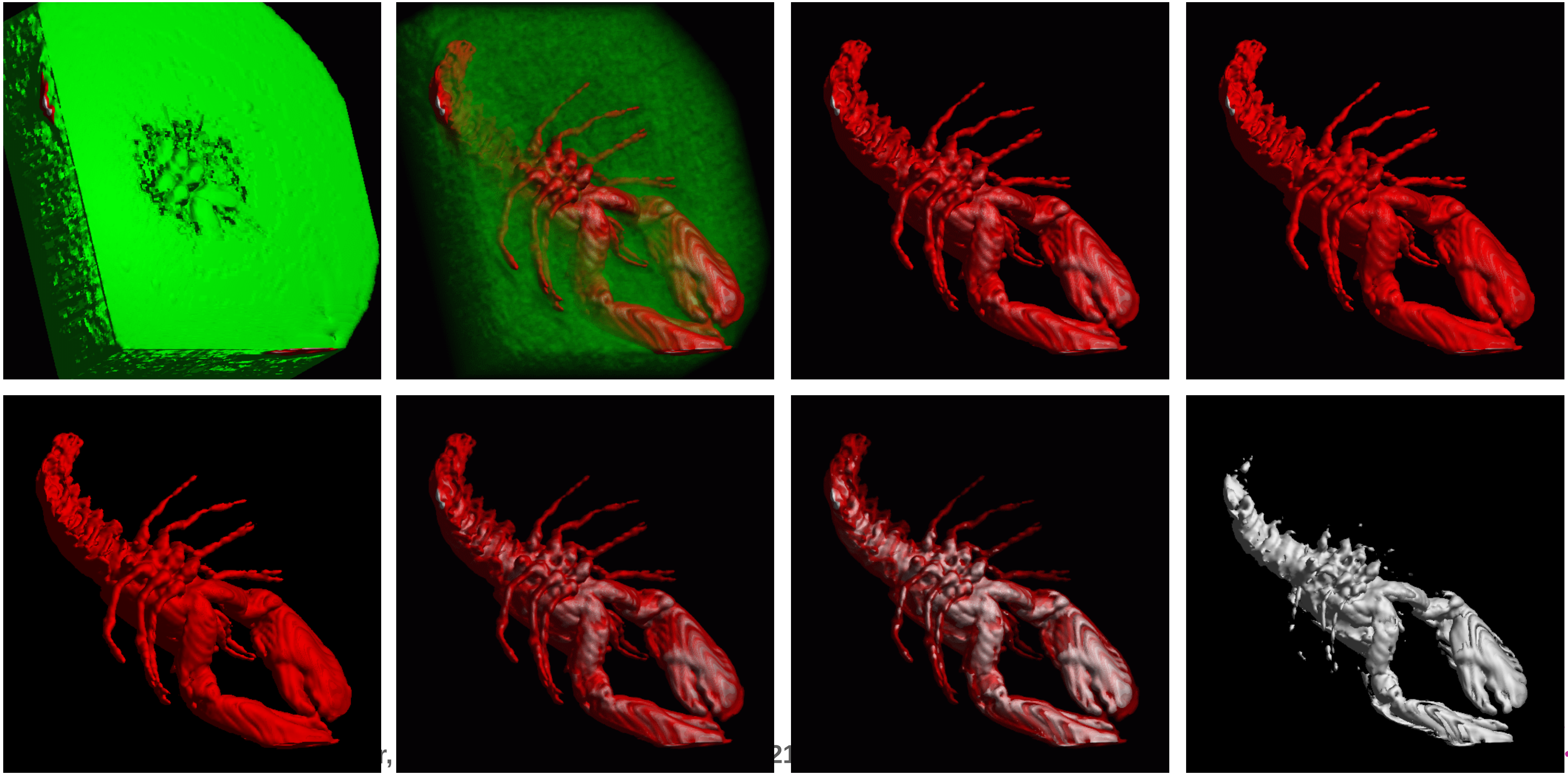
- Image results:
  - Strong dependence on transfer functions
  - Non-trivial specification
  - Limited segmentation possibilities



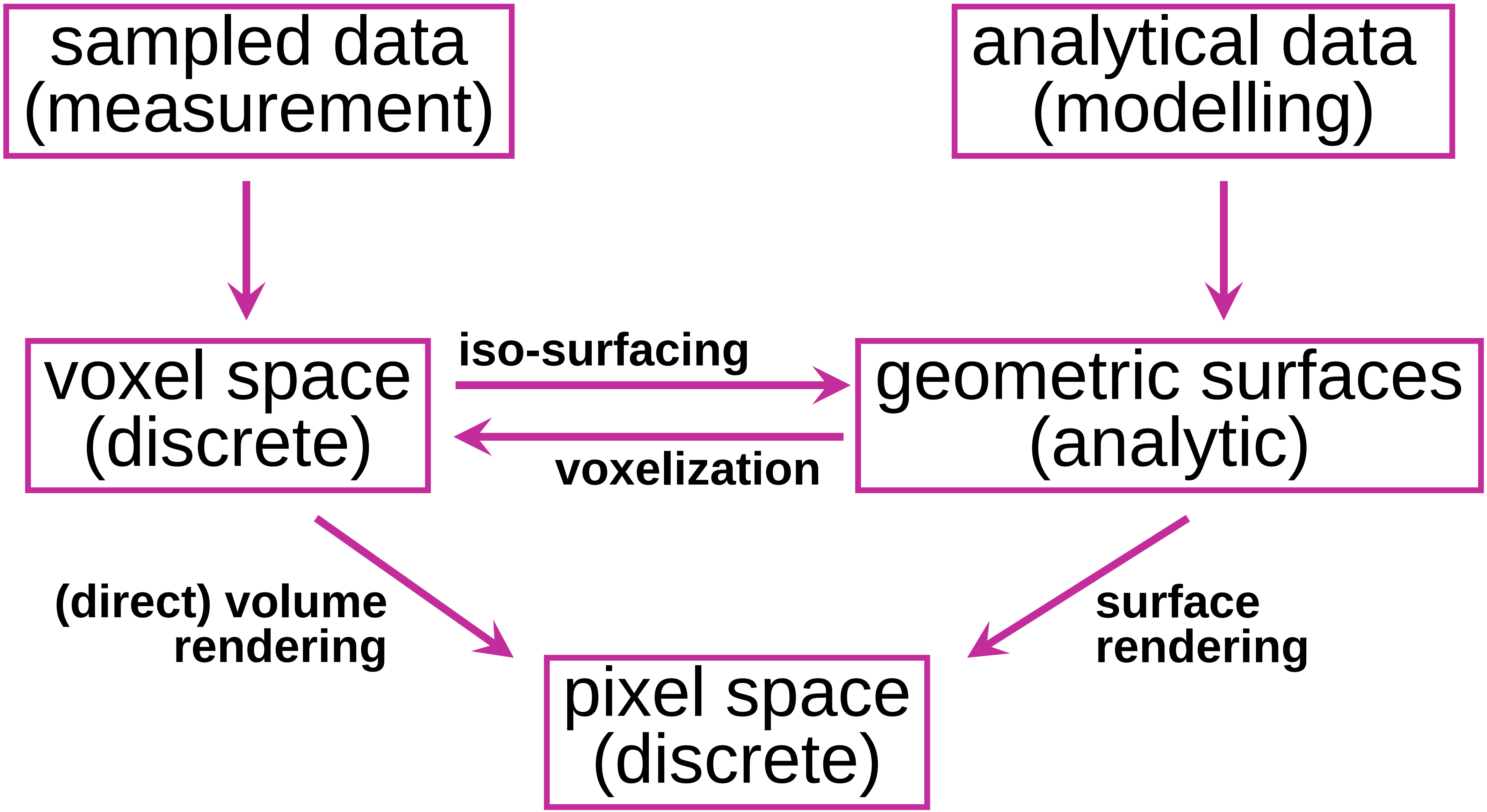


Lobster – Different Transfer Functions

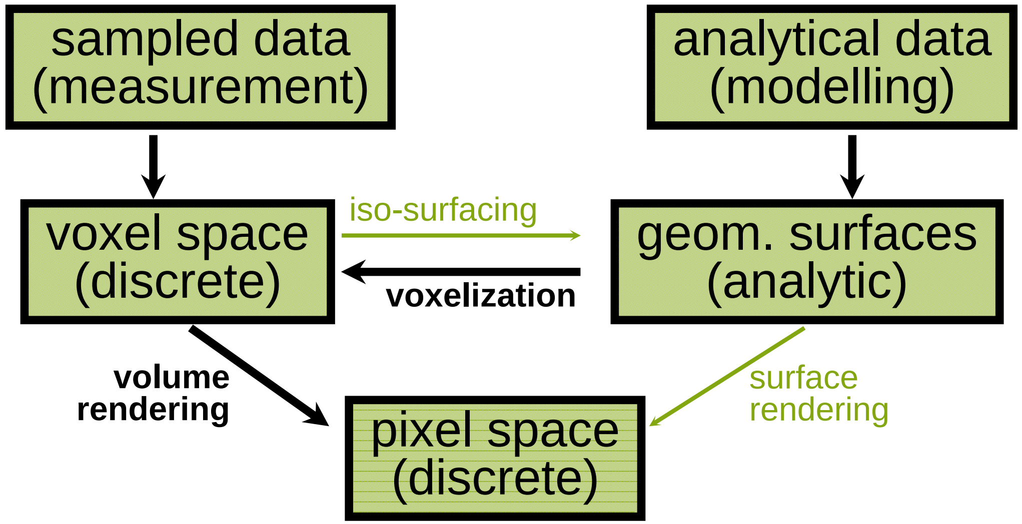
- Three objects: media, shell, flesh

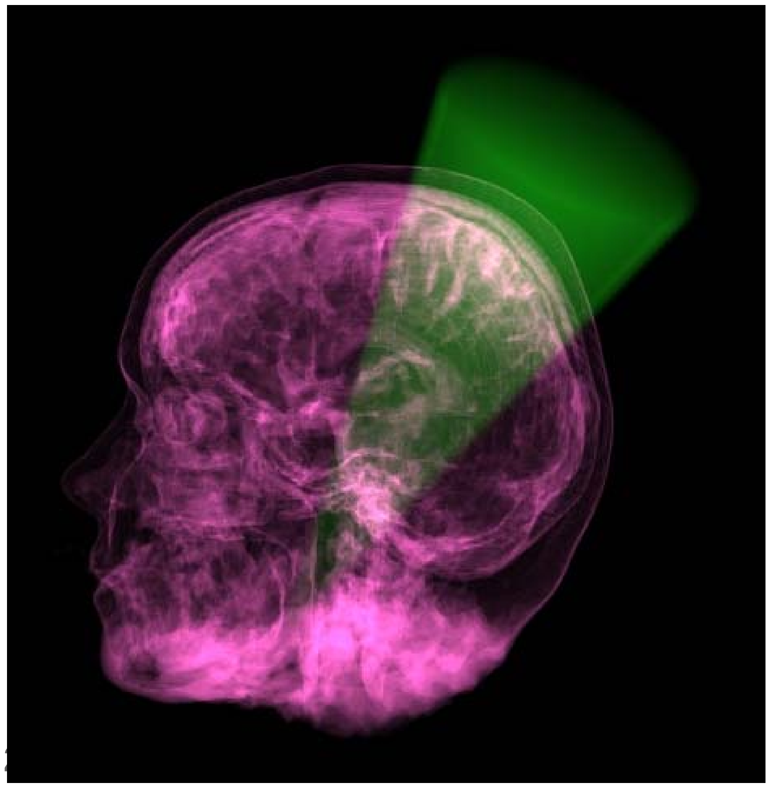


Concepts and Terms



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

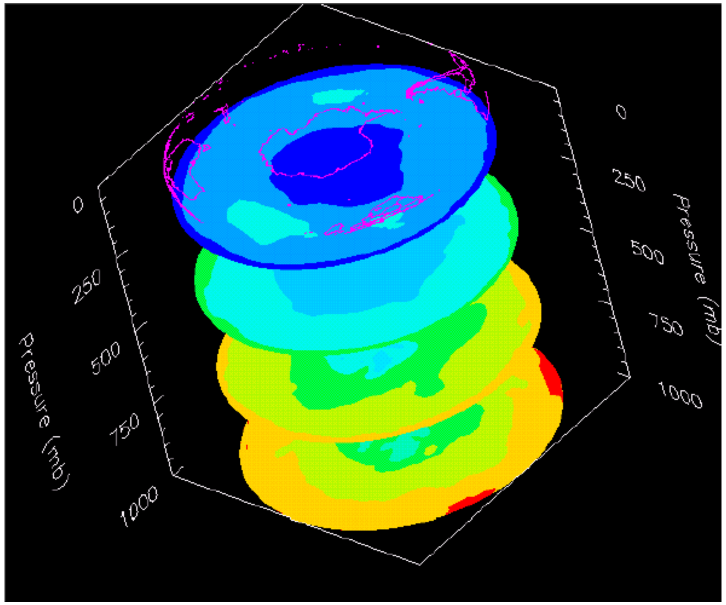


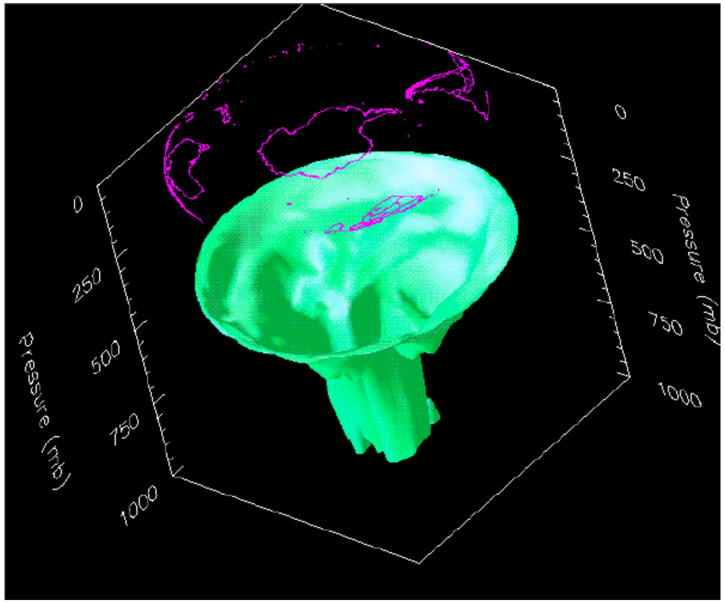


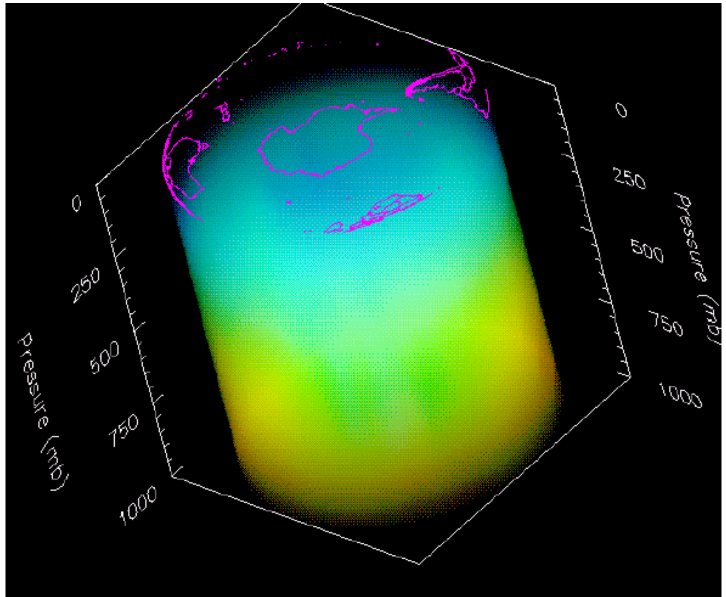
Slice vs. Surface vs. Volume Rendering

- 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

- 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

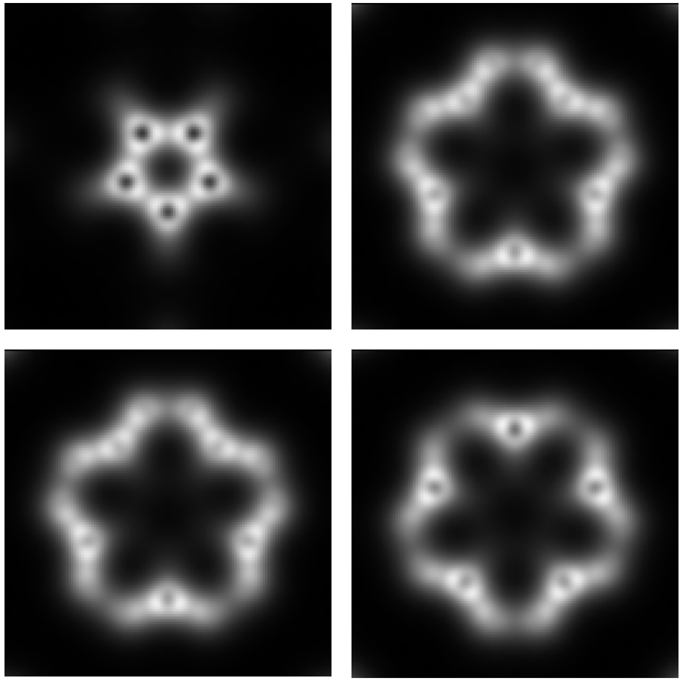
- Simple methods:
  - Slicing, MPR (multi-planar reconstruction)
- Direct volume visualization:
  - Ray casting
  - Shear-warp factorization
  - Splatting
  - 3D texture mapping
  - Fourier volume rendering
- Surface-fitting methods:
  - Marching cubes (marching tetrahedra)

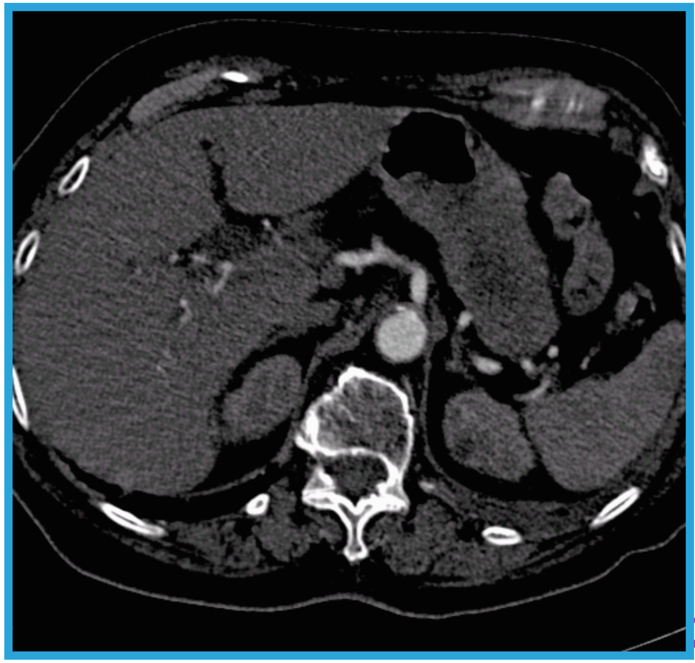
Simple Methods

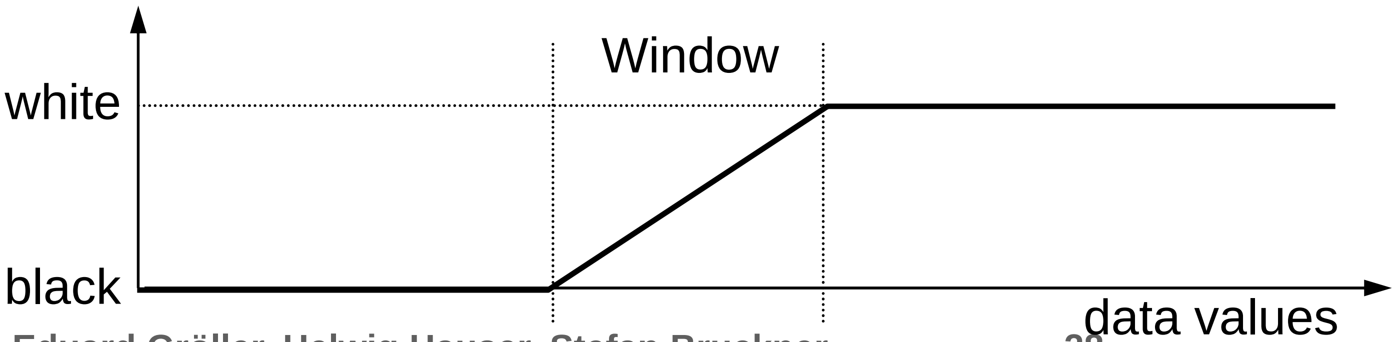
Slicing, etc.

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

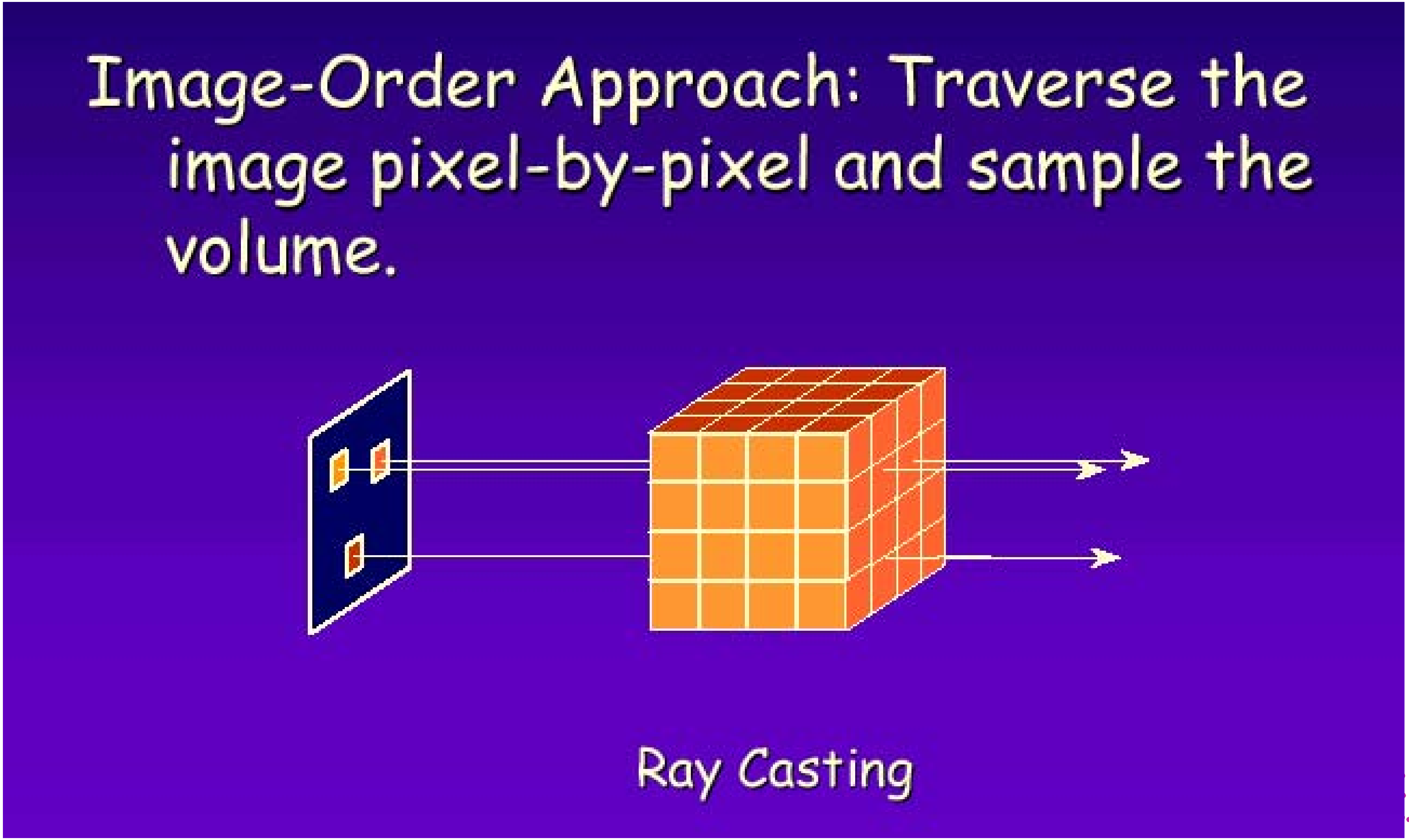
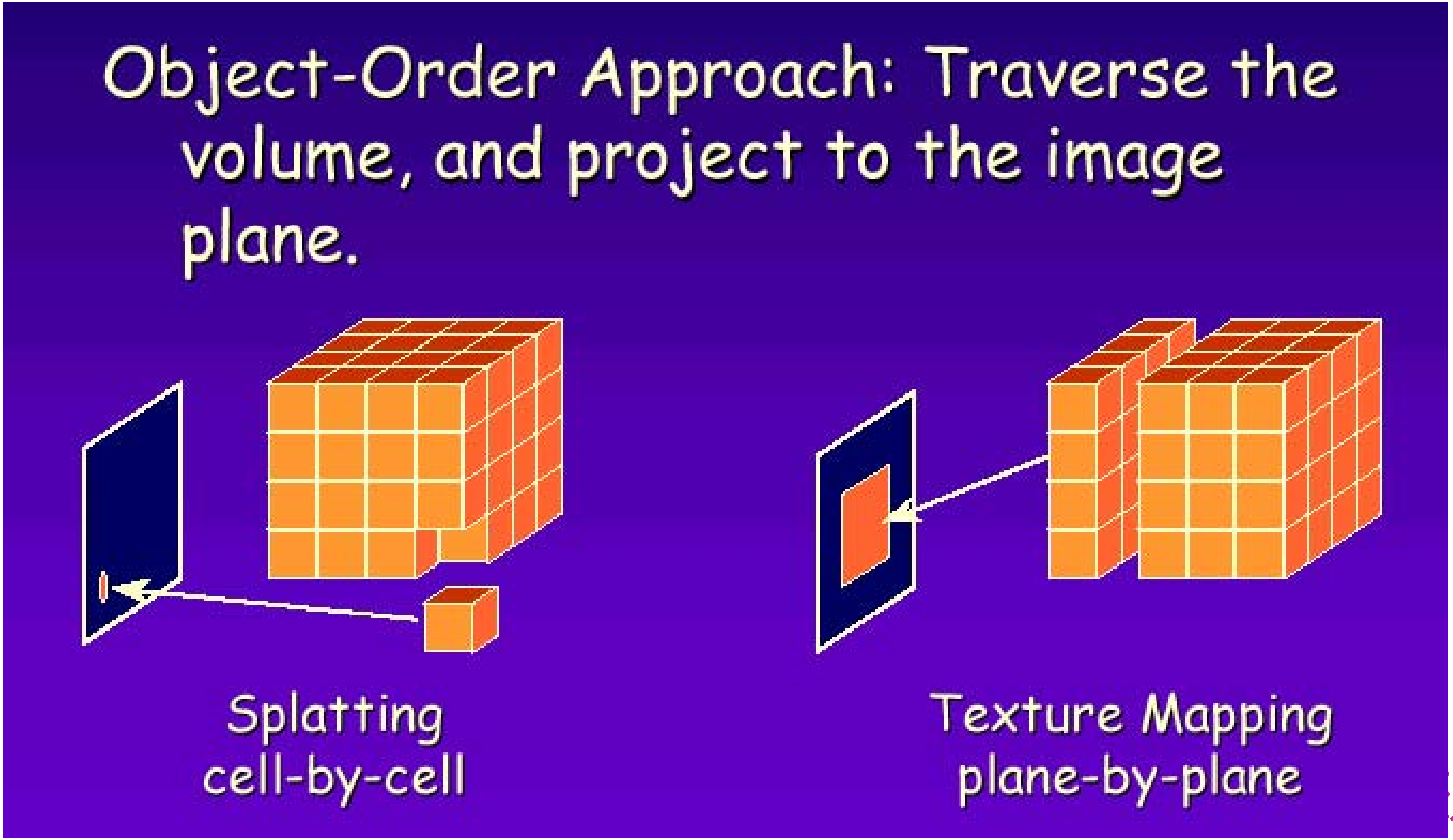


Image-Order vs. Object-Order



Ray Casting

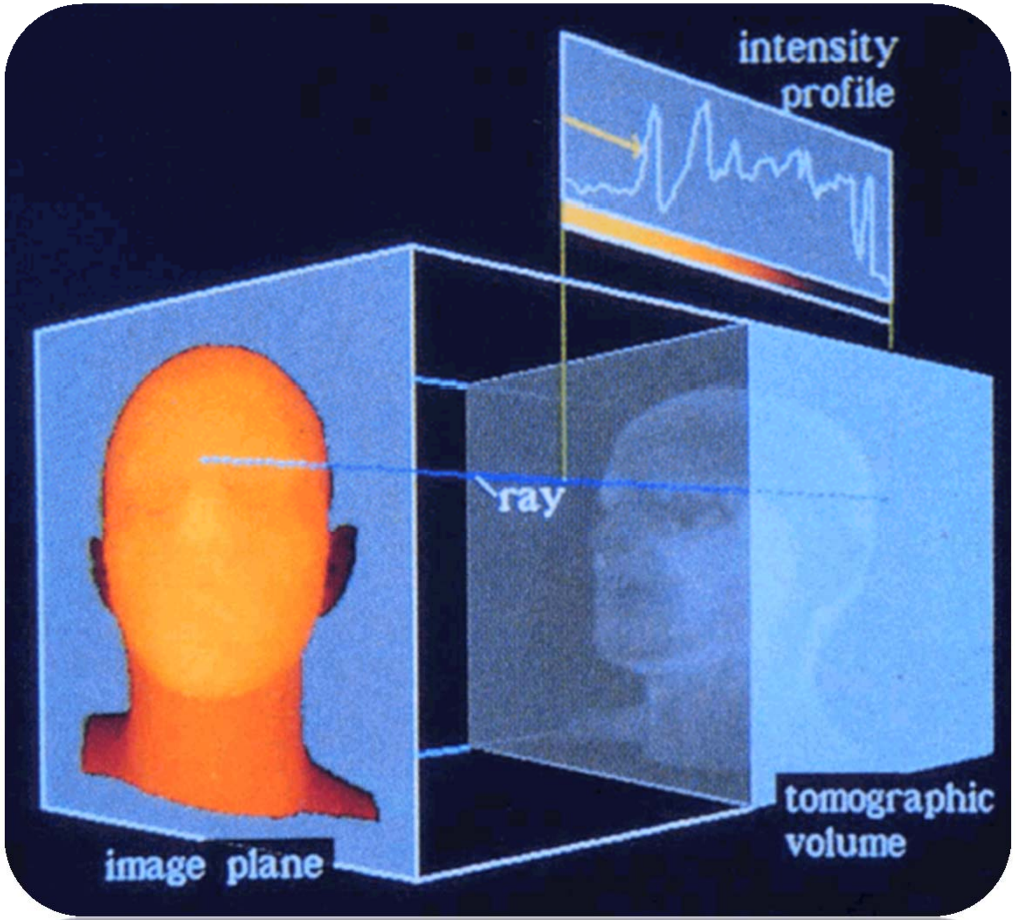
Image-Order Method

Ray Tracing vs. Ray Casting

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

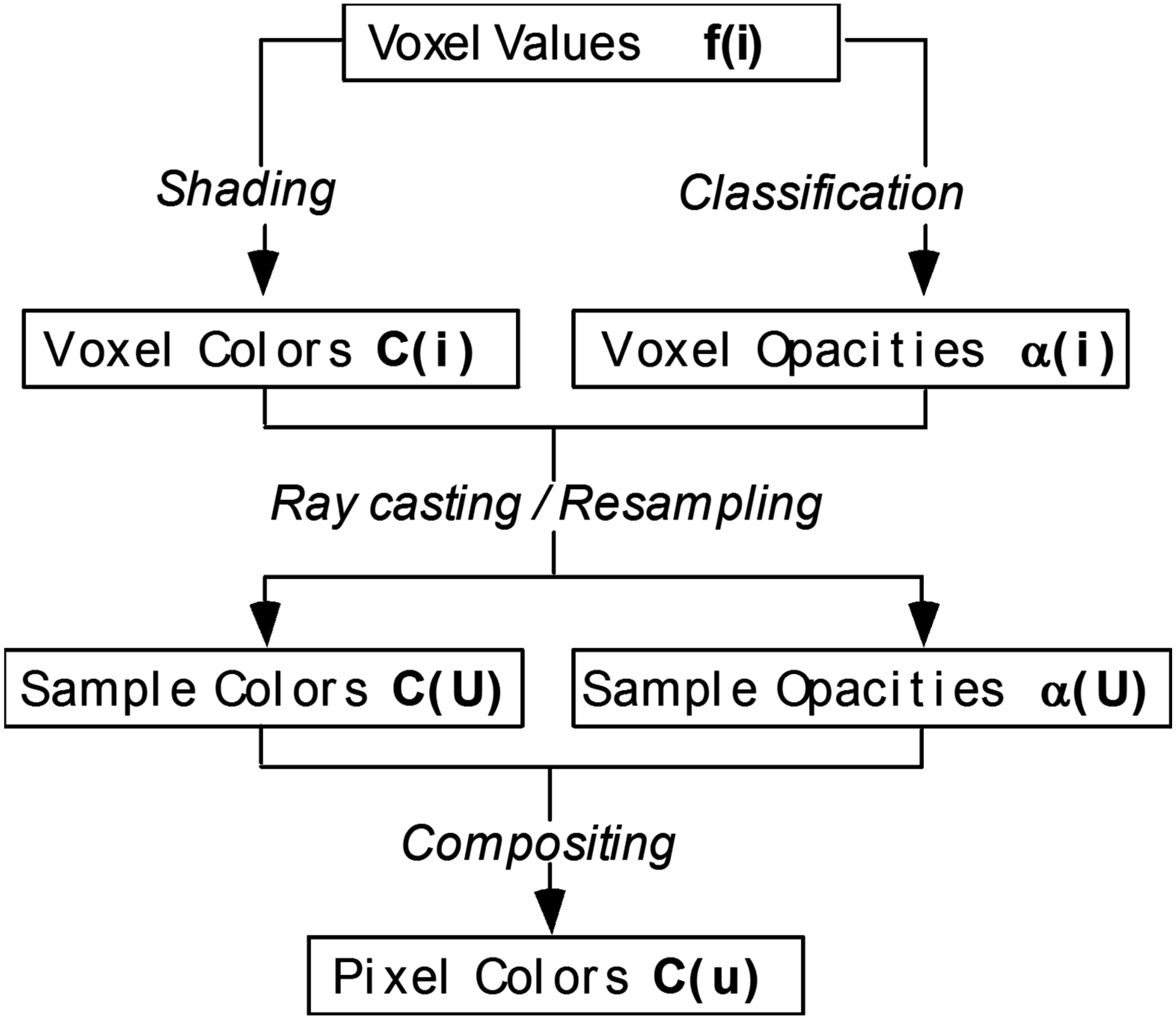
Ray Traversal through Volume Data

- 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)

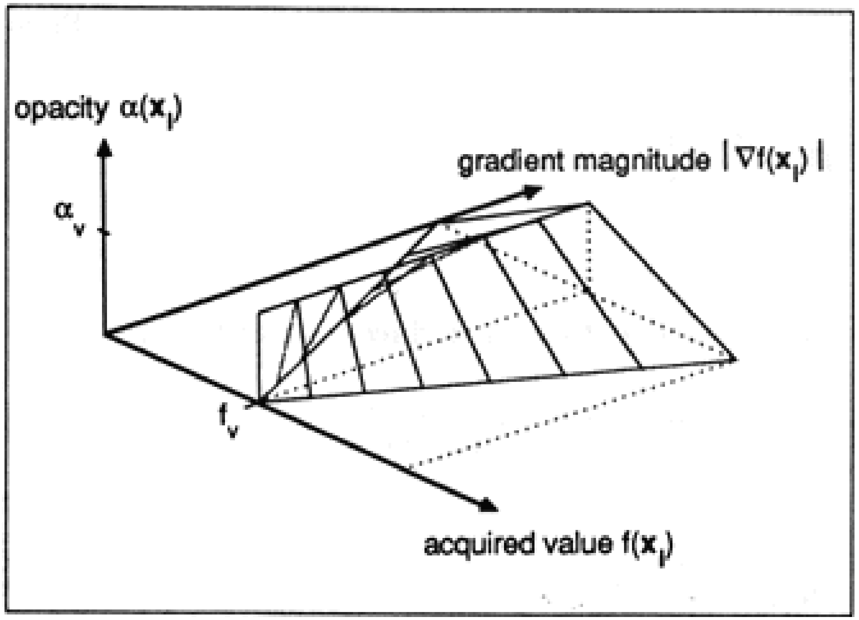


Standard Ray Casting

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

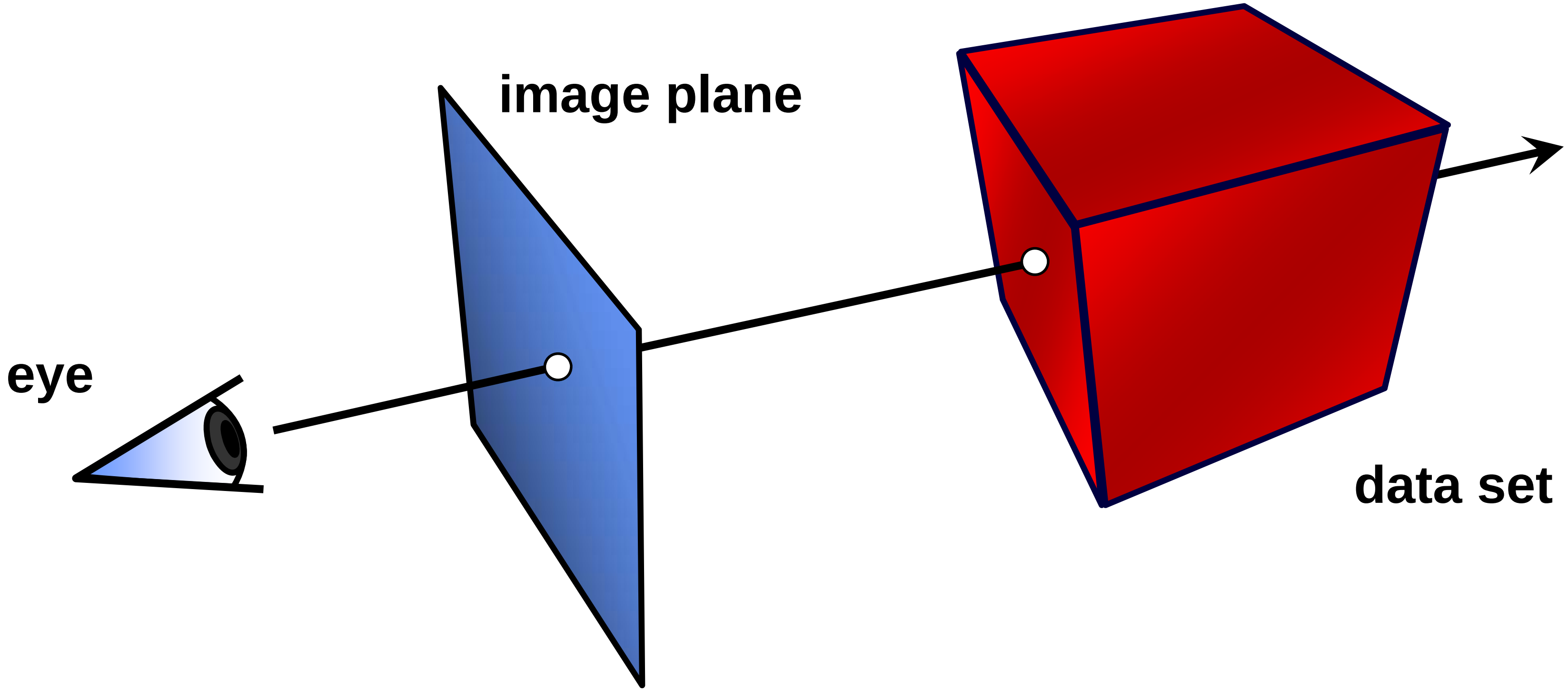


- 1. Step:
  - Shading, $f(i) \rightarrow C(i)$:
    - Apply transfer function
    - diffuse illumination (Phong), gradient ≈ normal
  - Classification, $f(i) \rightarrow \alpha(i)$:
    - Levoy '88, gradient enhanced
    - Emphasizes transitions
  - Nowadays: shading/classification after ray-casting/resampling

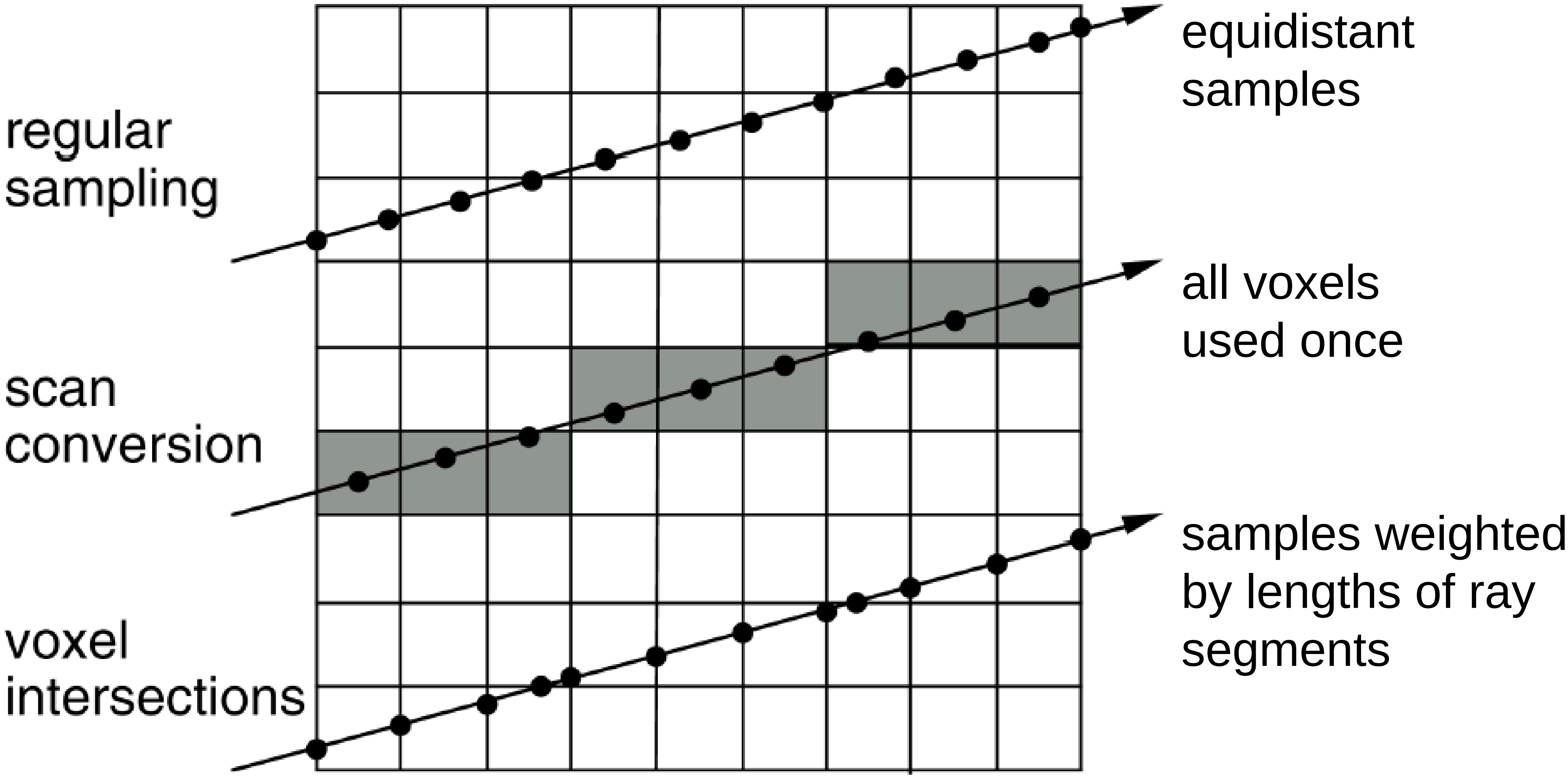


2. Ray Traversal

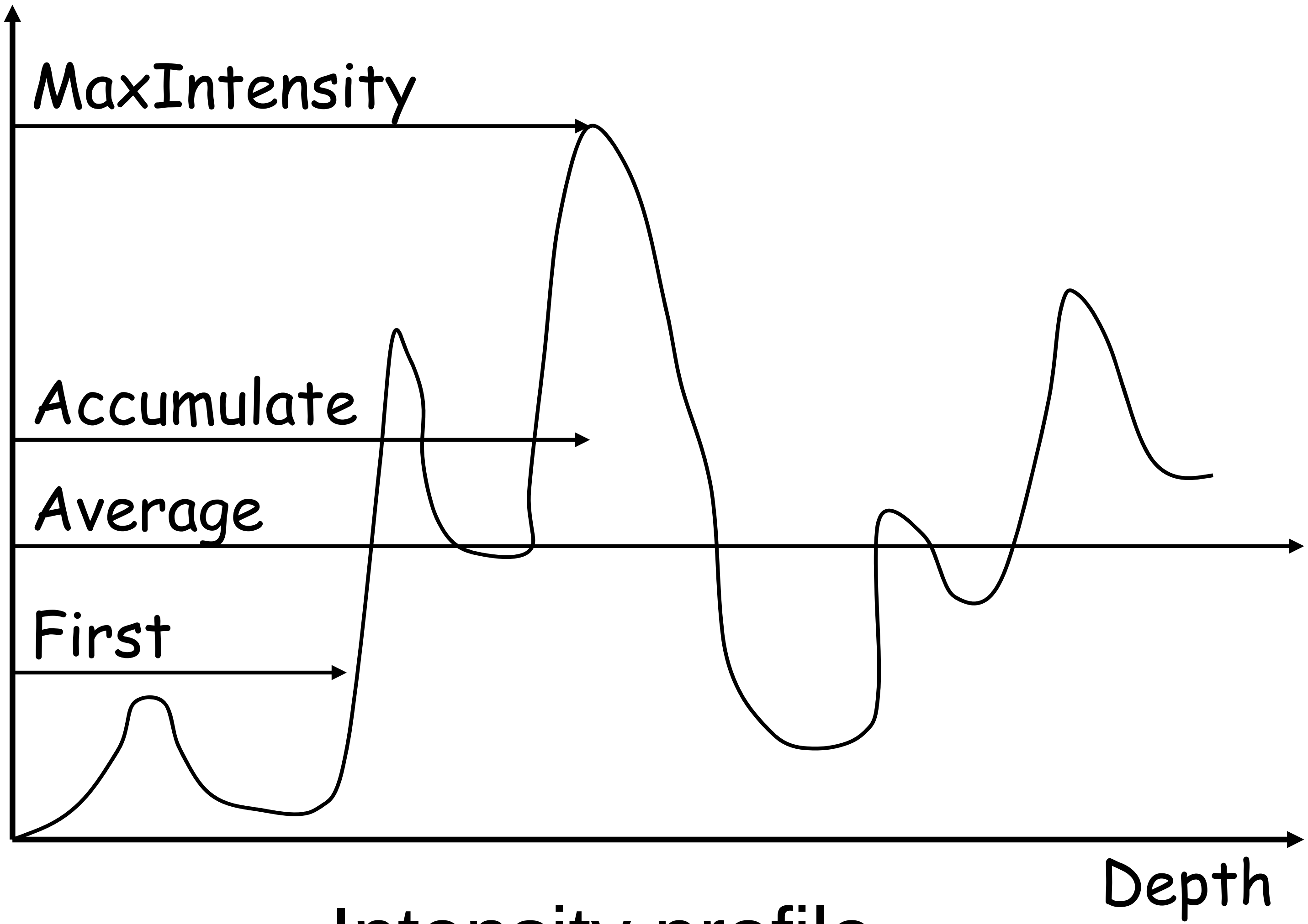
- Cast ray through the volume and perform sampling at discrete positions



2. Ray Traversal – Three Approaches



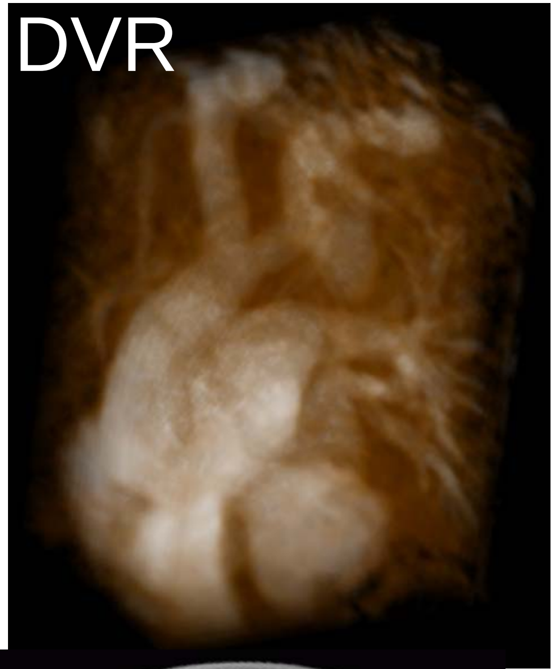
- Overview:
  - MIP ⇒
  - Compositing ⇒
  - X-Ray ⇒
  - First hit ⇒

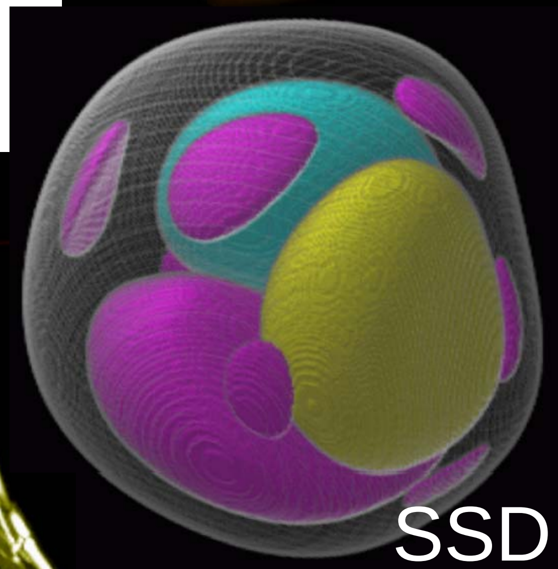


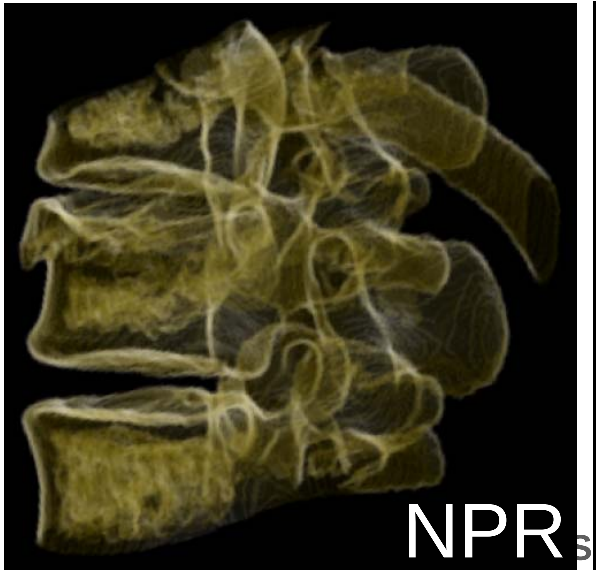
Intensity profile

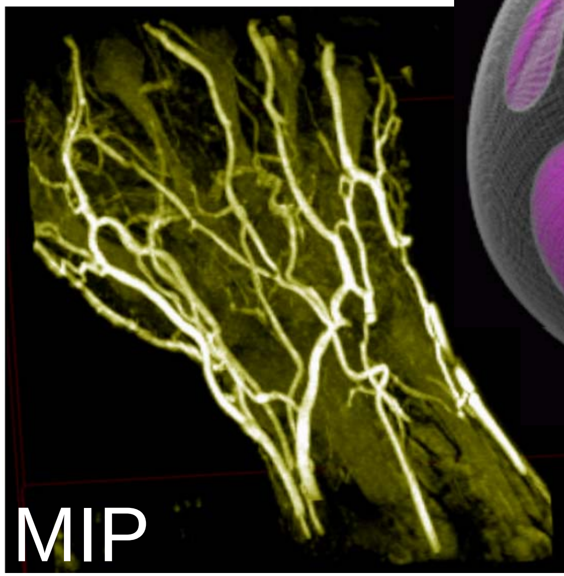
Types of Combinations

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





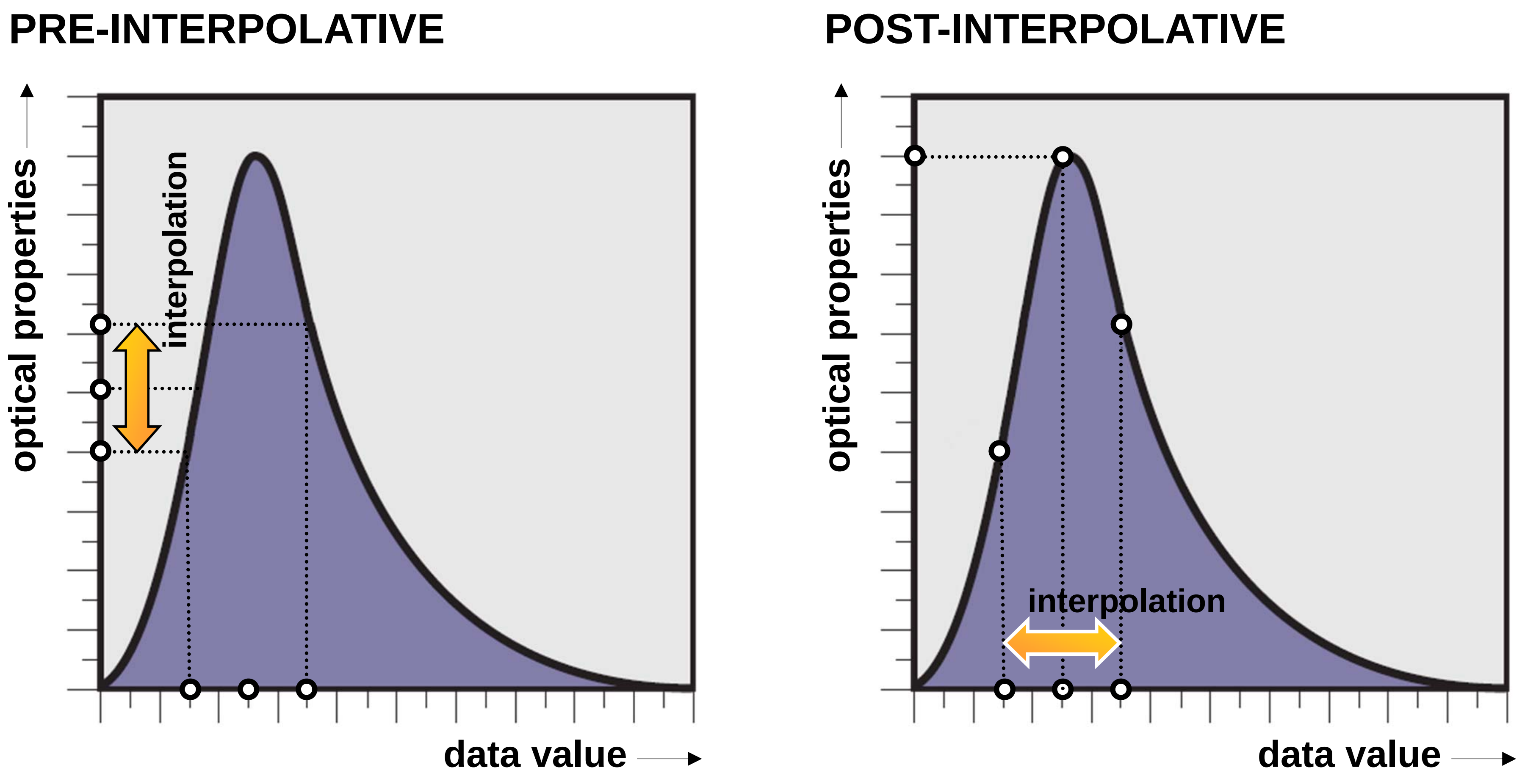




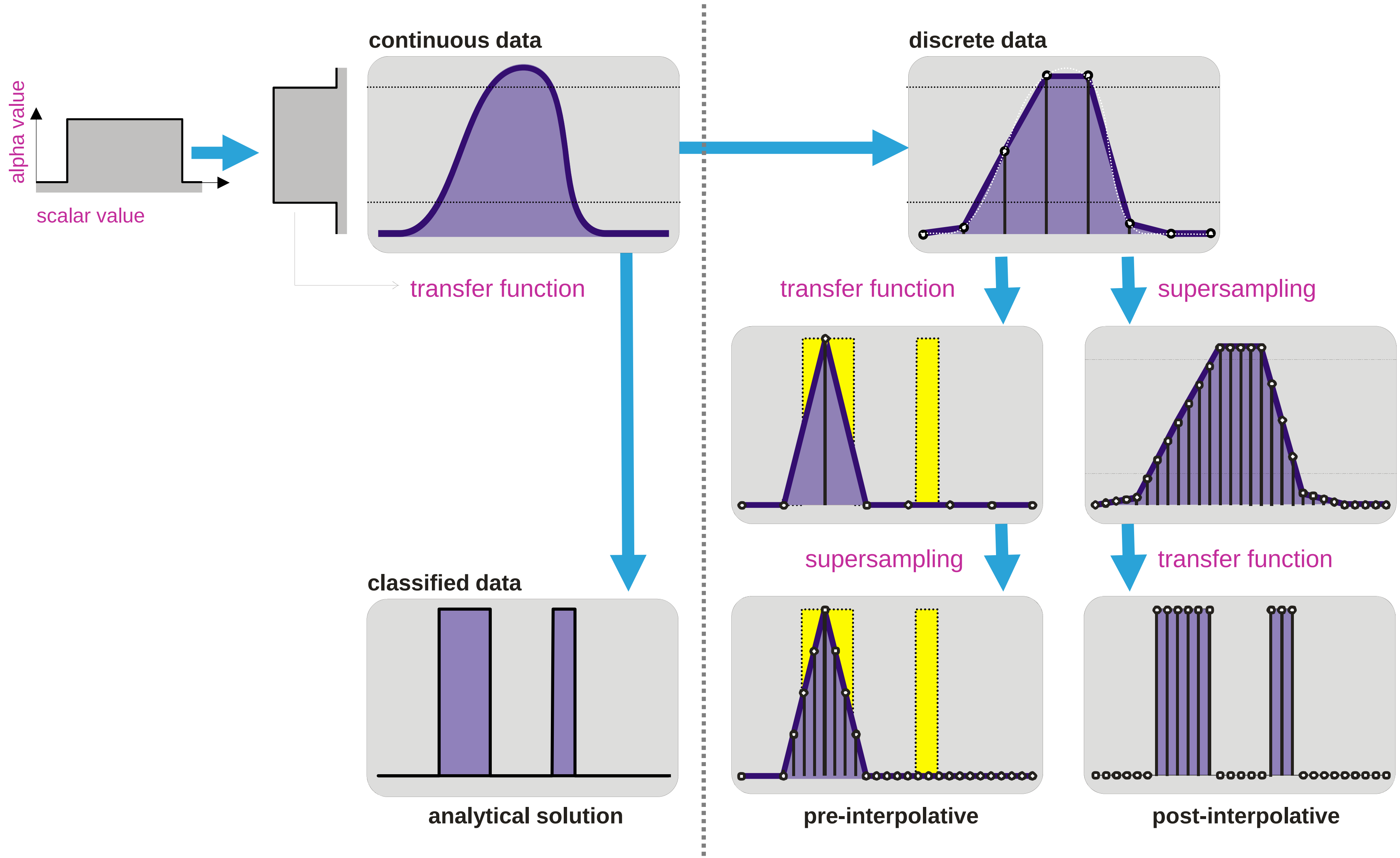
Classification Order (1)

- Shading/classification can occur before or after ray traversal
  - **Pre-interpolative:** classify all data values and then interpolate between RGBA-tuples
  - **Post-interpolative:** interpolate between scalar data values and then classify the result

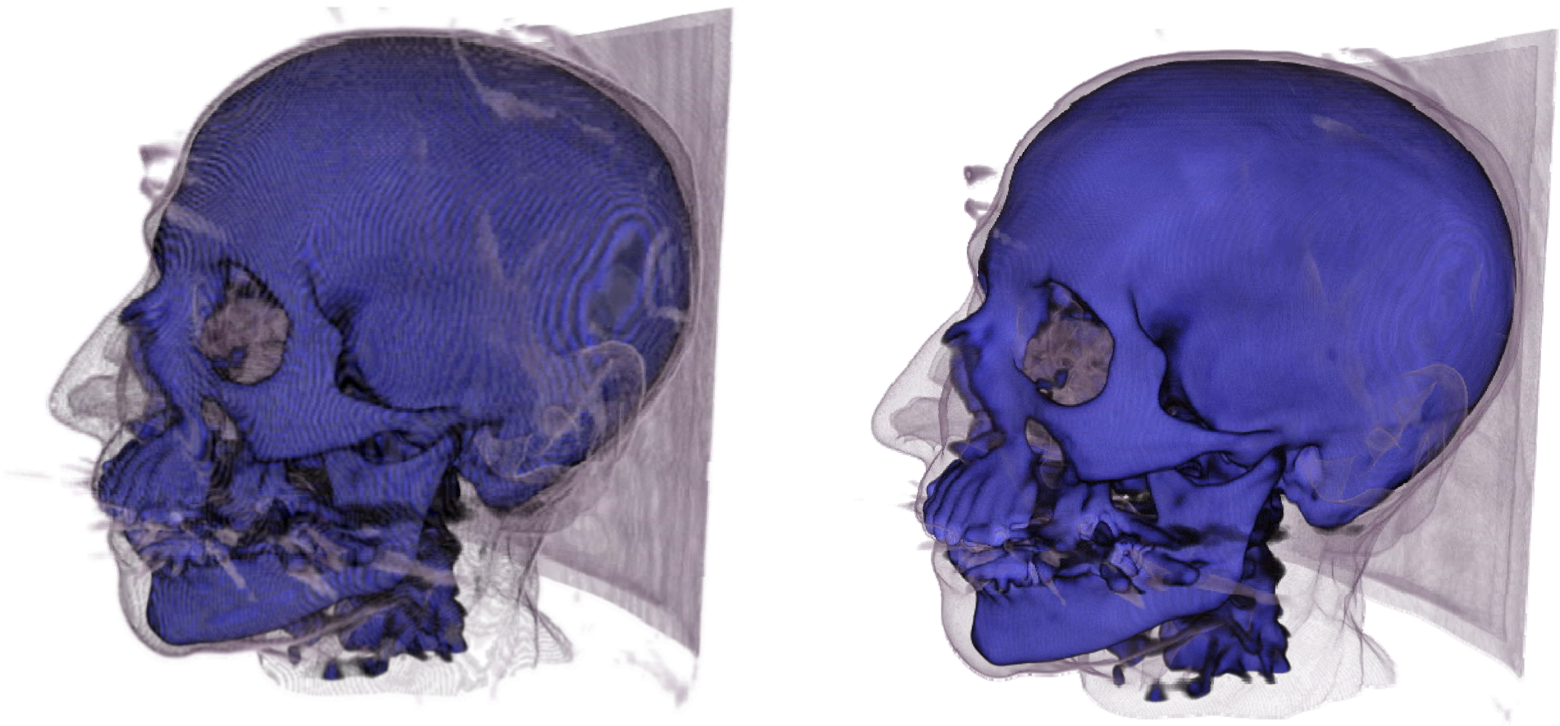
Classification Order (2)



Classification Order (3)



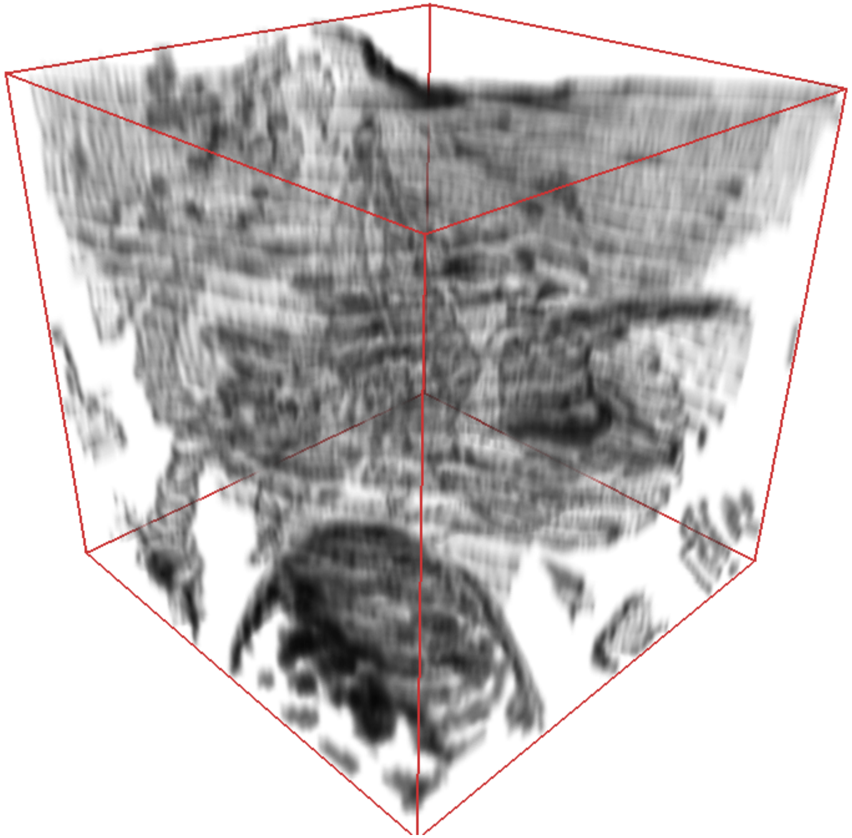
Classification Order: Example 1



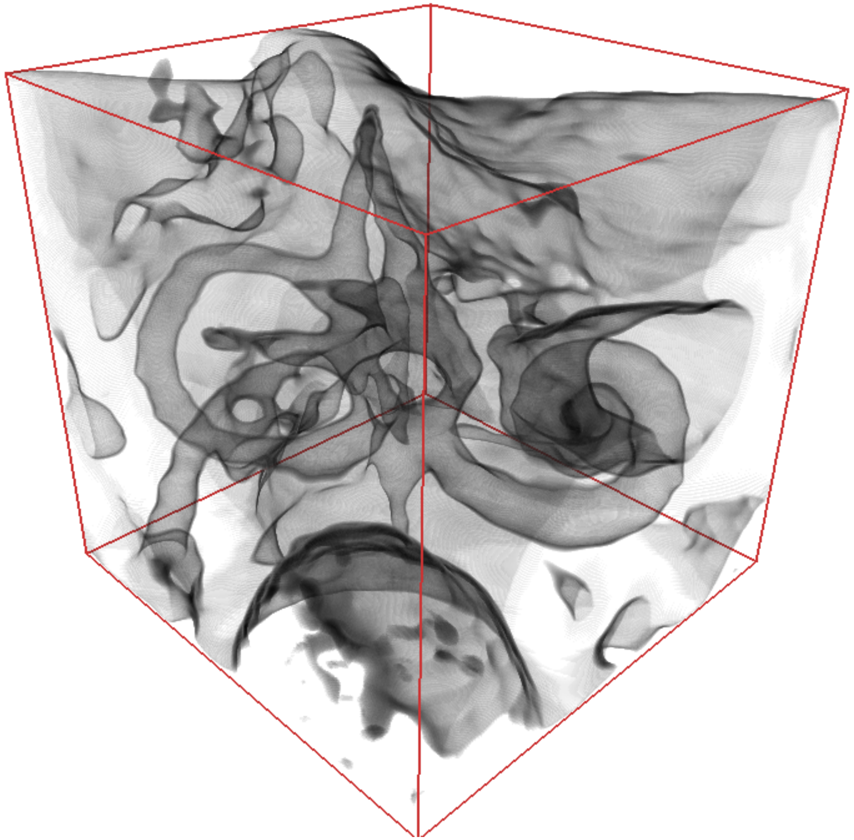
pre-interpolative **post-interpolative**

same transfer function, resolution, and sampling rate

Classification Order: Example 2



pre-interpolative



post-interpolative

same transfer function, resolution, and sampling rate

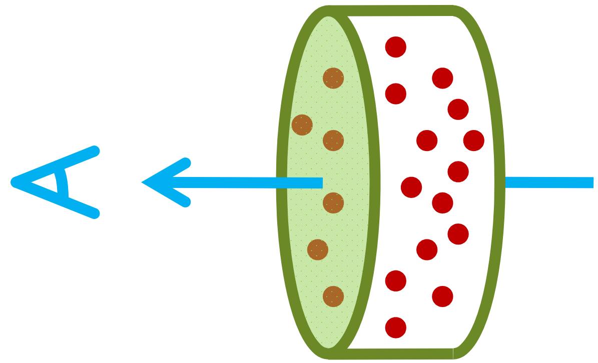
α-Compositing –

a Specific Optical Model for Volume Rendering

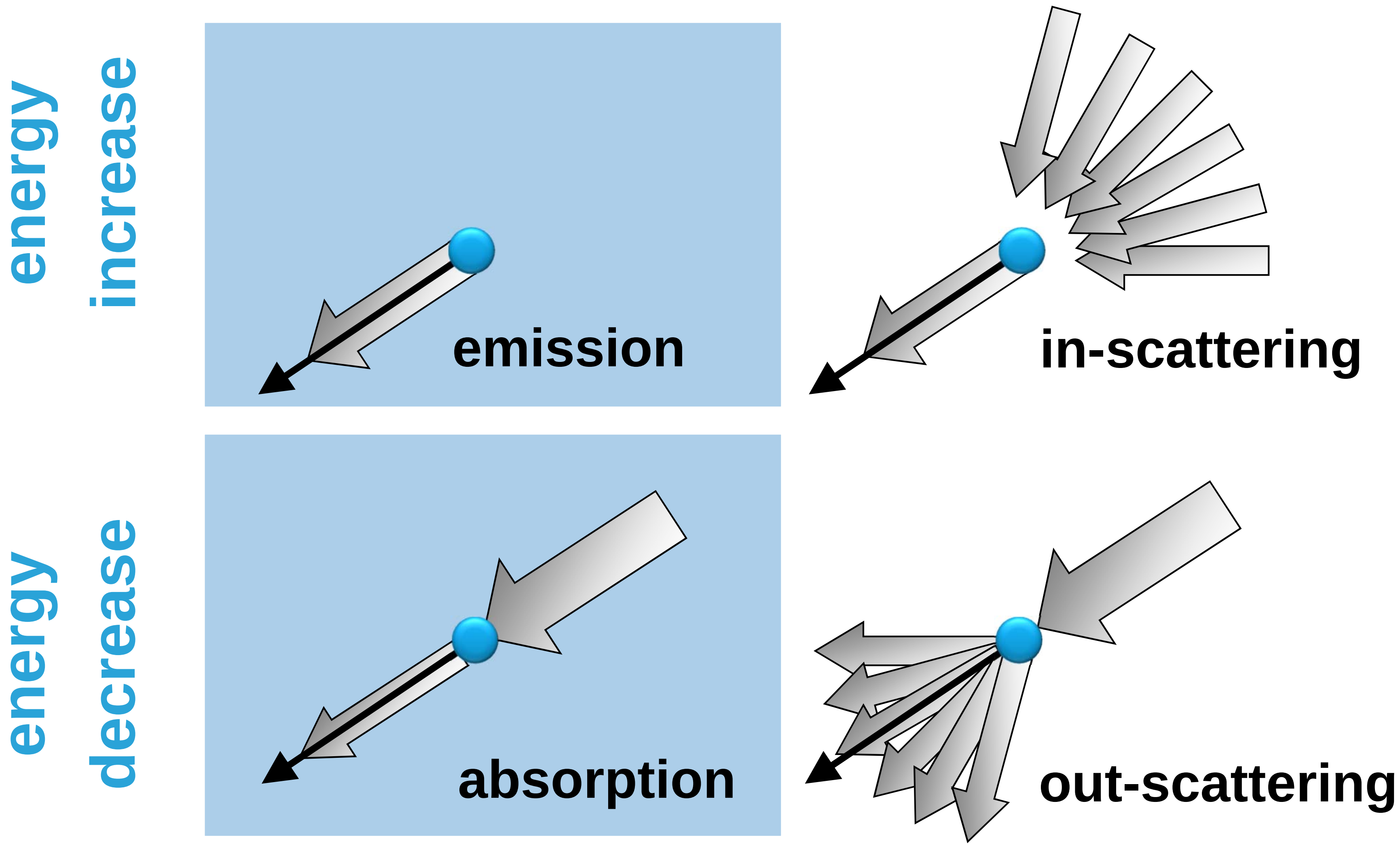
Display of

Semi-Transparent Media

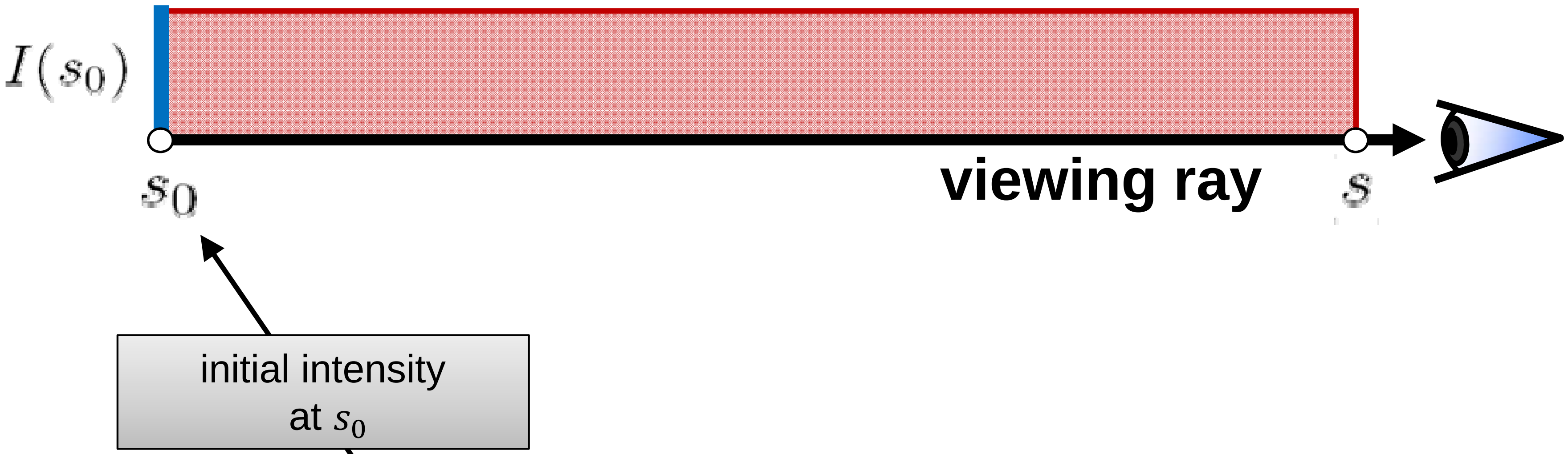
- Various models (Examples):
  - Emission only (light particles)
  - Absorption only (dark fog)
  - Emission & absorption (clouds)
  - Single scattering, w/o shadows
  - Multiple scattering
- Two approaches:
  - Analytical model (via differentials)
  - Numerical approximation (via differences)



Physical Model of Radiative Transfer



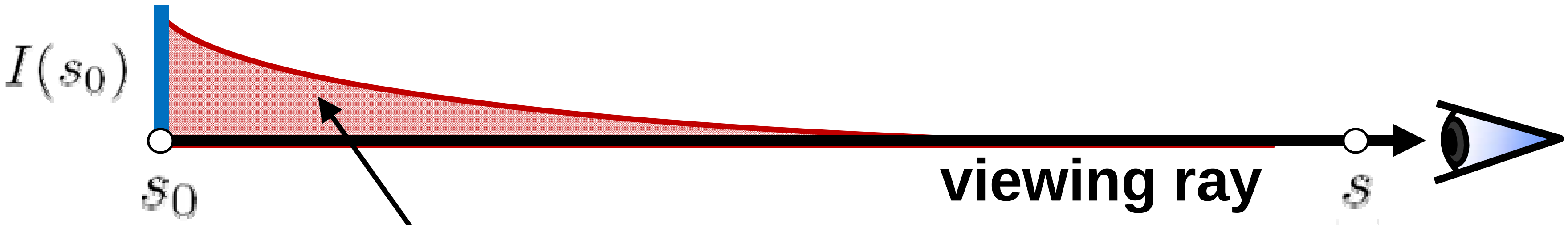
Analytical Model (1)



$$I(s) = I(s_0)$$

without absorption all the initial radiant energy would reach the point s

Analytical Model (2)



absorption between s_0 and s

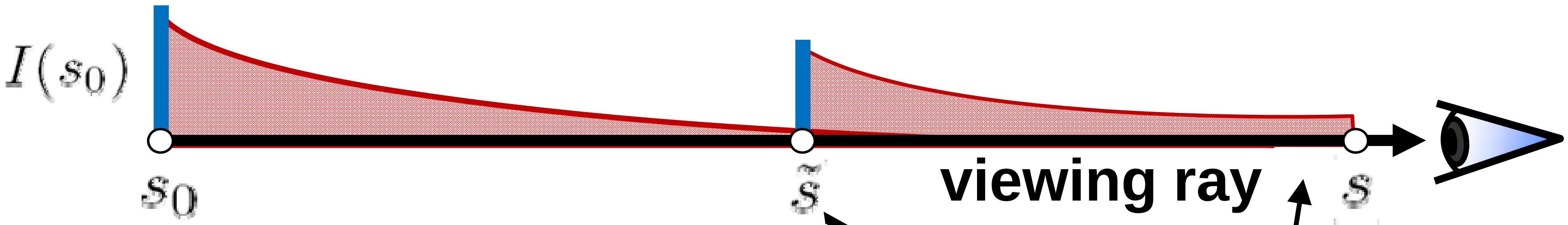
$$I(s) = I(s_0)\, e^{-\tau(s_0, s)}$$

Extinction τ

Absorption κ

$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)\, ds.$$

Analytical Model (3)



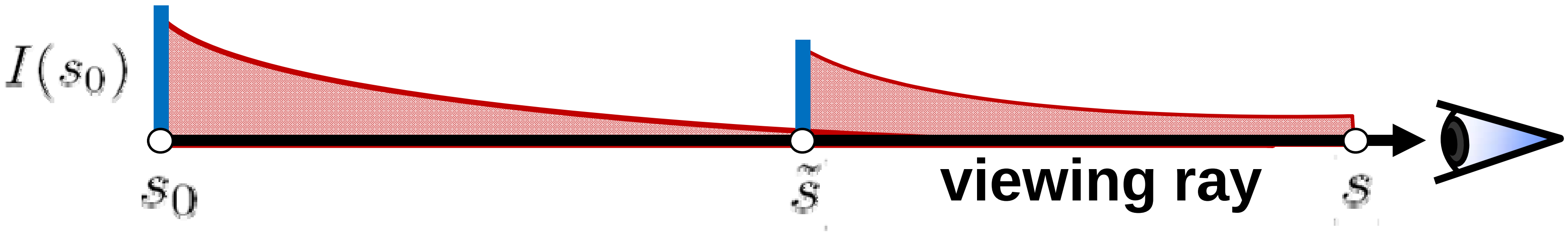
one point $\tilde{s}$ along the viewing ray emits additional radiant energy

active emission at point $\tilde{s}$

absorption between $\tilde{s}$ and s

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + q(\tilde{s})\, e^{-\tau(\tilde{s},s)}$$

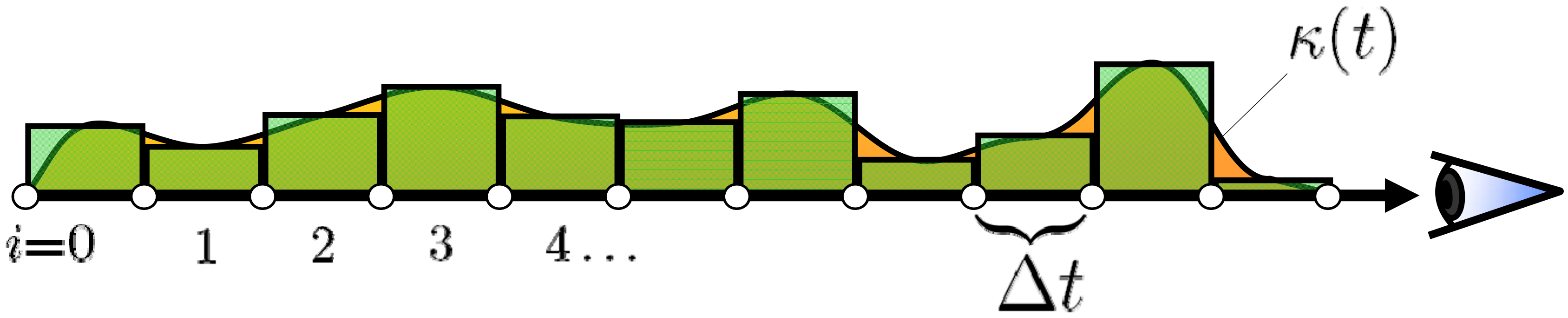
Analytical Model (4)



every point $\tilde{s}$ along the viewing ray emits additional radiant energy

$$I(s) = I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)}\, d\tilde{s}$$

Numerical Approximation (1)



Extinction: $$\tau(0,t) = \int_0^t \kappa(\hat{t})\, d\hat{t}$$

approximate integral by Riemann sum:

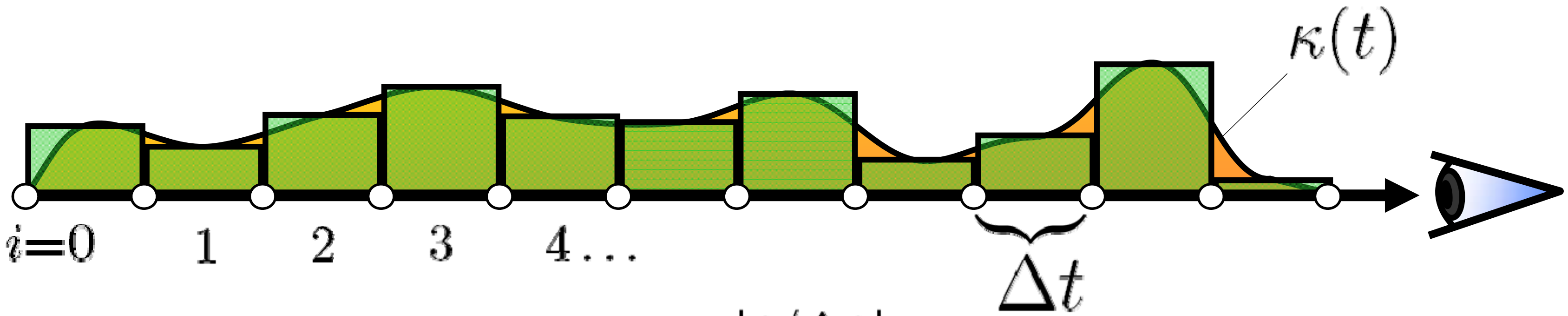
$$\tau(0,t) \approx \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

Numerical Approximation (2)

$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = e^{-\sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t}$$

Numerical Approximation (3)

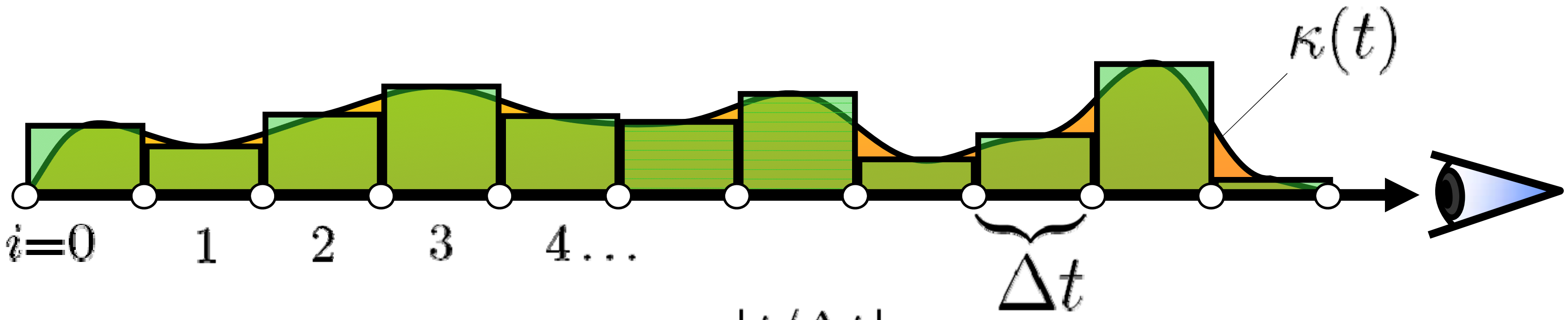


$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} e^{-\kappa(i \cdot \Delta t)\, \Delta t}$$

now we introduce opacity:

$$A_i = 1 - e^{-\kappa(i \cdot \Delta t)\, \Delta t}$$



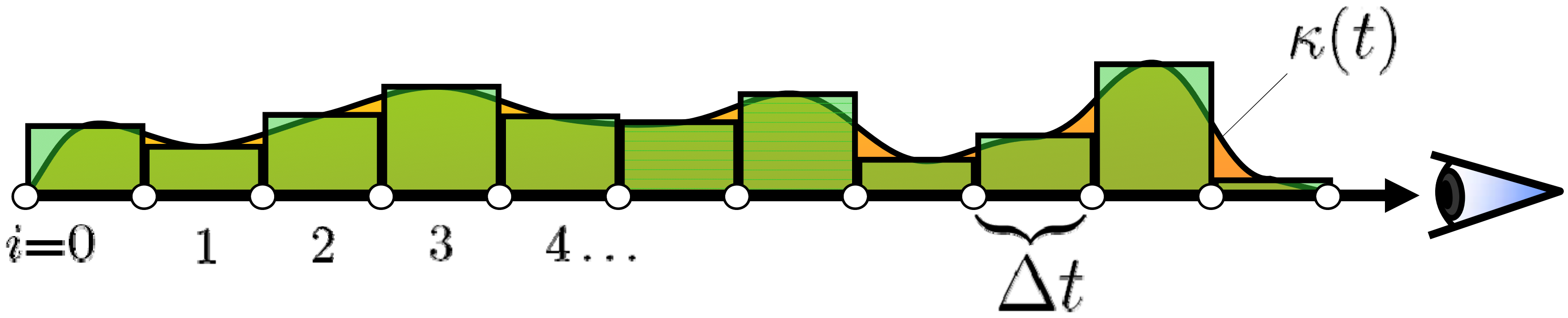
$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t$$

$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} e^{-\kappa(i \cdot \Delta t)\, \Delta t}$$

now we introduce opacity:

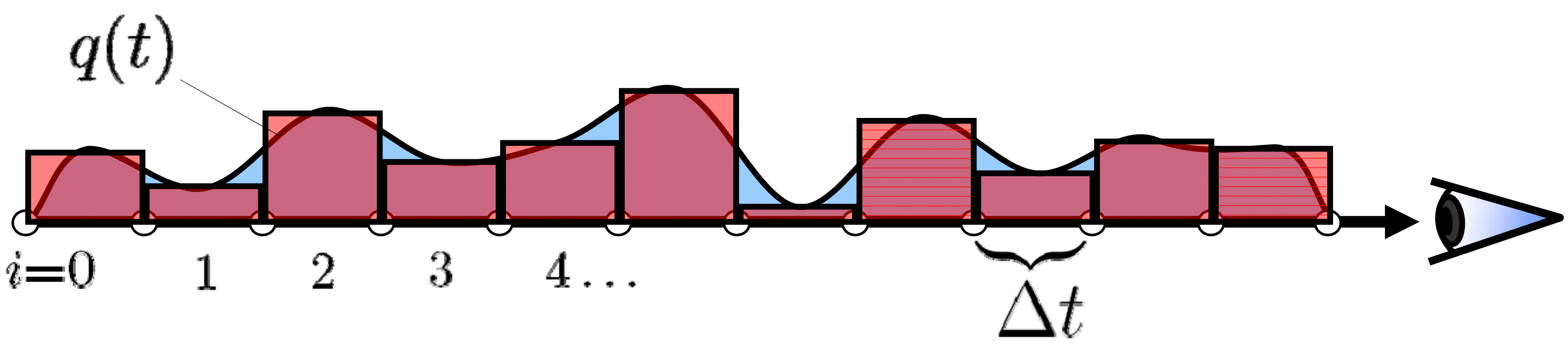
$$(1 - A_i) = e^{-\kappa(i \cdot \Delta t)\, \Delta t}$$

Numerical Approximation (5)



$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

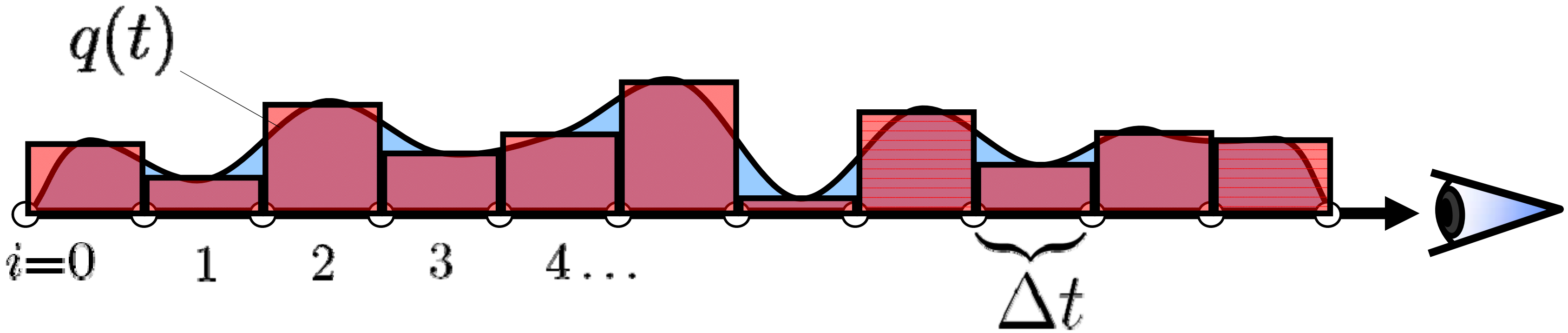
Numerical Approximation (6)



$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

$$q(t) \approx C_i = c(i \cdot \Delta t)\,\Delta t$$

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \, e^{-\tilde{\tau}(0,t)}$$

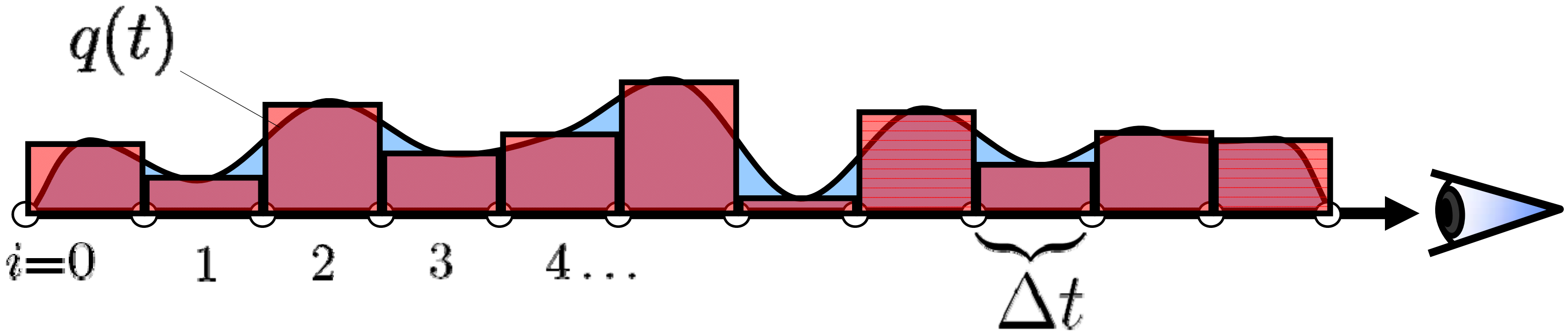


$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

$$q(t) \approx C_i = c(i \cdot \Delta t)\, \Delta t$$

$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \prod_{j=0}^{i-1} (1 - A_j)$$

Numerical Approximation (8)



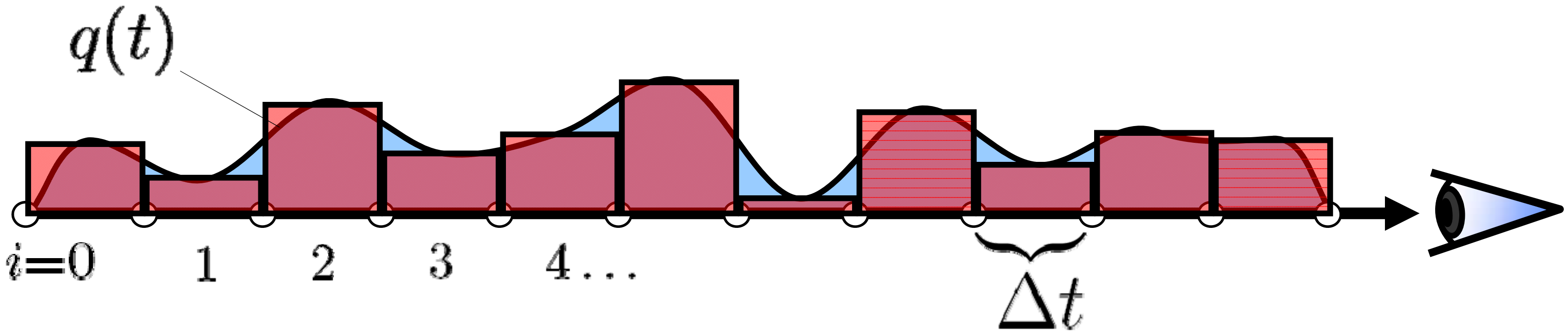
$$\tilde{C} = \sum_{i=0}^{\lfloor T/\Delta t \rfloor} C_i \prod_{j=0}^{i-1} (1 - A_j)$$

can be computed recursively:

$$C'_i = C_i + (1 - A_i) C'_{i-1}$$

- radiant energy observed at position i (C'_i)
- radiant energy emitted at position i (C_i)
- absorption at position i (A_i)
- radiant energy observed at position $i-1$ (C'_{i-1})

Numerical Approximation (9)



back-to-front compositing

$$C'_i = C_i + (1 - A_i)C'_{i-1}$$

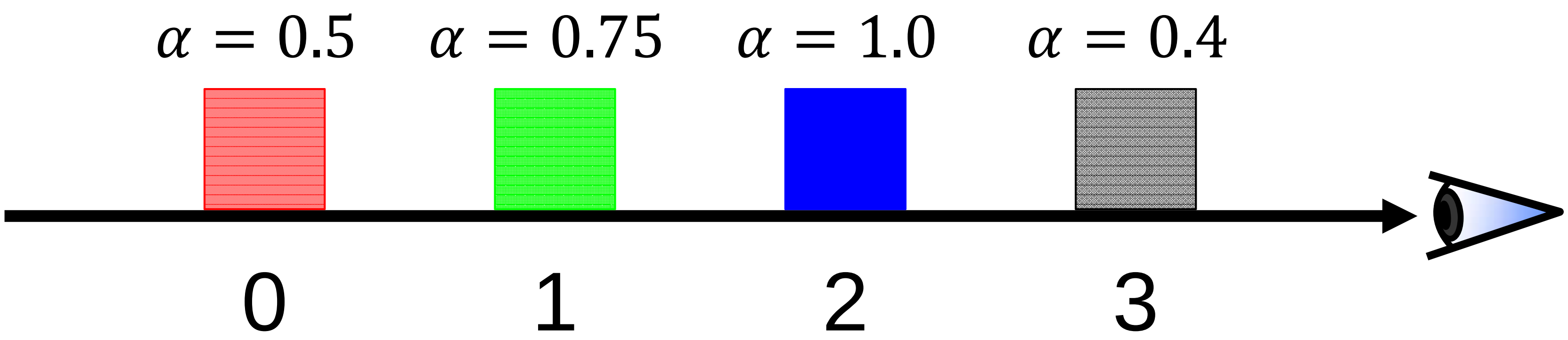
front-to-back compositing

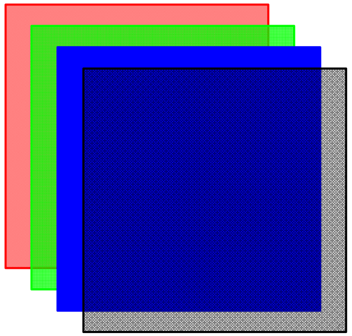
$$C'_i = C'_{i+1} + (1 - A'_{i+1})C_i$$

$$A'_i = A'_{i+1} + (1 - A'_{i+1})A_i$$

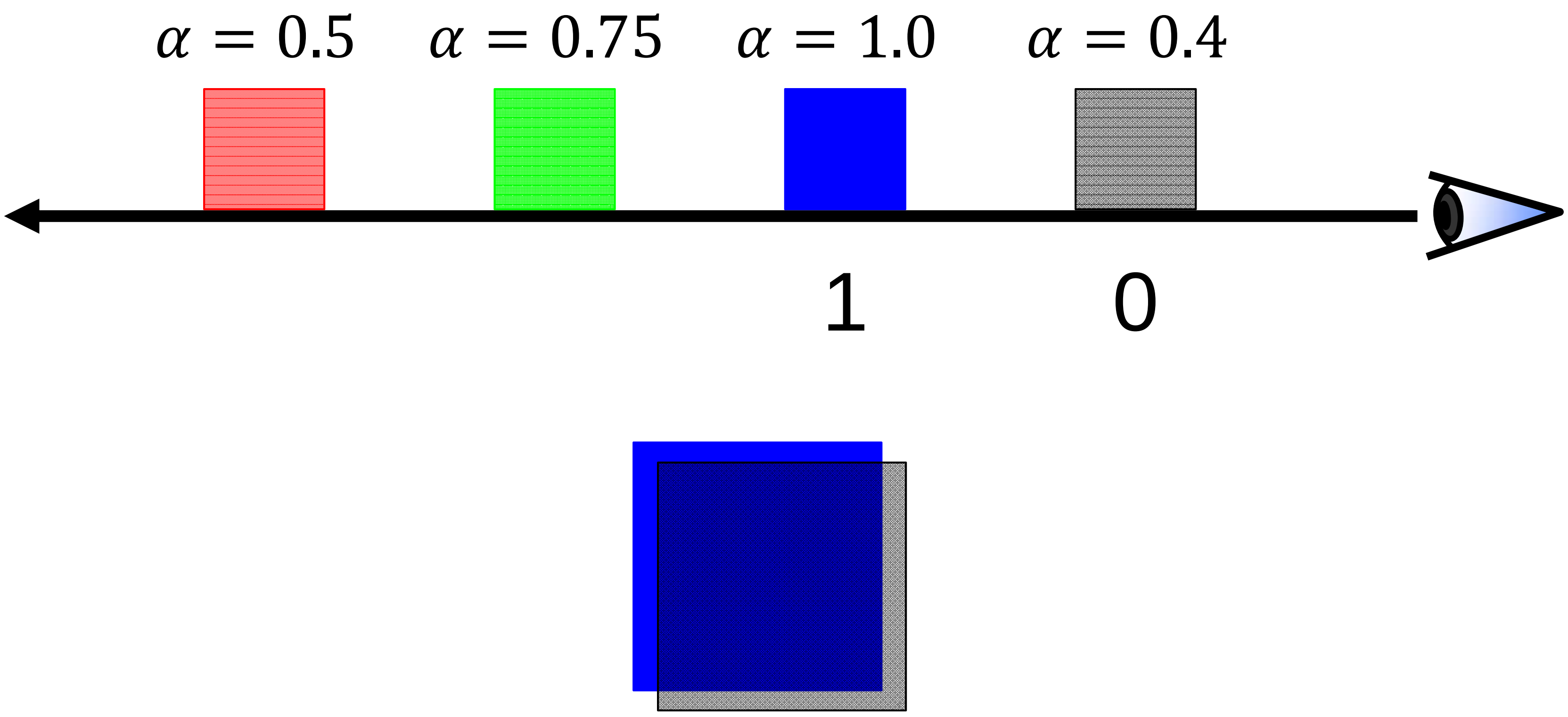
early ray termination: stop the calculation when $A'_i \approx 1$

Back-to-Front Compositing: Example





Front-to-Back Compositing: Example



Summary

- Emission Absorption Model

$$I(s) \;=\; I(s_0)\, e^{-\tau(s_0,s)} + \int_{s_0}^{s} q(\tilde{s})\, e^{-\tau(\tilde{s},s)}\, \mathrm{d}\tilde{s}$$

- Numerical Solutions **[pre-multiplied alpha assumed]**

back-to-front iteration

$$C'_i \;=\; C_i + (1 - A_i) C'_{i-1}$$

front-to-back iteration

$$C'_i \;=\; C'_{i+1} + (1 - A'_{i+1}) C_i$$

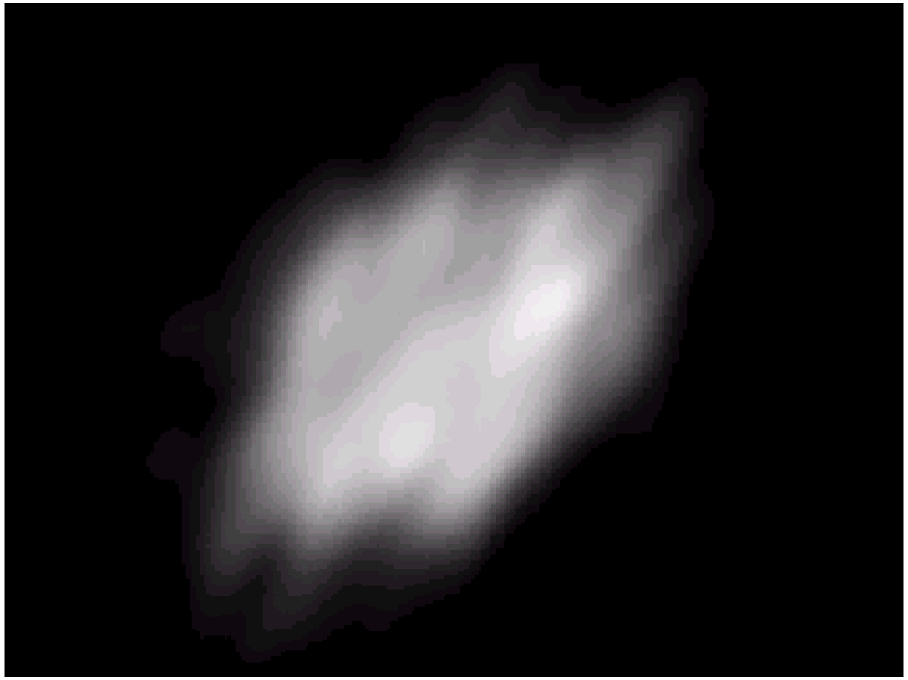
$$A'_i \;=\; A'_{i+1} + (1 - A'_{i+1}) A_i$$

Pre-Multiplied Alpha

- Color values are stored pre-multiplied by their opacity: $(\alpha r, \alpha g, \alpha b)$
- Consequence: transparent red is the same as transparent black, etc.
- Simplifies blending: color and alpha values are treated equally
- Can result in loss of precision

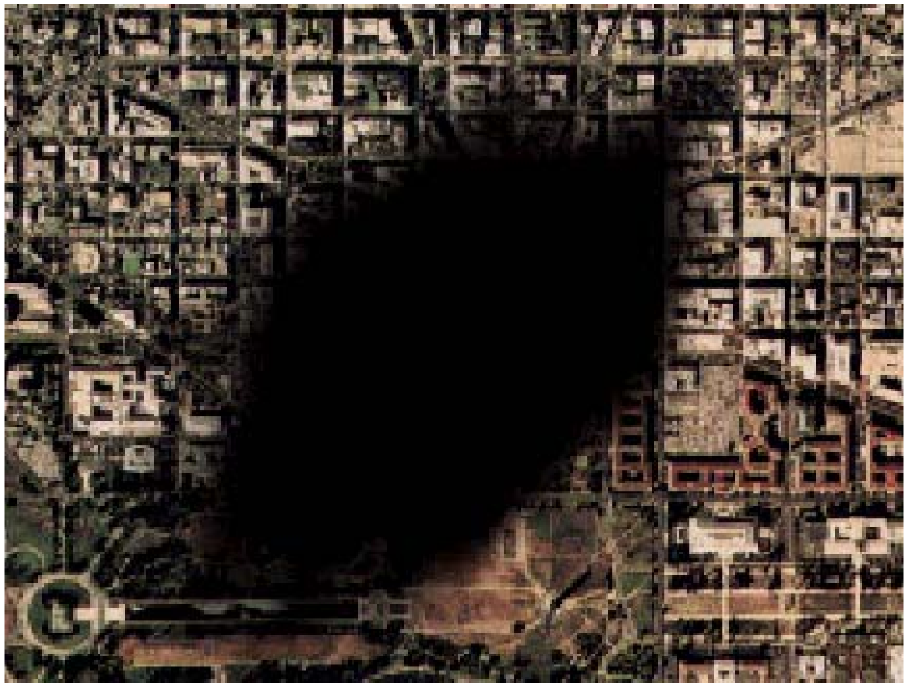
Emission or/and Absorption

Emission
only



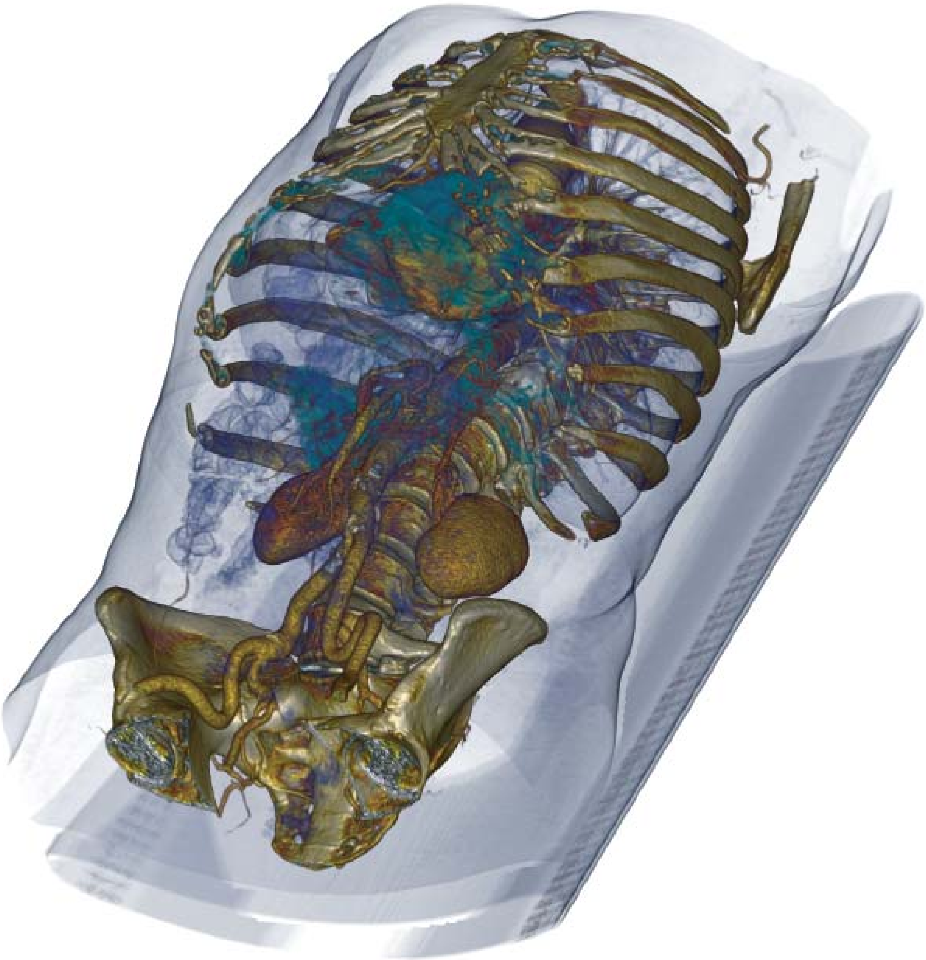
Emission
and Absorption

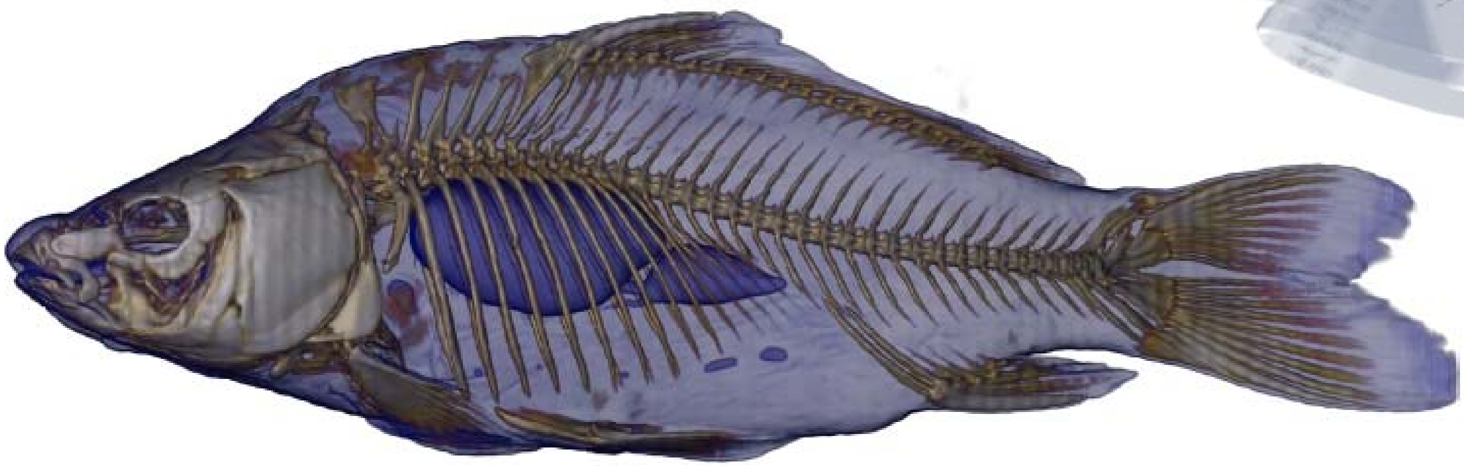
Absorption
only



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

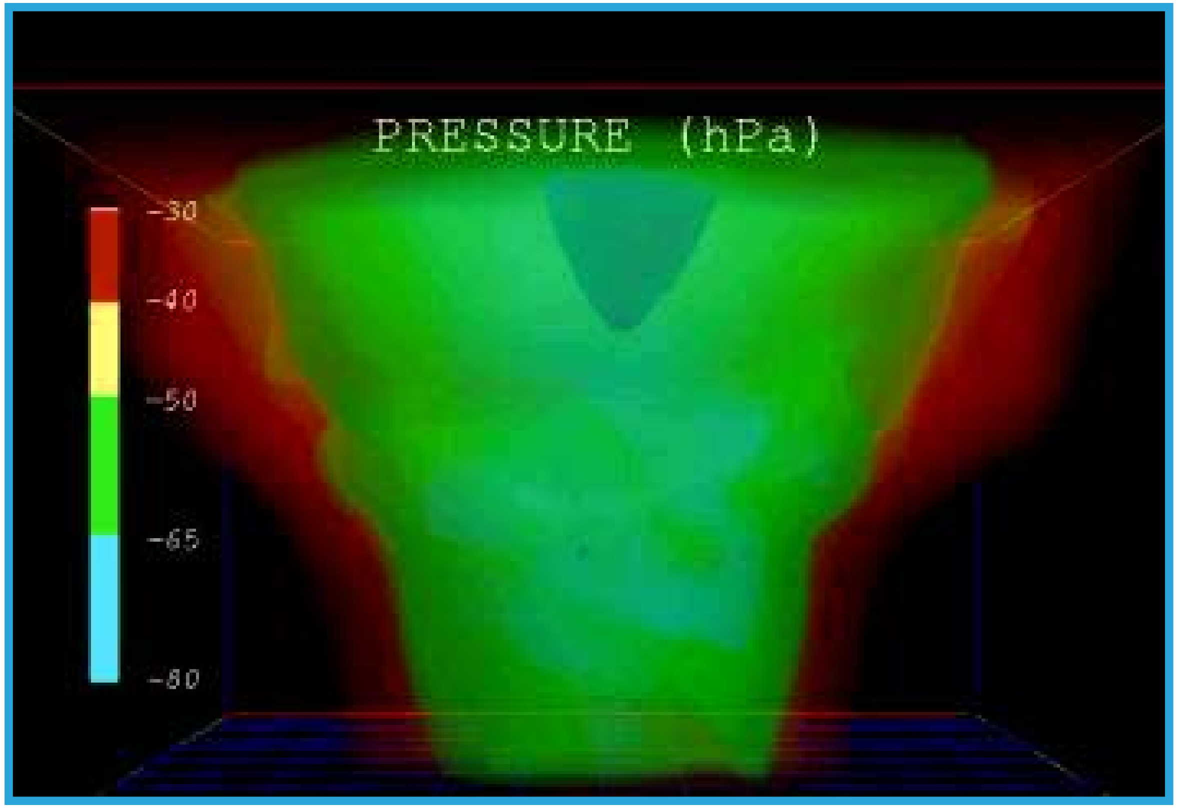
Ray Casting – Examples

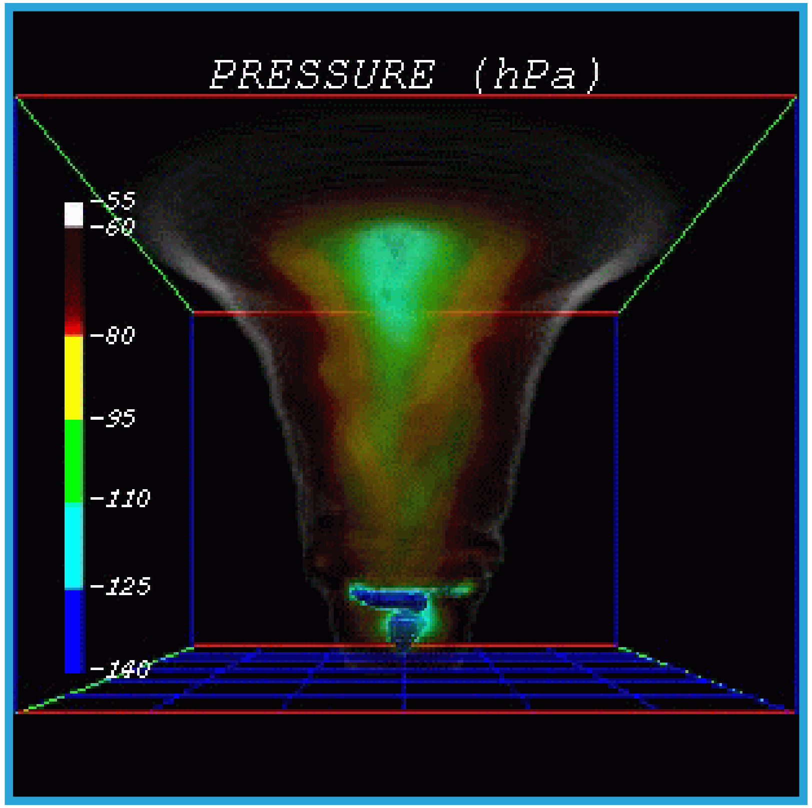




Ray Casting – Further Examples

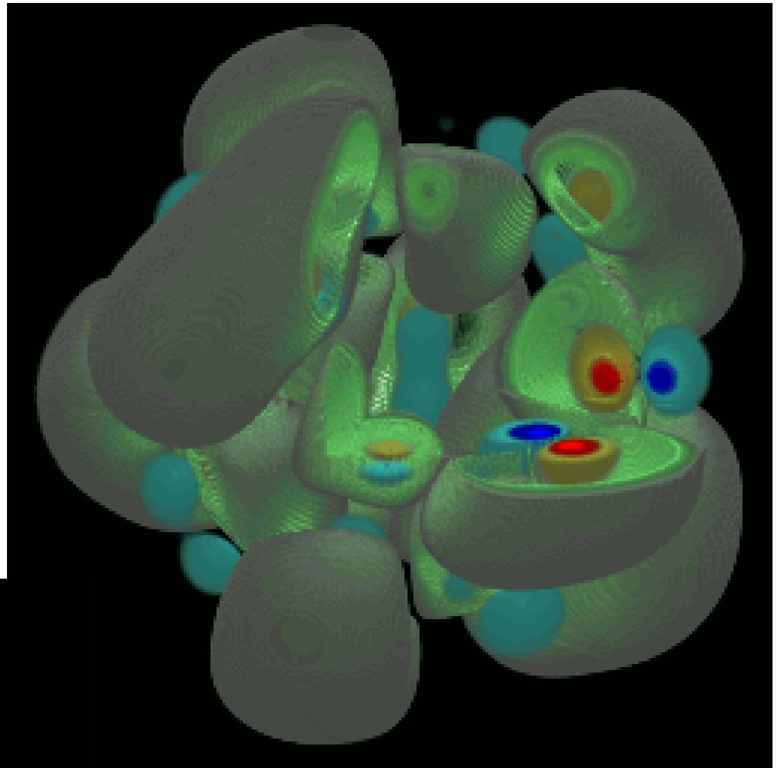
- Tornado Visualization:

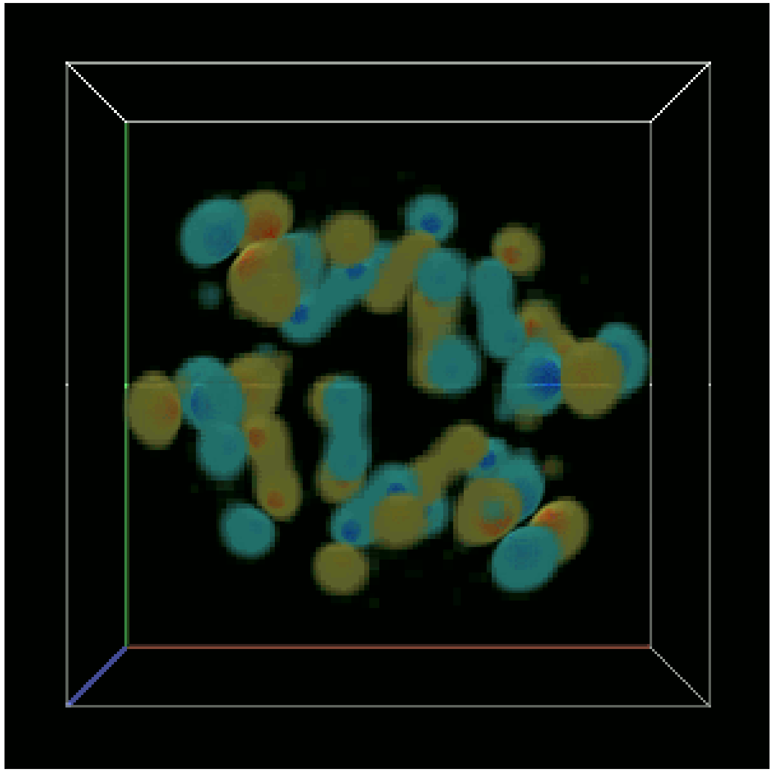


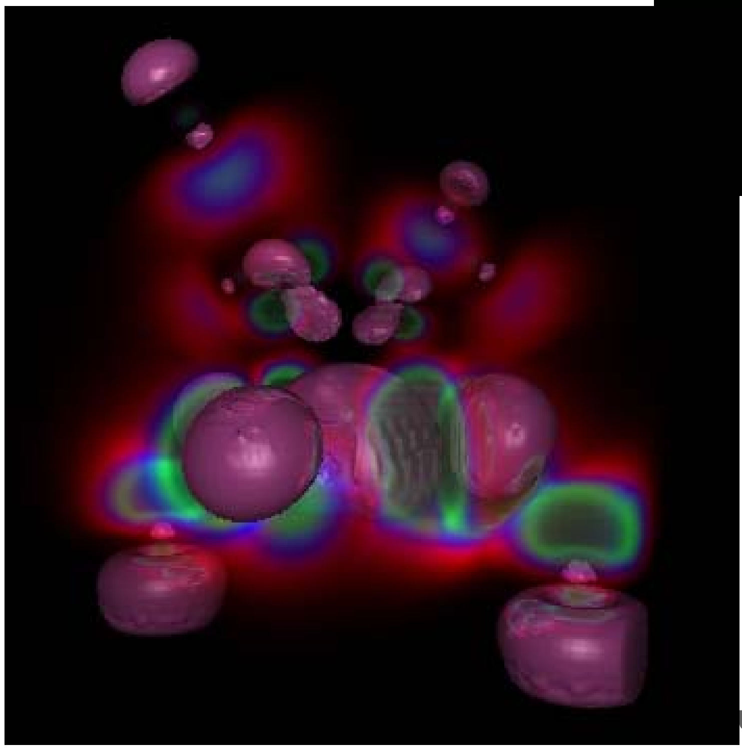


Ray Casting – Further Examples

- Molecular data:







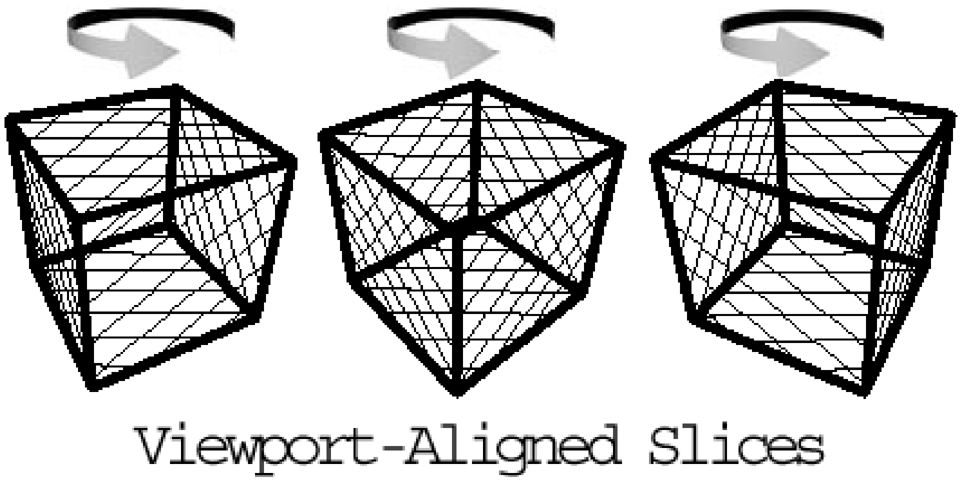
Hardware-Volume Visualization

Faster with Hardware?!

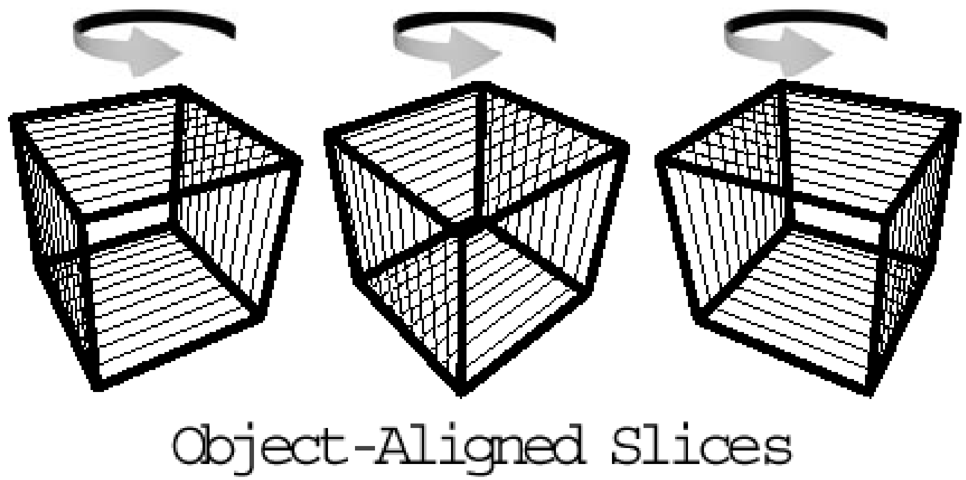
Two Approaches

- 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

- 3D-textures:
  - Number of slices varies



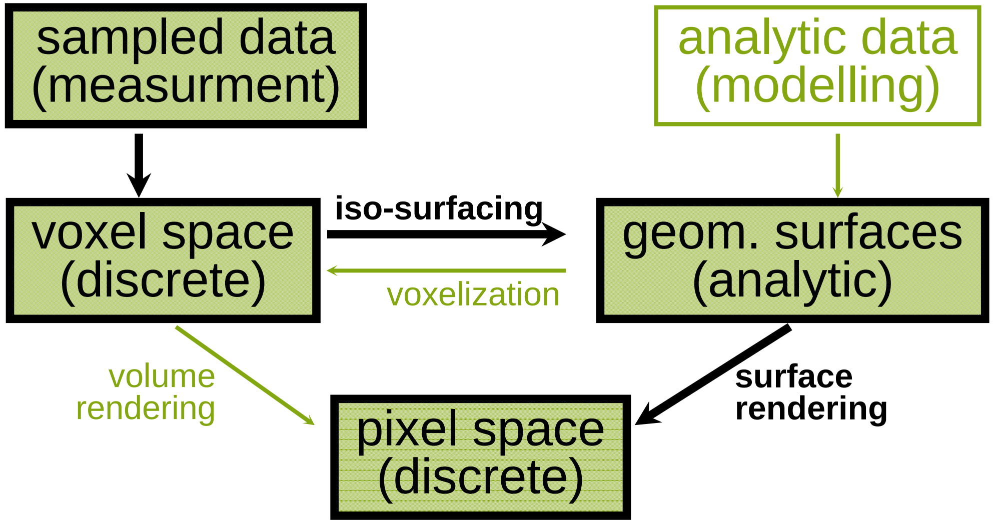
- 2D-textures:
  - Stack change: discontinuity

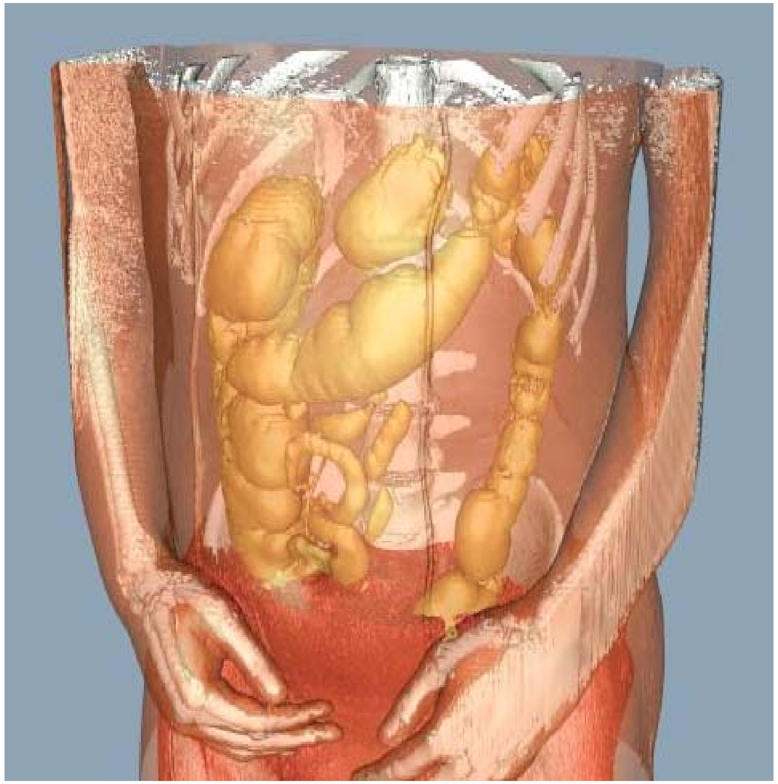


Indirect Volume Visualization

Iso-Surface-Display

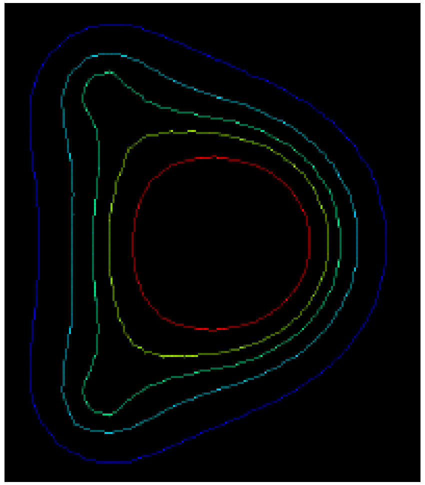
- Example
  - CT measurement
  - Iso-stack-conversion
  - Iso-surface-calculation (marching cubes)
  - Surface rendering (OpenGL)

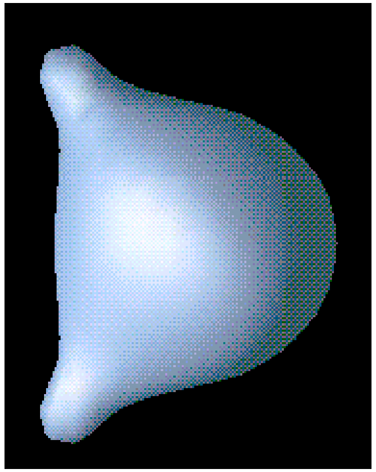




Iso-Surfaces

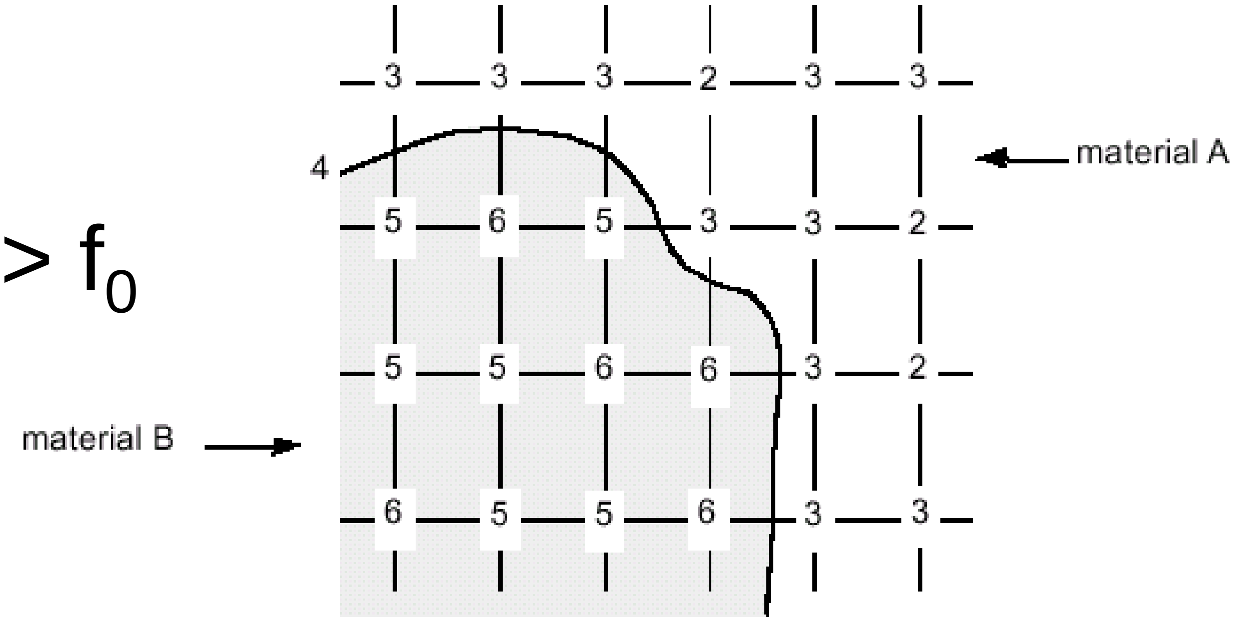
- Intermediate representation
- Aspects:
  - Preconditions:
    - expressive Iso-value, Iso-value separates materials
    - Interest: in transitions
  - Very selective (binary selection / omission)
  - Uses traditional hardware
  - Shading ⇒ 3D-impression!





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

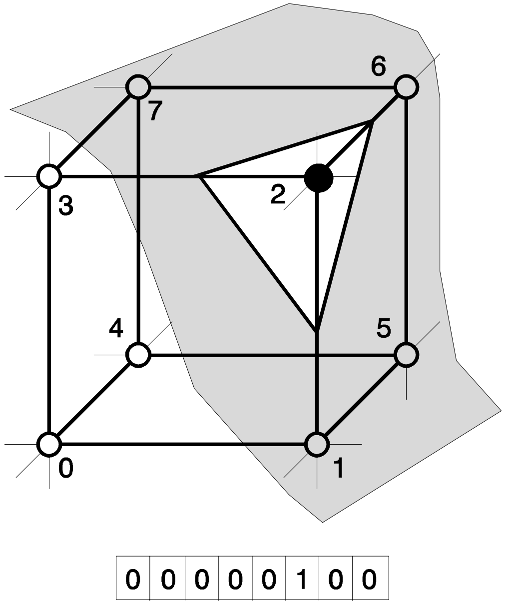


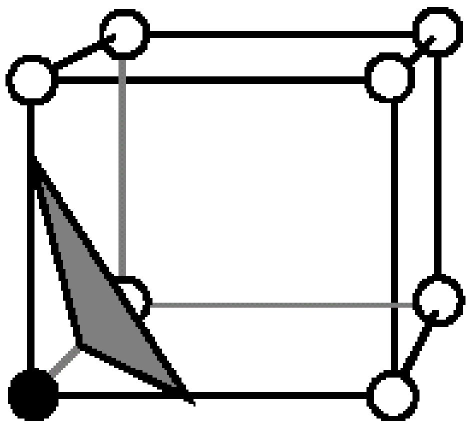
Marching Cubes (MC)

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

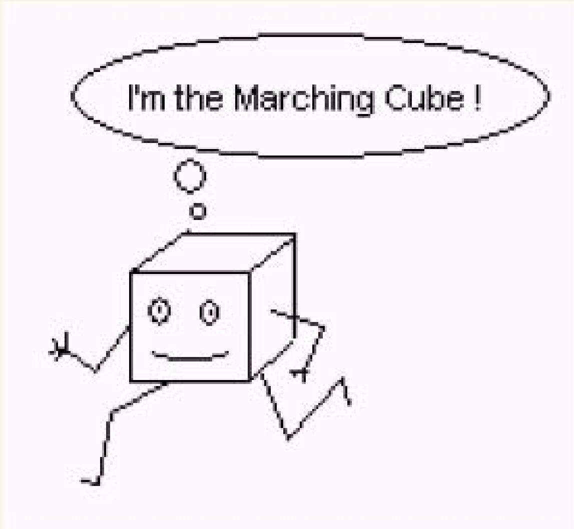




Marching Cubes

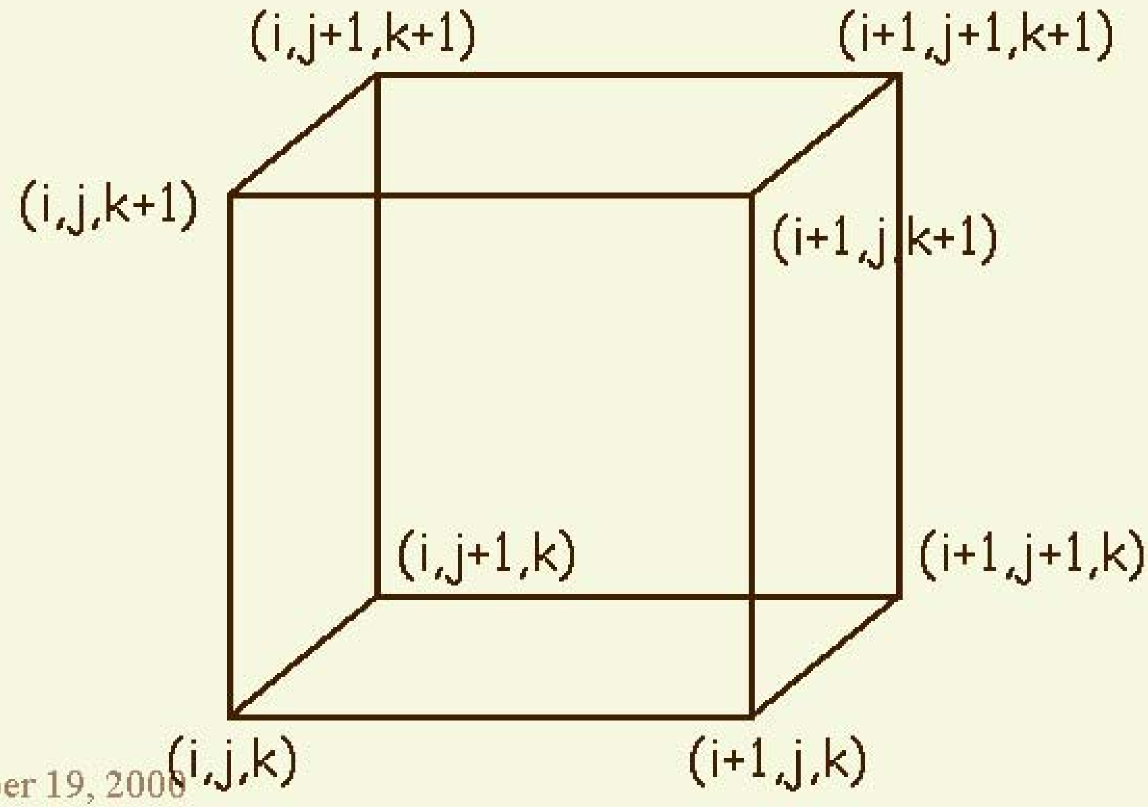
- ✓ Cell consists of 4(8) pixel (voxel) values: (i+[01], j+[01], k+[01])

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



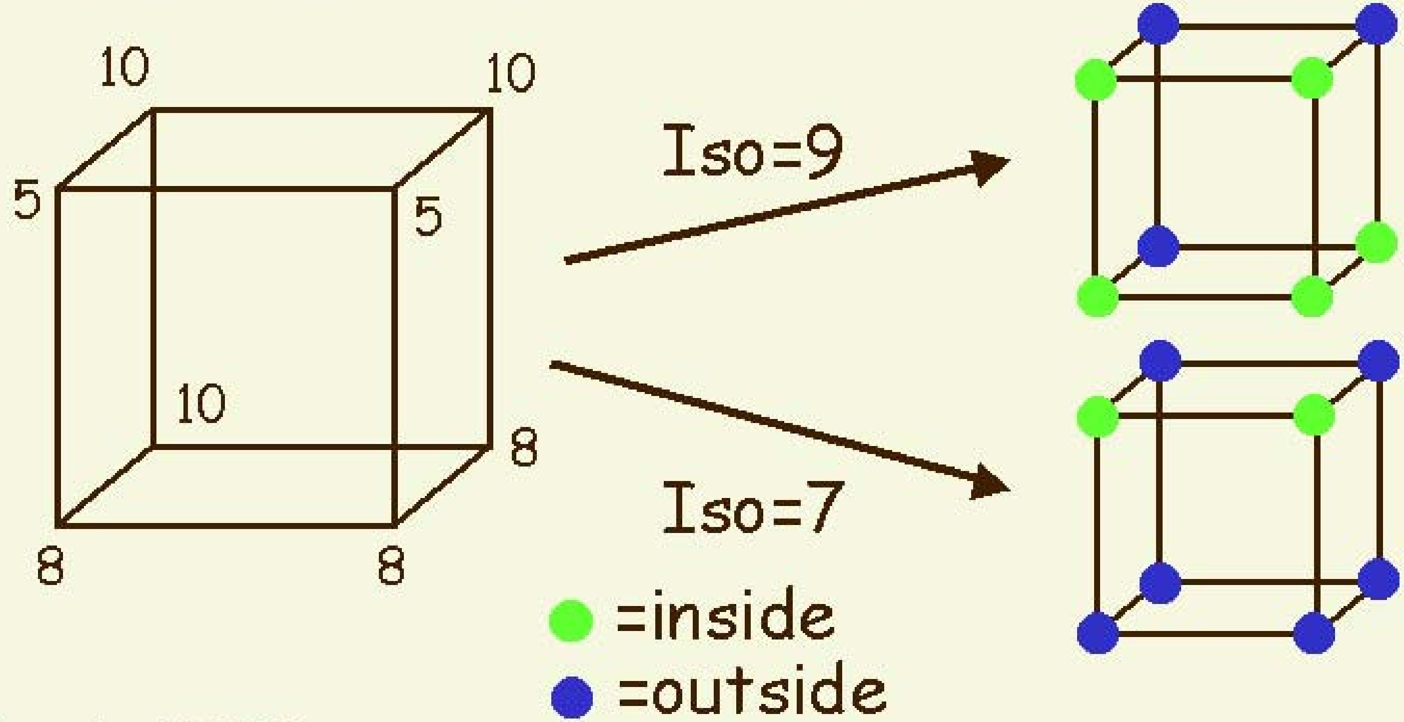
MC 1: Create a Cube

- ✓ Consider a Cube defined by eight data values:



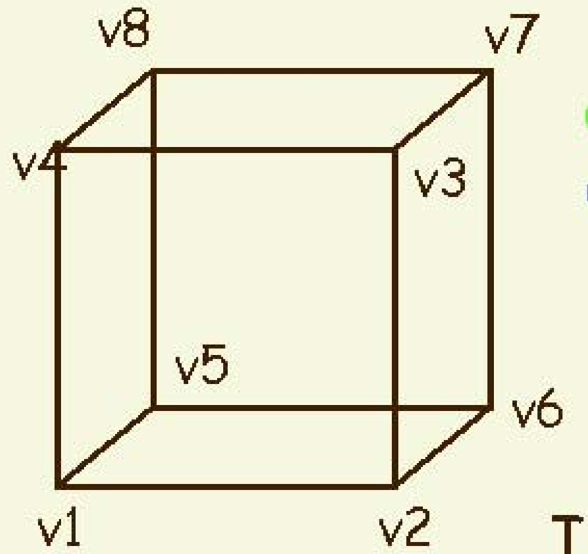
MC 2: Classify Each Voxel

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

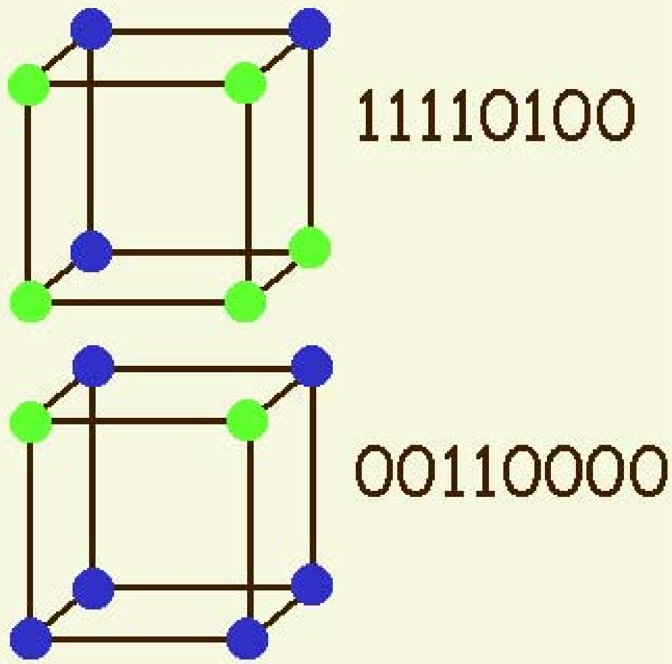


MC 3: Build An Index

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



- inside =1
- outside=0

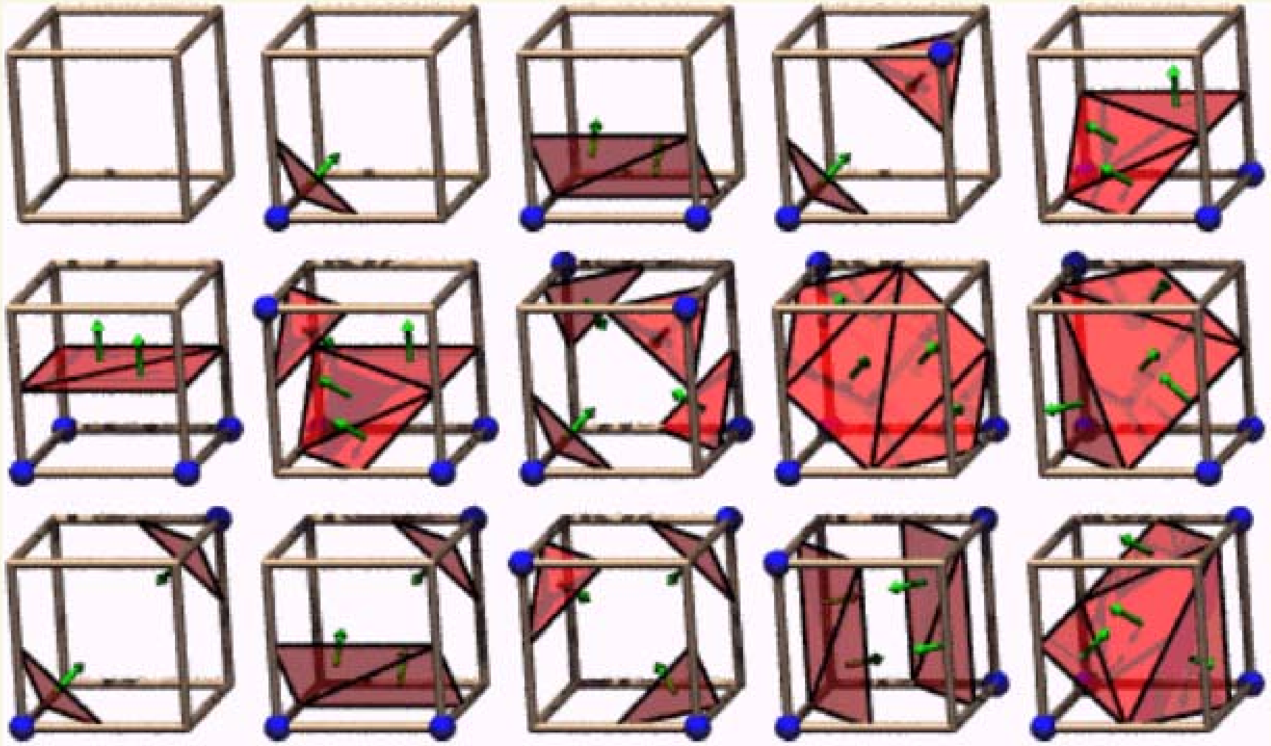


Index:

v1	v2	v3	v4	v5	v6	v7	v8

MC 4: Lookup Edge List

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

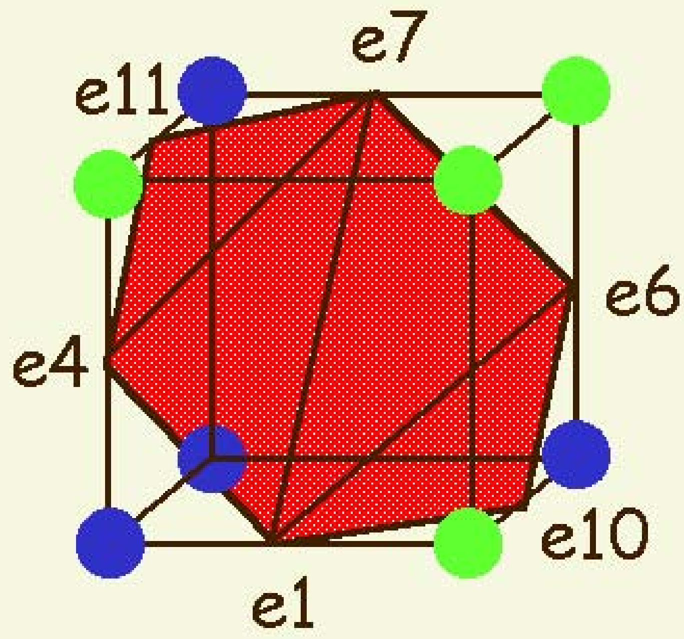


The 15 Cube Combinations

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

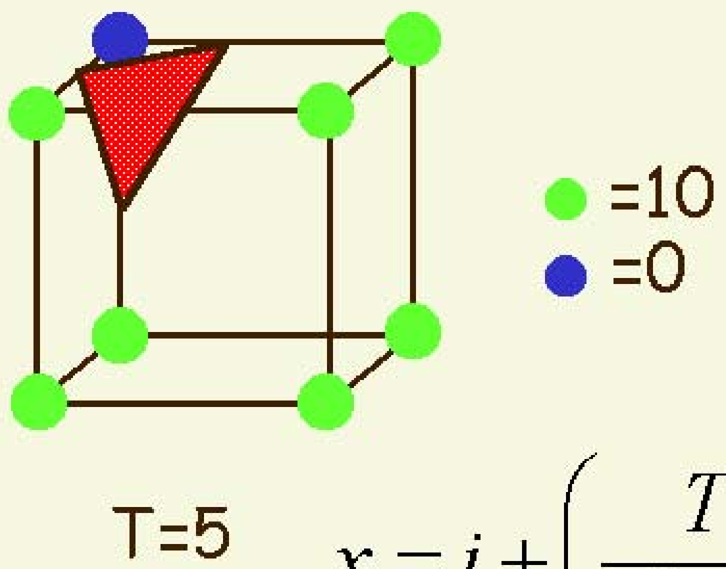
MC 5: Example

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

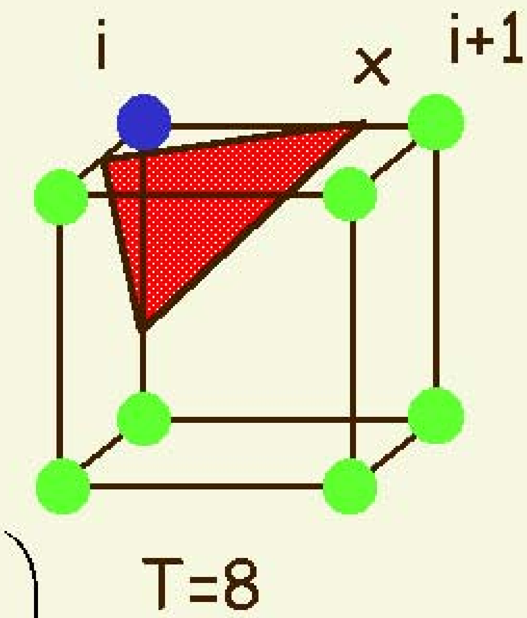


MC 6: Interp. Triangle Vertex

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



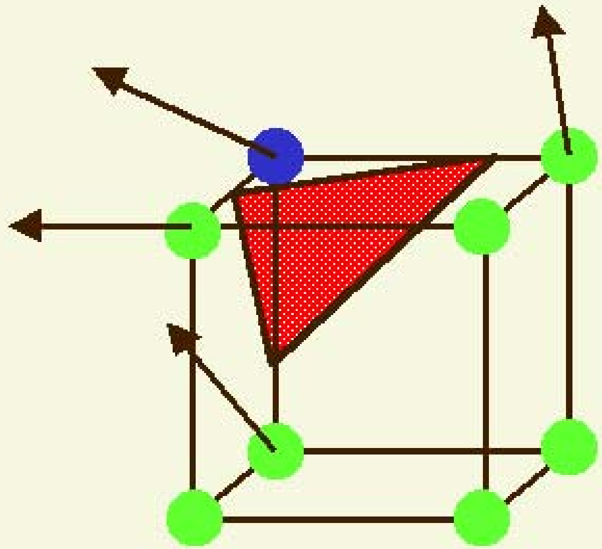
$$x = i + \left(\frac{T - v[i]}{v[i+1] - v[i]} \right)$$



MC 7: Compute Normals

- ✓ Calculate the normal at each cube vertex

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

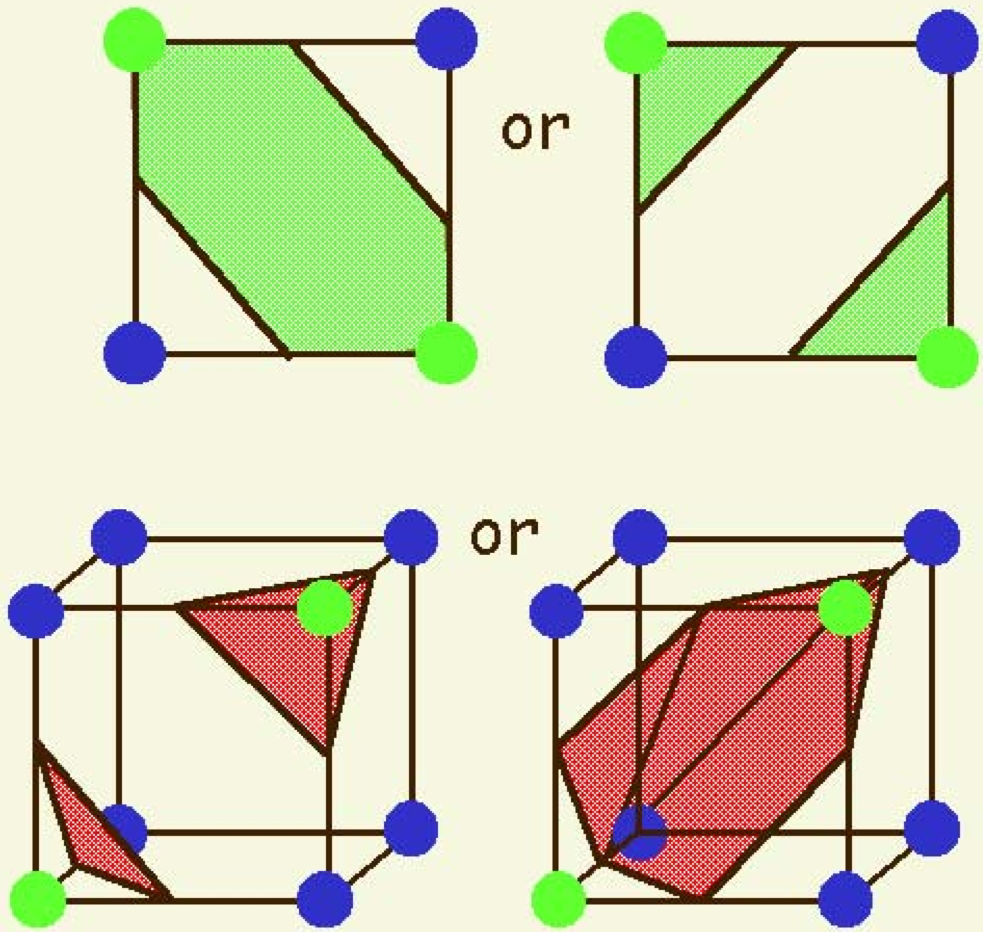


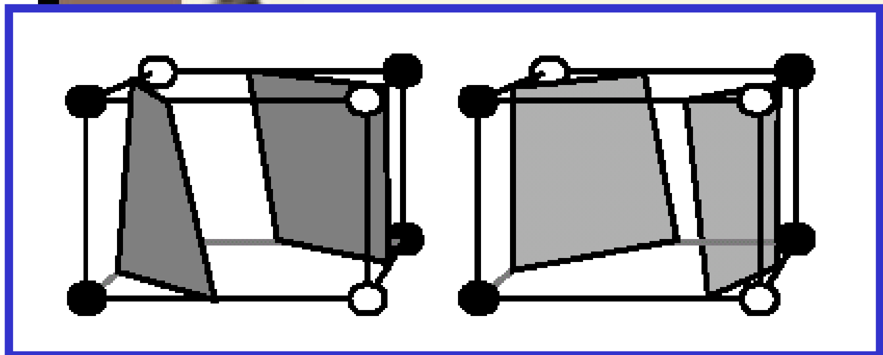
$$\vec{N} = \frac{\vec{G}}{|\vec{G}|}$$

- ✓ Use linear interpolation to compute the polygon vertex normal

MC 8: Ambiguous Cases

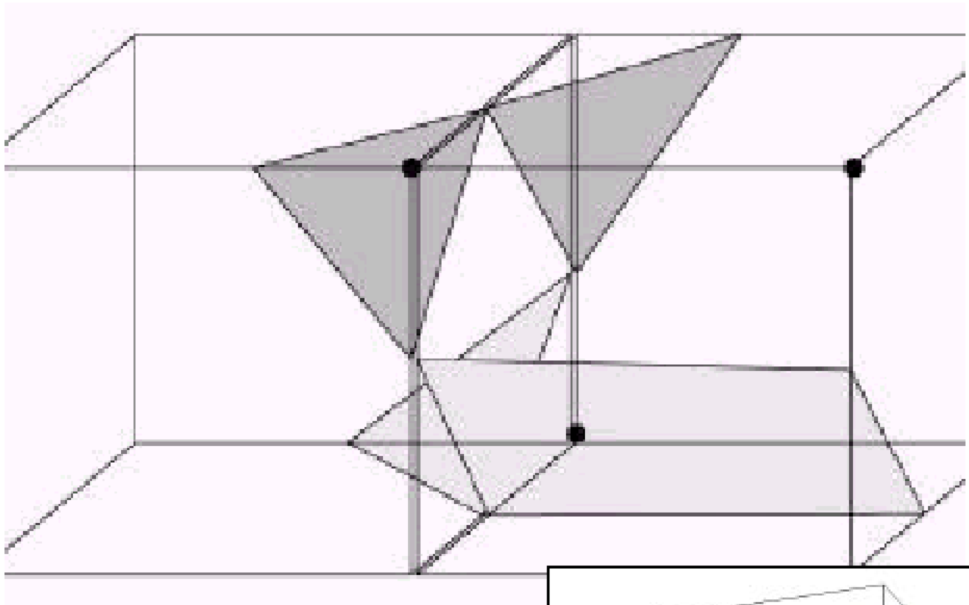
- Ambiguous cases: 3, 6, 7, 10, 12, 13
- Adjacent vertices: different states
- Diagonal vertices: same state
- Resolution: decide for one case

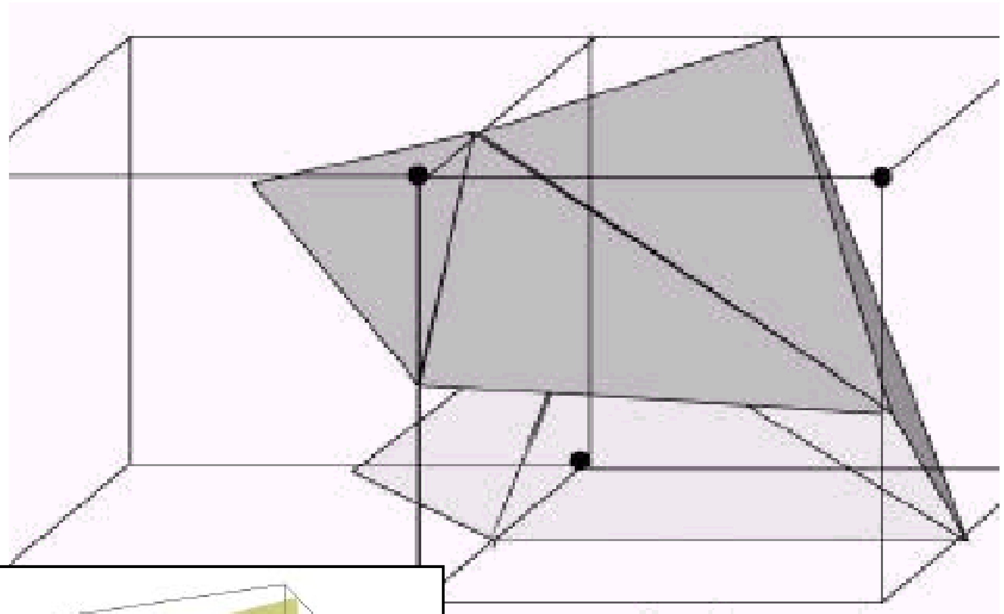




Danger: Holes!

- Wrong vs. correct classification!





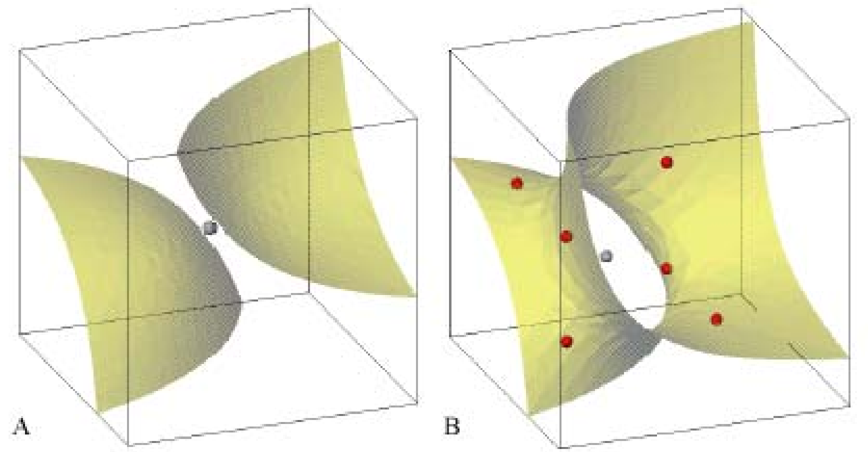
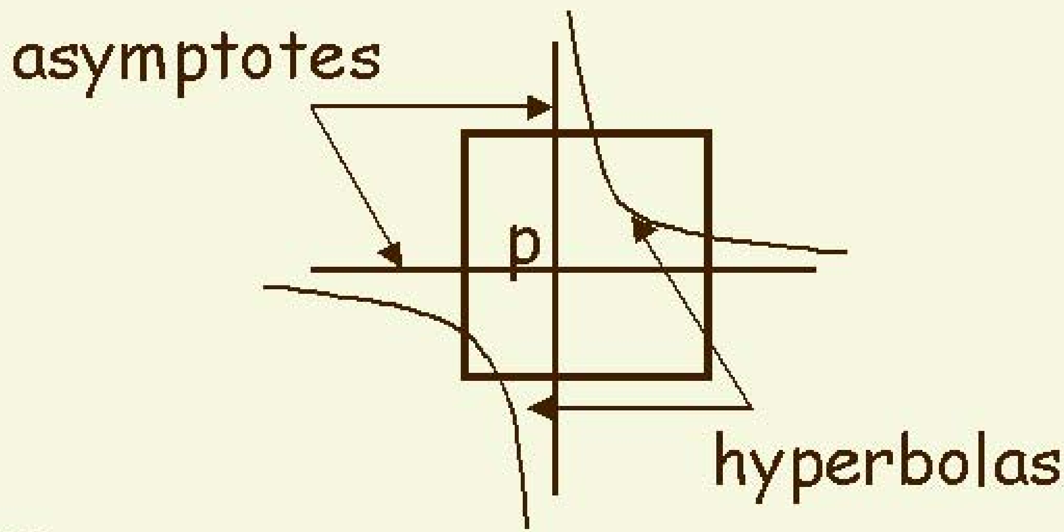


Figure 4: *Two internal configurations for the Marching Cubes configuration 5*

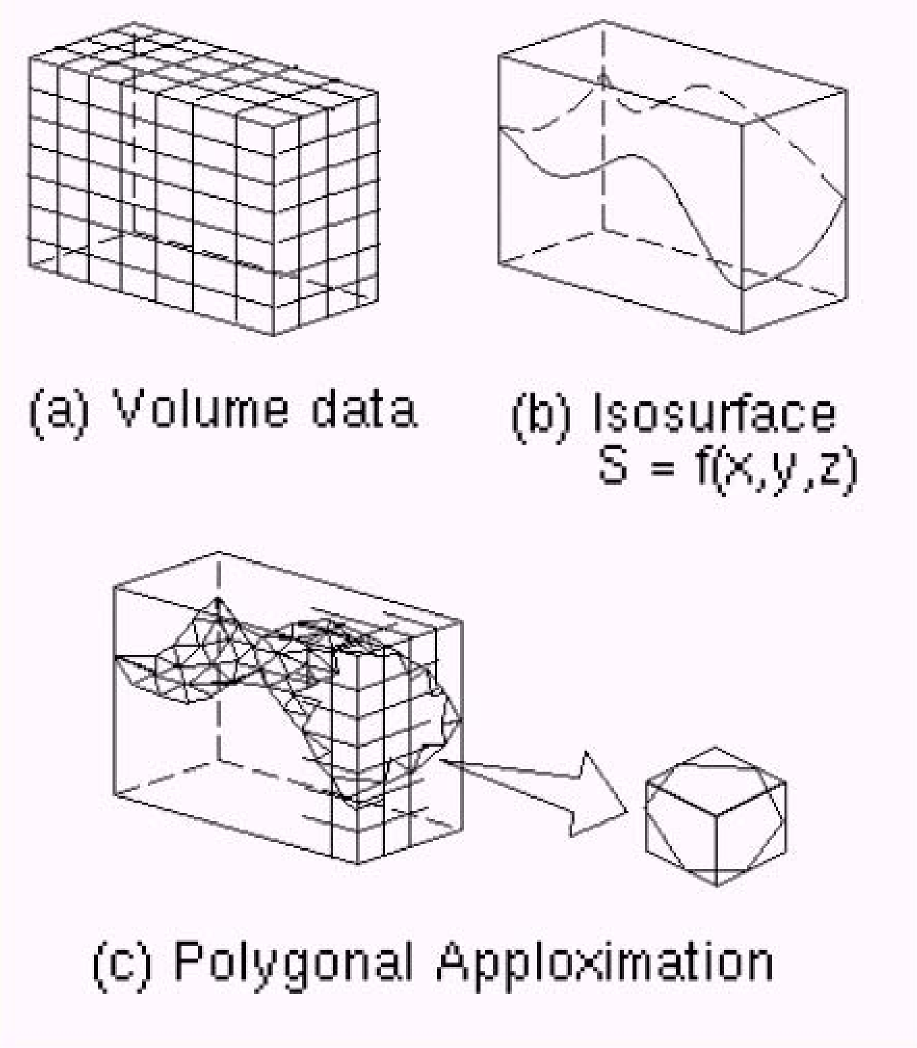
MC 9: Asymptotic Decider

- ✓ 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



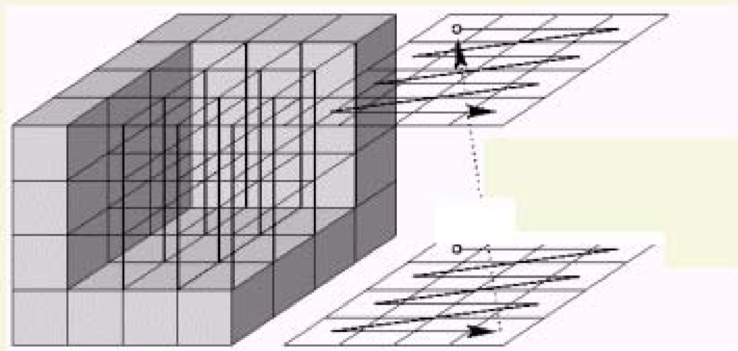
Marching Cubes - Summary 1

- 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.



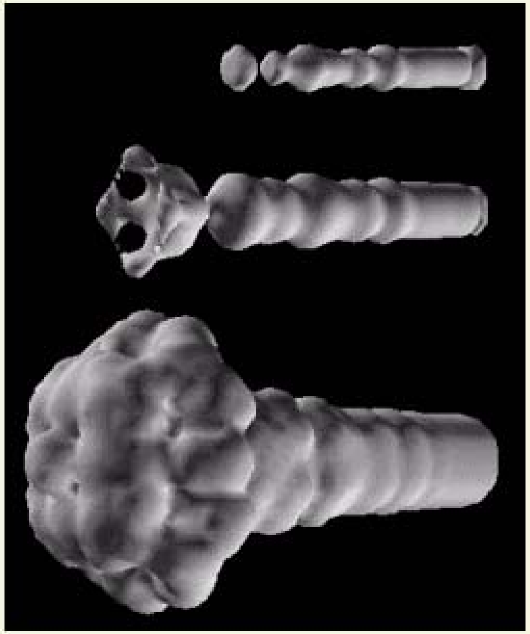
Marching Cubes - Summary 2

- ✓ 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

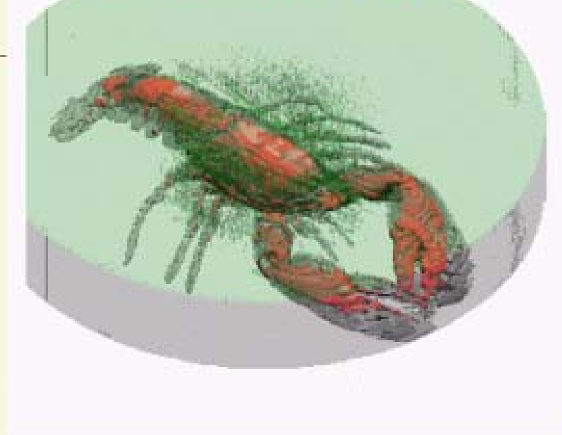


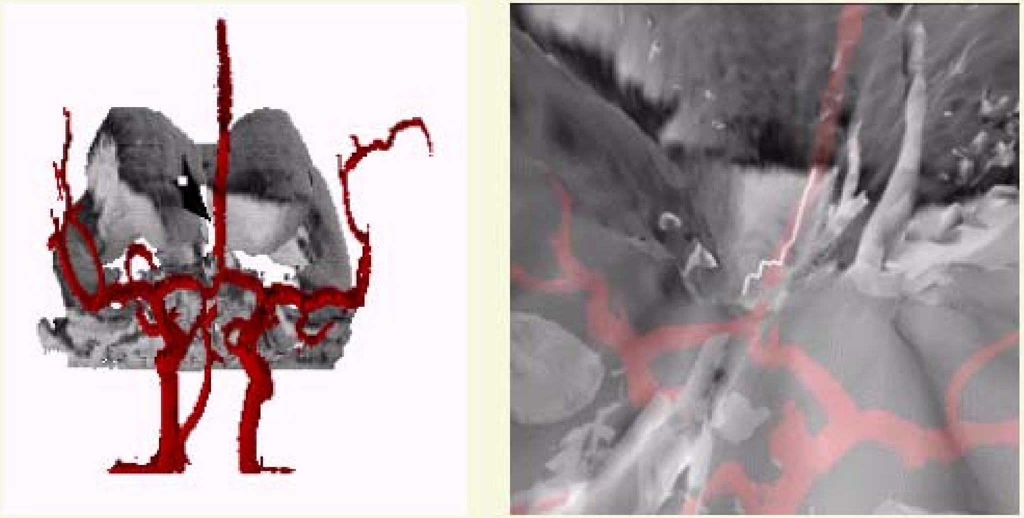
MC Examples

1 Iso-surface



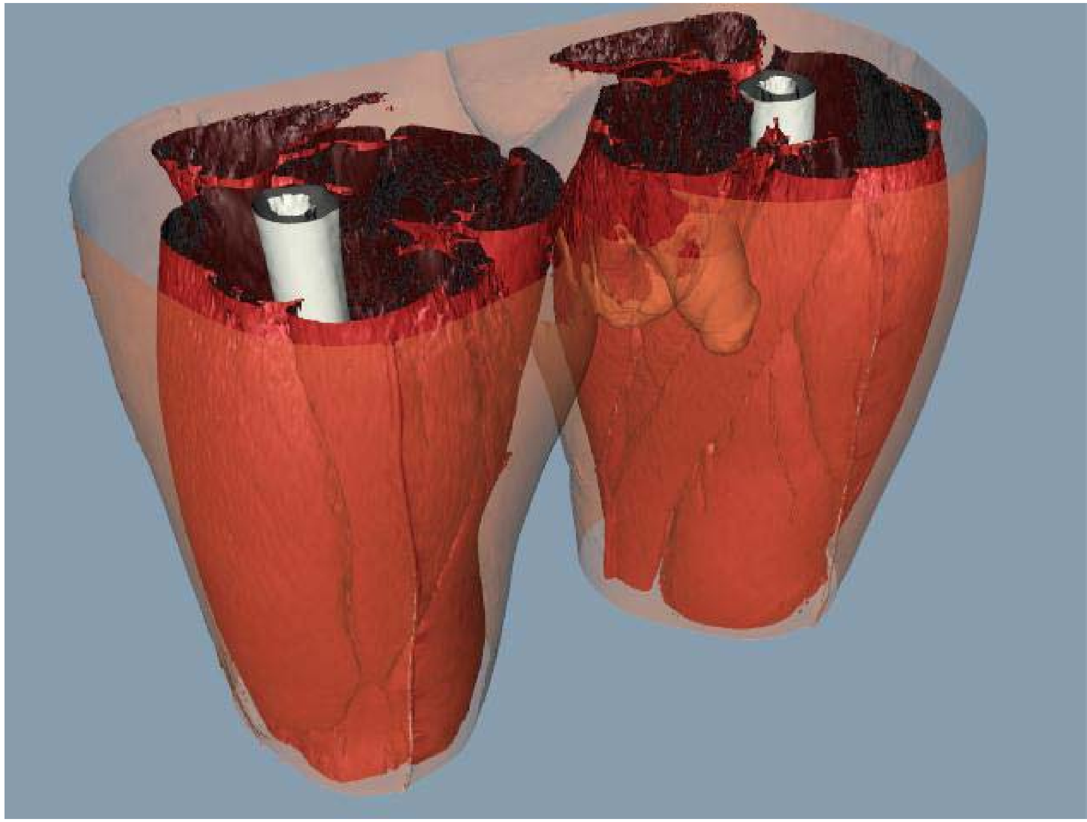
3 Iso-surfaces

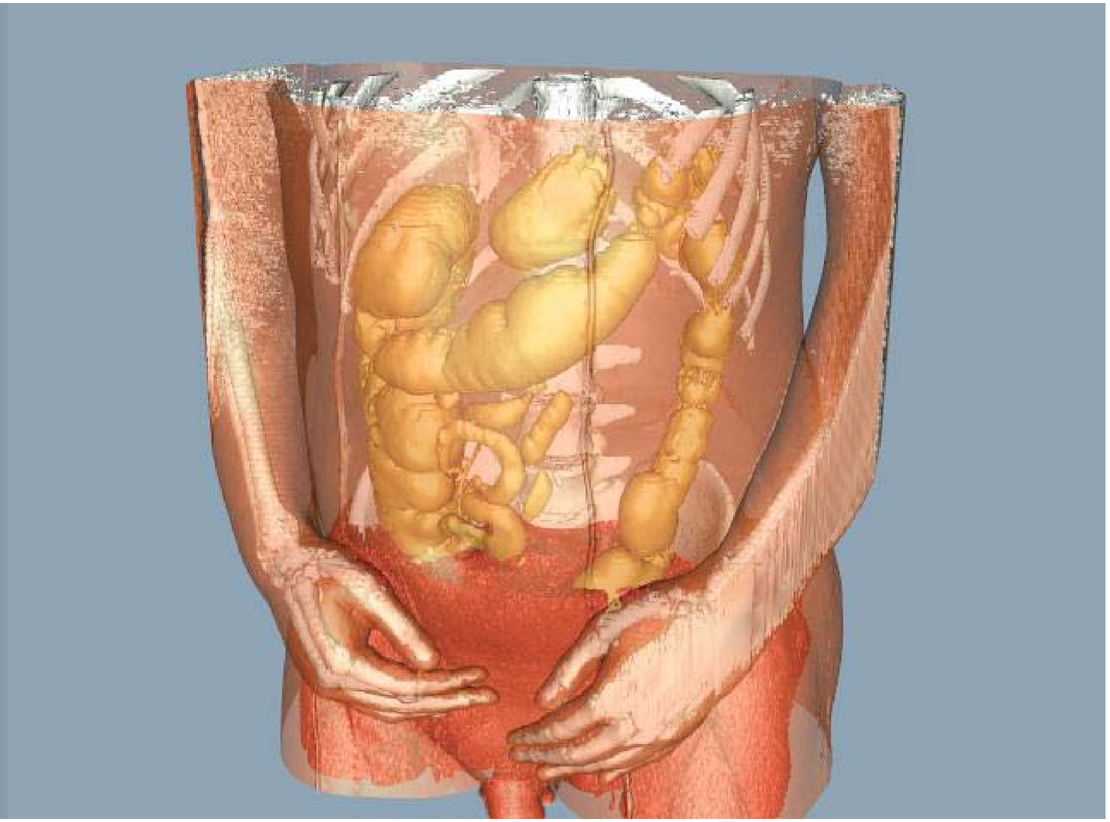


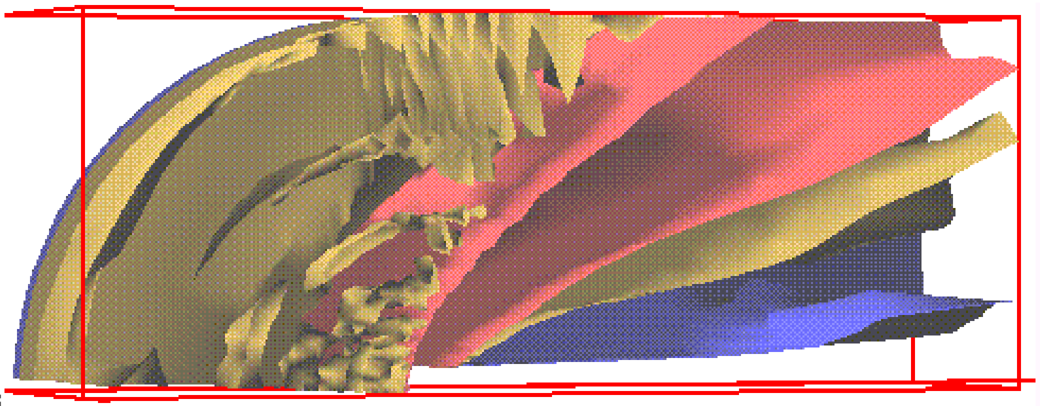


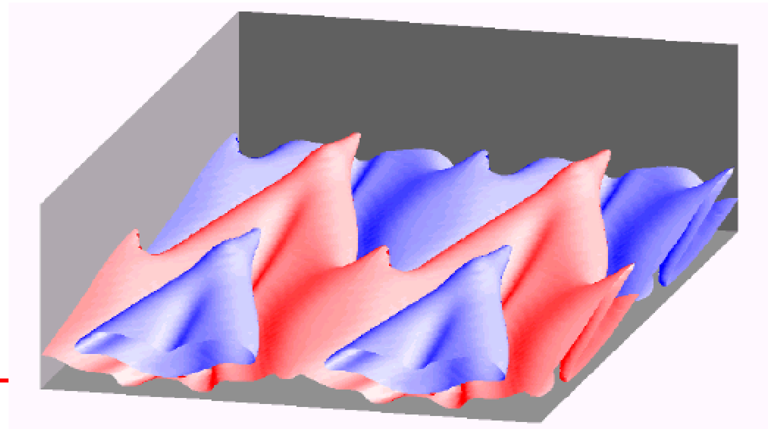
2 Iso-surfaces

Further Examples

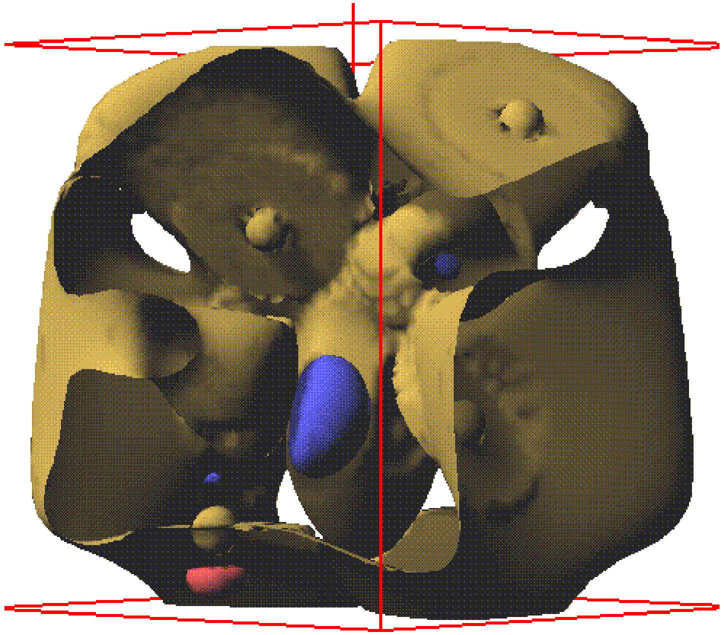


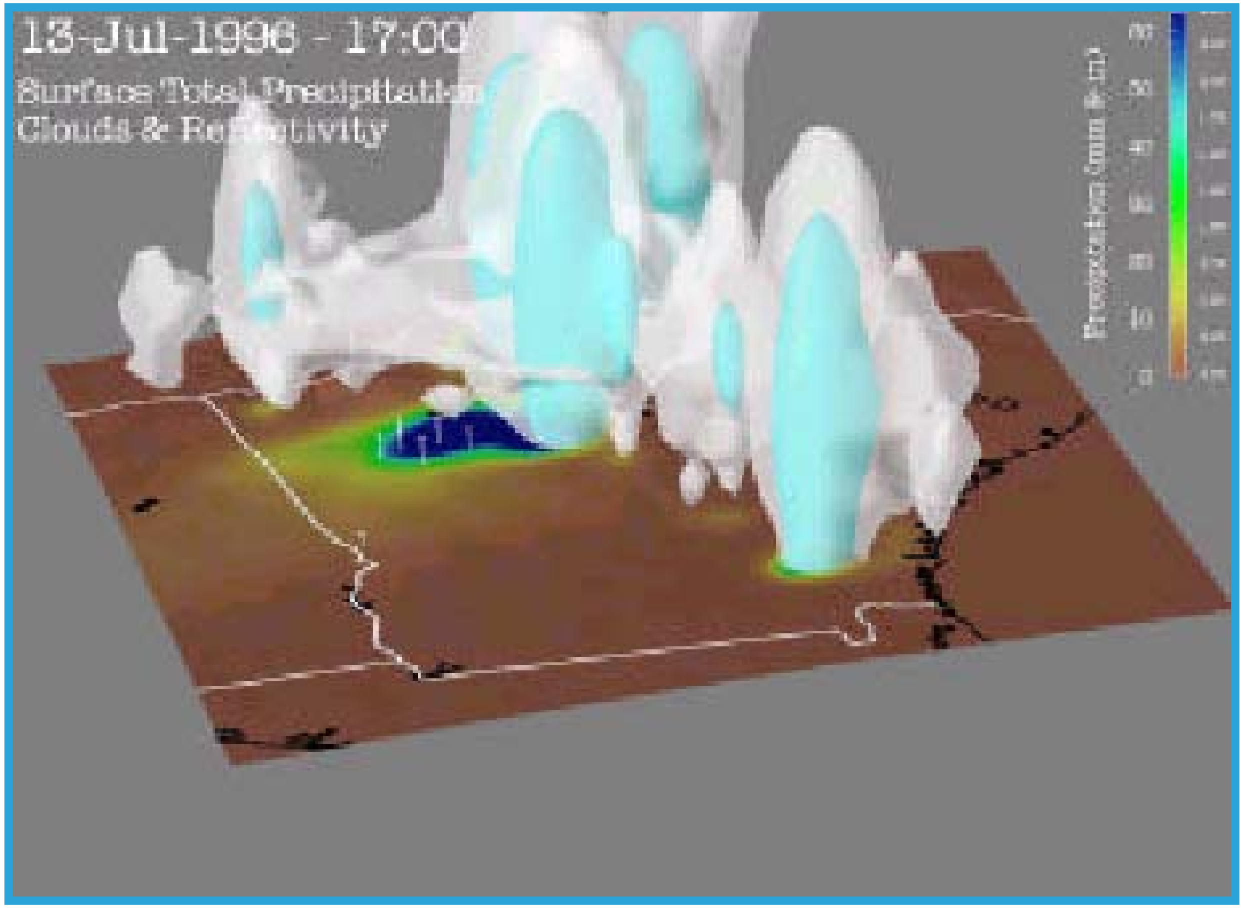


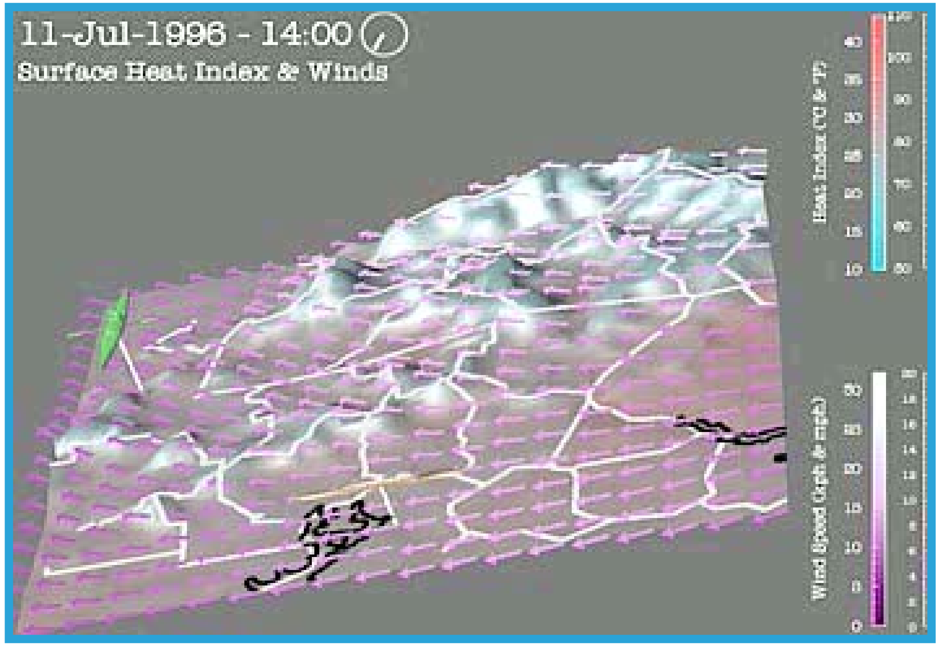




Even Further Examples

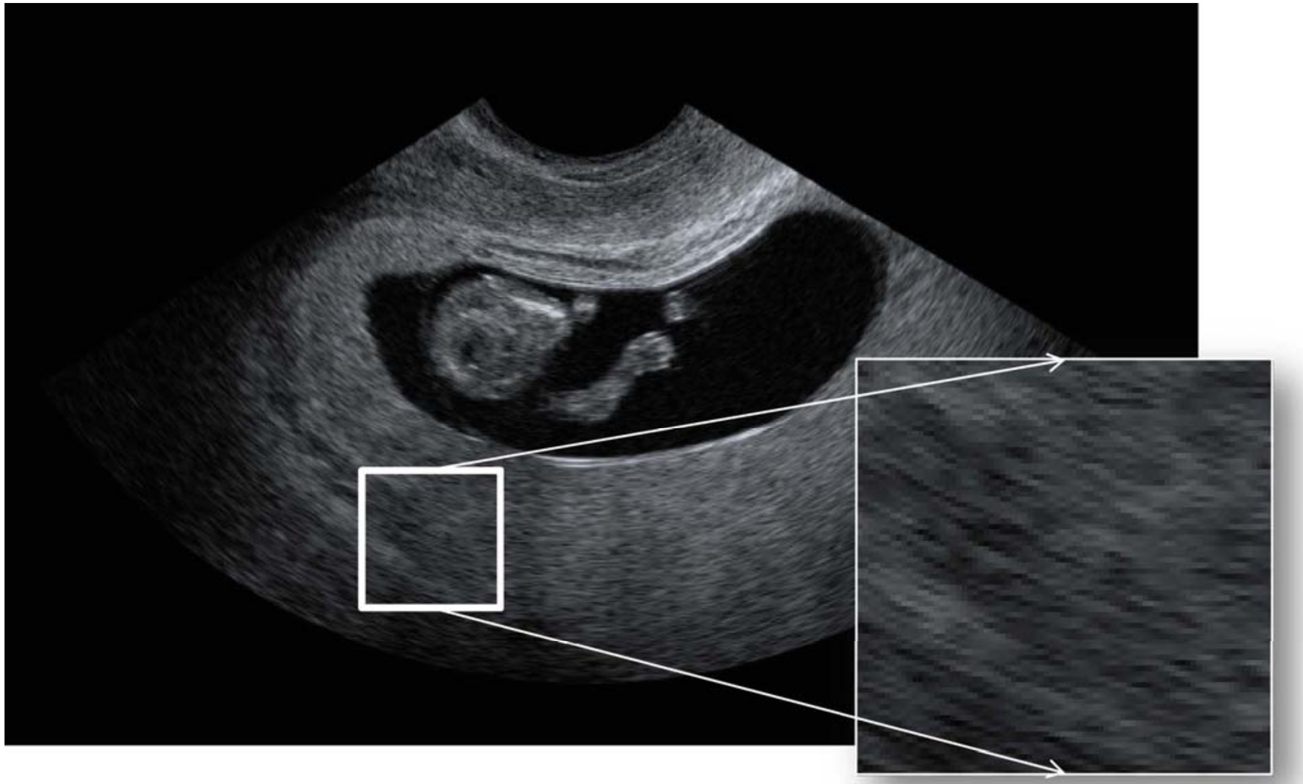






HD*live* [Varchola, Karimov et al.]

- LiveFetoscopic Visualization of 4D Ultrasound Data
- Joint project with Kretztechnik (GE)




3D

NEXT

SKYCAST

Overhead this month in parts of the world

Early October
Comet ISON visible

October 18
Penumbral lunar eclipse

Baby Pictures

"The number one thing parents want to see is if babies have ten fingers and ten toes," says engineer Karl-Heinz Lumpi. His team developed software that shows the digits in full-color 3-D. Beyond allaying parents' curiosity, the more exact image of what's going on in the womb may play a role in diagnostics. Doctors who were formerly resigned to a blurred heartbeat can now see inside that organ's chambers.

It's all in the lighting. The image starts out like a traditional 3-D ultrasound's. Then a computer program adds virtual illumination, mimicking how light plays across human skin—reflecting, casting shadows, and giving shape. As in regular photography, the light source is movable. Plus the image is rotatable, so wriggling fingers, or a floating umbilical cord like this eight-month fetus's, likely won't hinder a thorough exam. —*Johnna Rizzo*

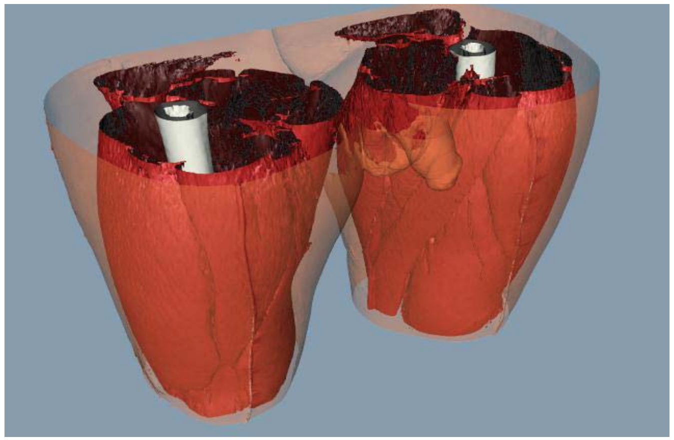
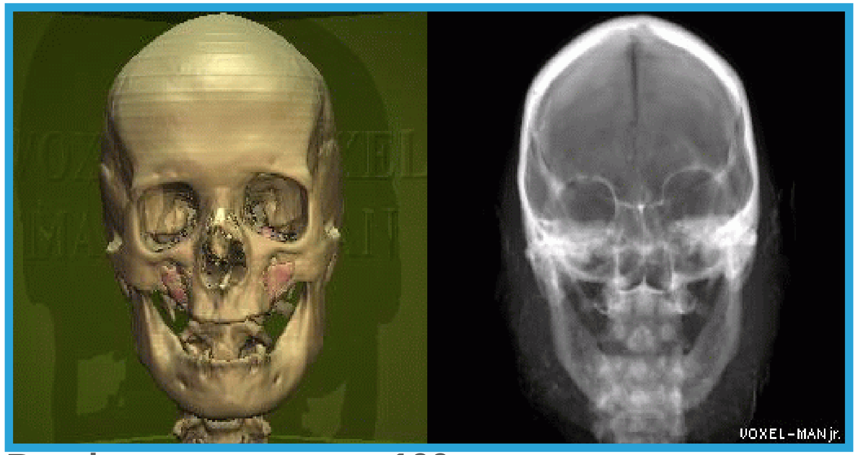
PHOTO: BERNARD BENOIT, SCIENCE SOURCE

Conclusion
Volume Visualization

General Remarks

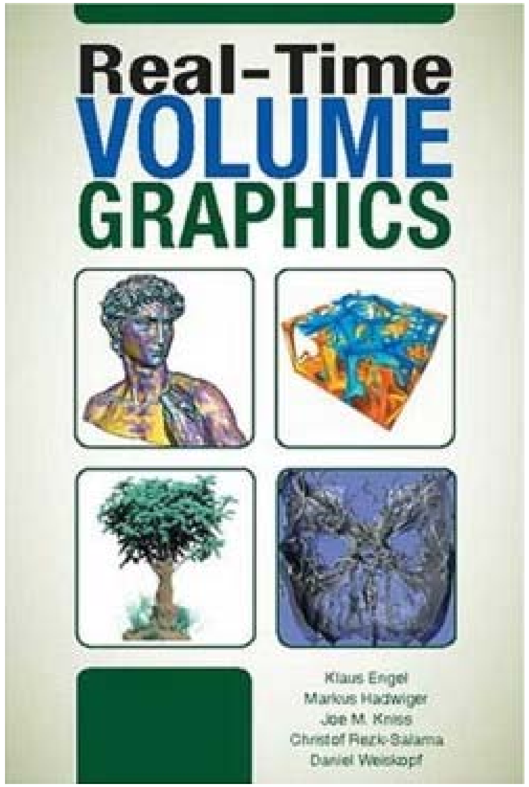
Surface vs. Volume Rendering

- 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/
- **The Virtual Autopsy Table** (https://www.youtube.com/watch?v=bws6vWM1v6g)
- **VISUAPPS** (http://www.visuapps.com/)



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)
  - Roger Crawfis (Ohio State Univ.)
  - Hanspeter Pfister (MERL)
  - Dirk Bartz
  - Markus Hadwiger
  - Christof Rezk Salama