2IMV20

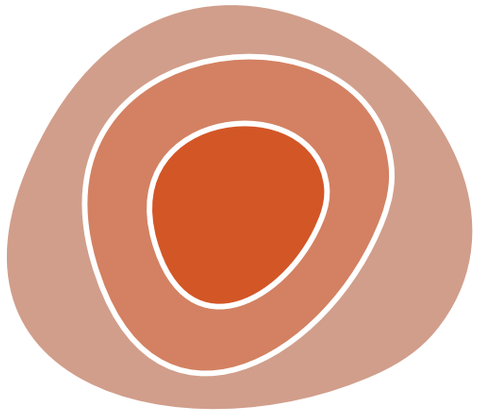
Encoding: spatial data

Technische Universiteit
Eindhoven
University of Technology

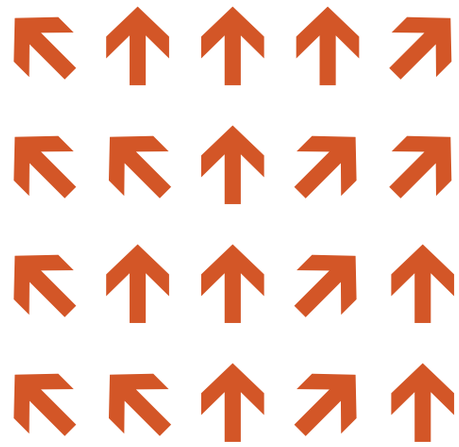
Recap

Encoding: spatial data

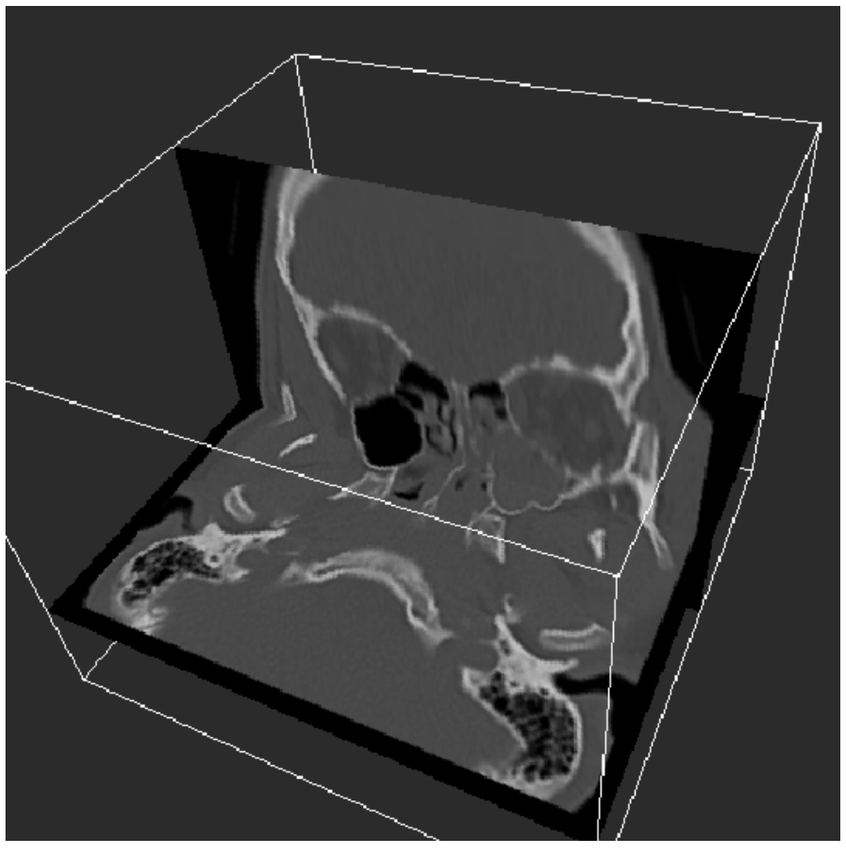
- Scalar fields (one value per cell)
  - isocontours
  - (direct) volume rendering



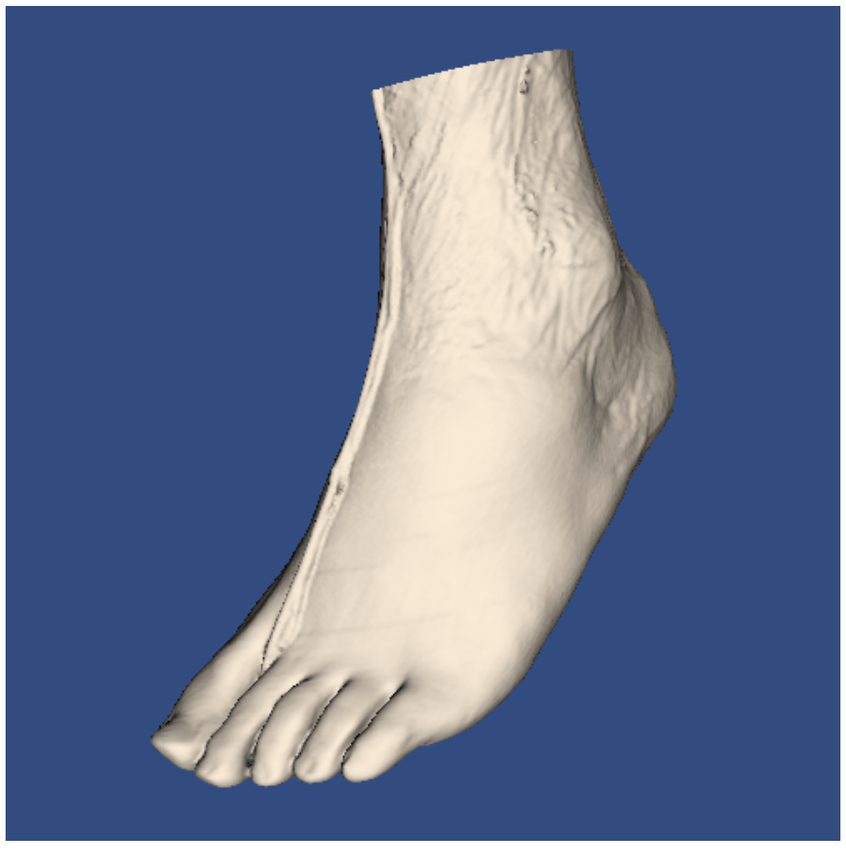
- Vector and tensor fields (many values per cell)



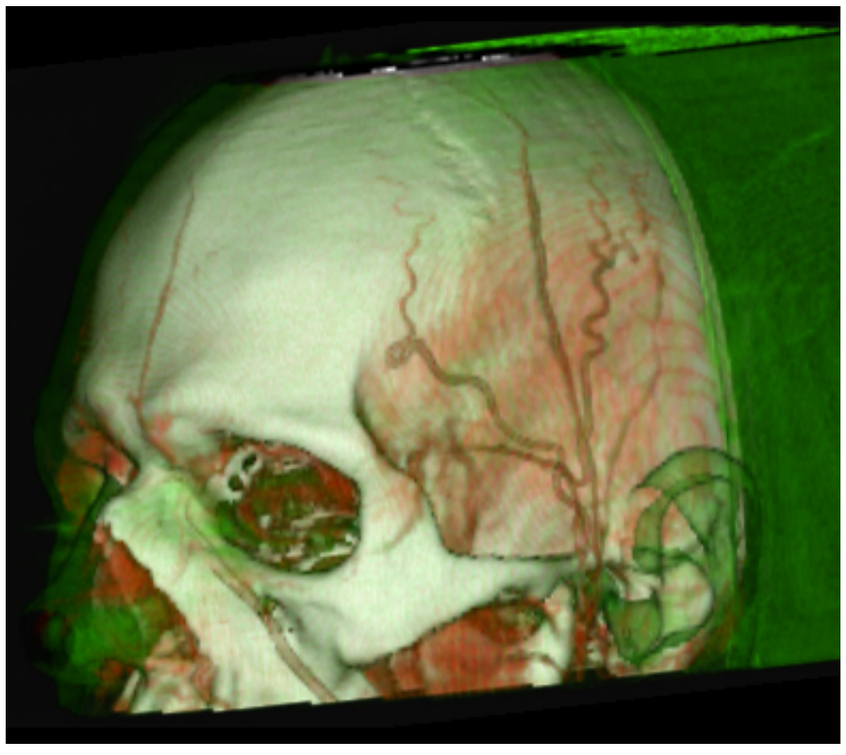
Volume rendering methods



Slice-by-slice



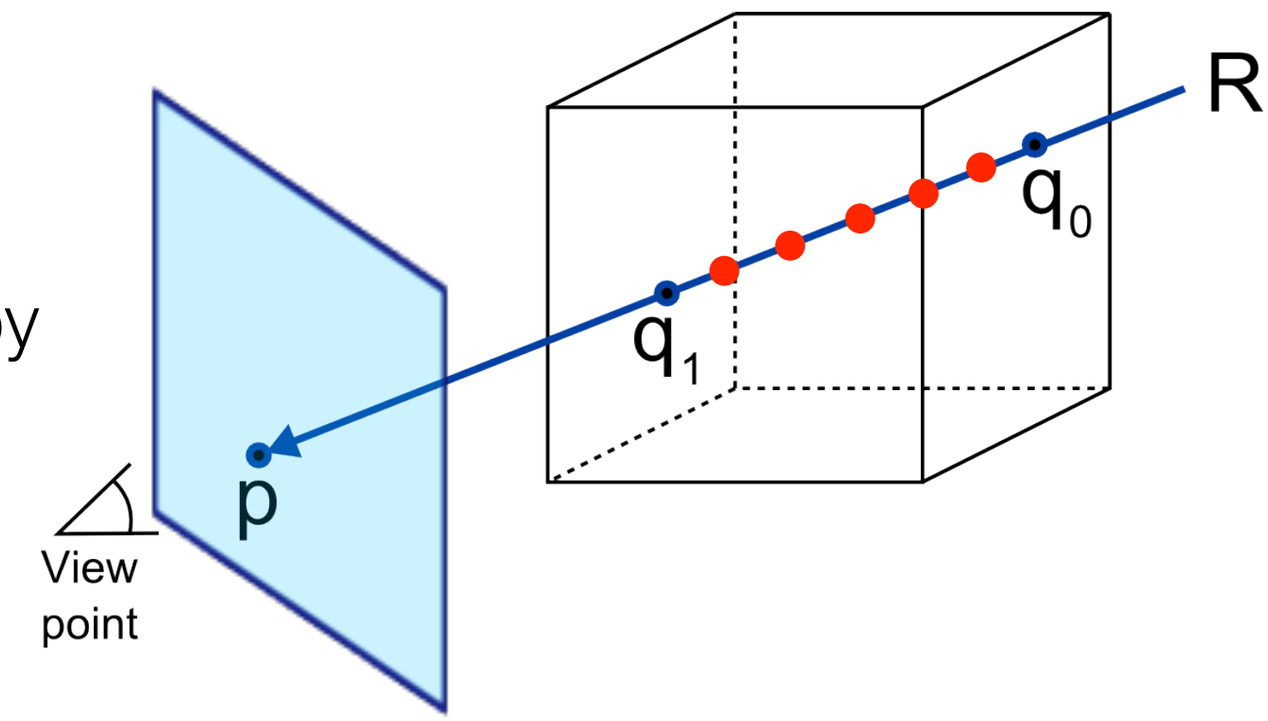
Isosurfaces



Direct volume rendering

Direct volume rendering: ray casting

- For every pixel **p** in final image, cast ray **R** perpendicular to view plane
- Combine samples **q** on the ray by a ray function **F**

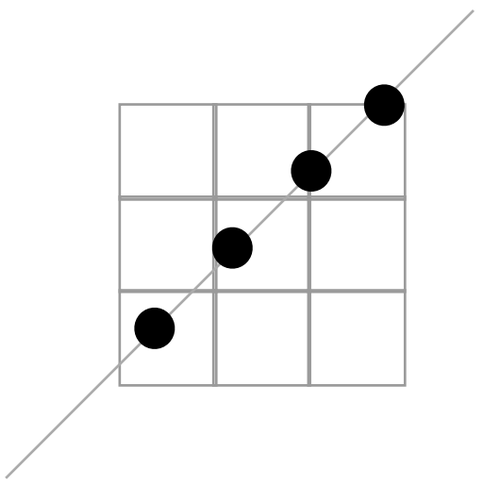


R parametrization $$q_t = q_0 + t(q_1 - q_0) \quad t \in [0, 1]$$

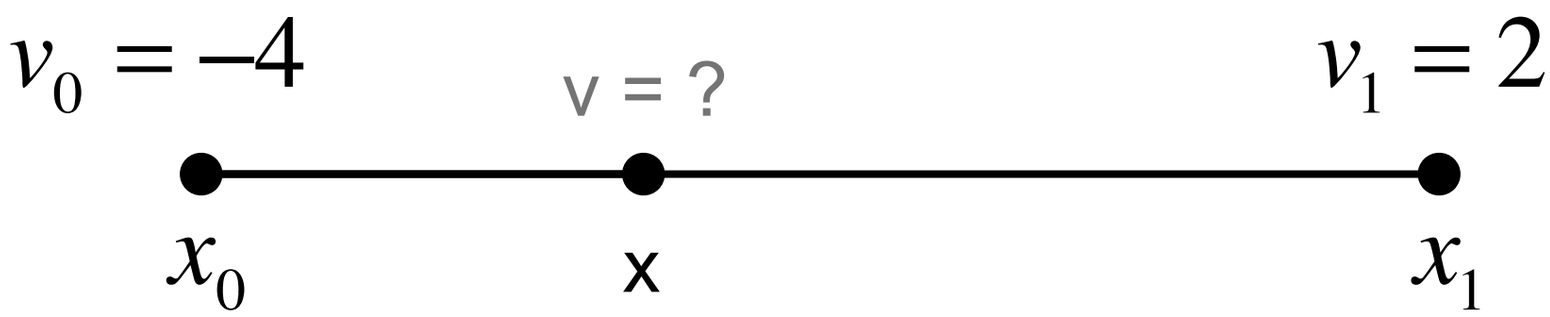
scalar values along R $$s_t = f(q_t)$$

pixel value in p $$I(p) = F(s_t)$$

Interpolation



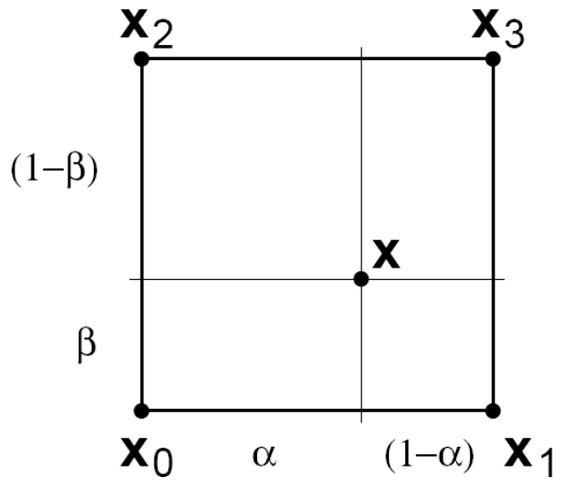
Linear interpolation

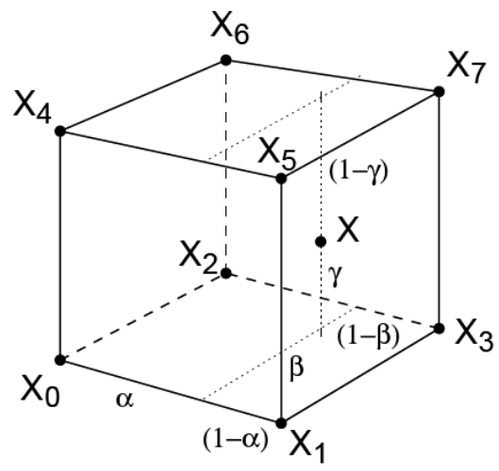


$$\alpha = \frac{x - x_0}{x_1 - x_0}$$

$$v = (1 - \alpha)v_0 + \alpha v_1$$

Bi-linear & Tri-linear interpolation



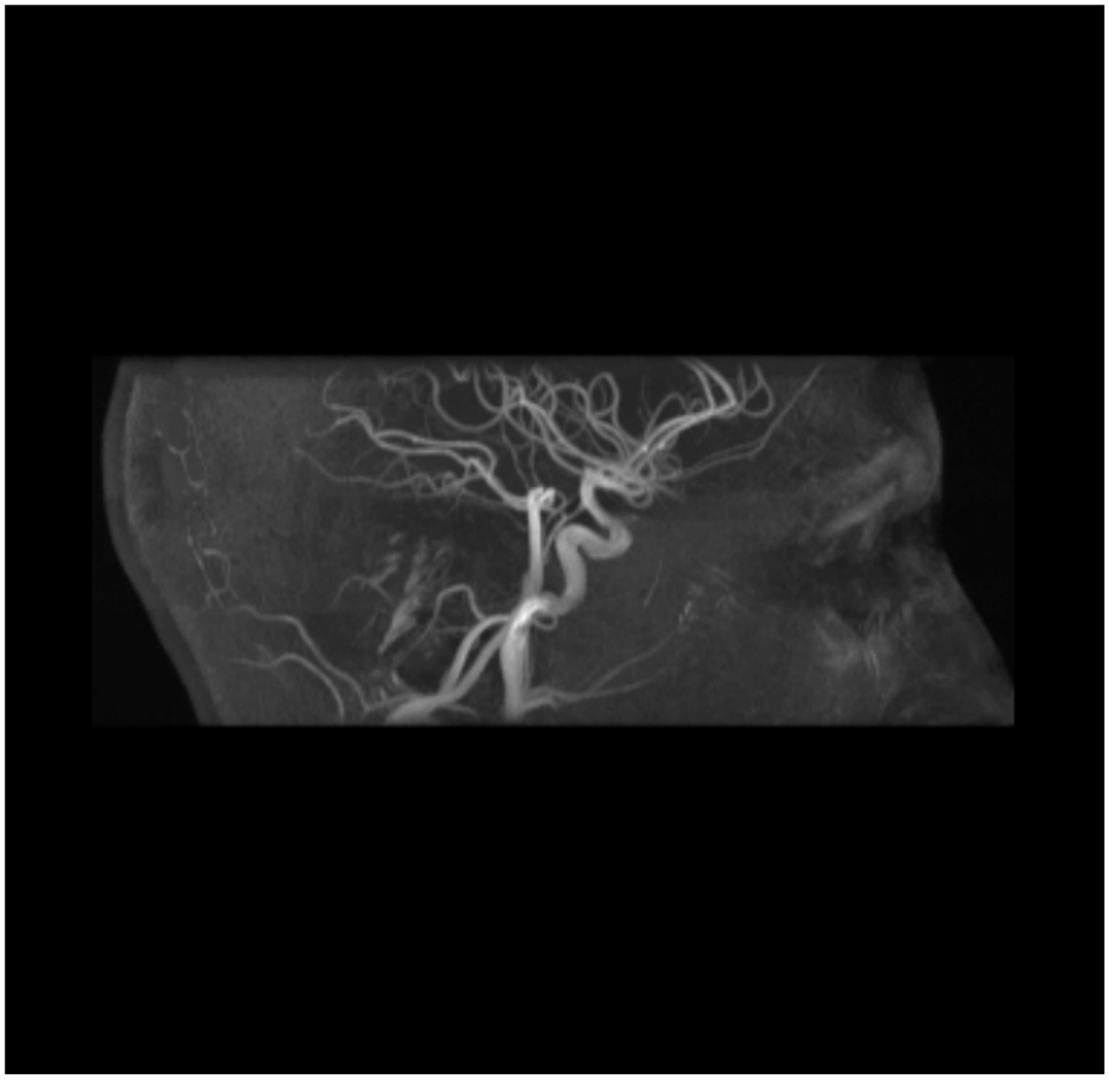


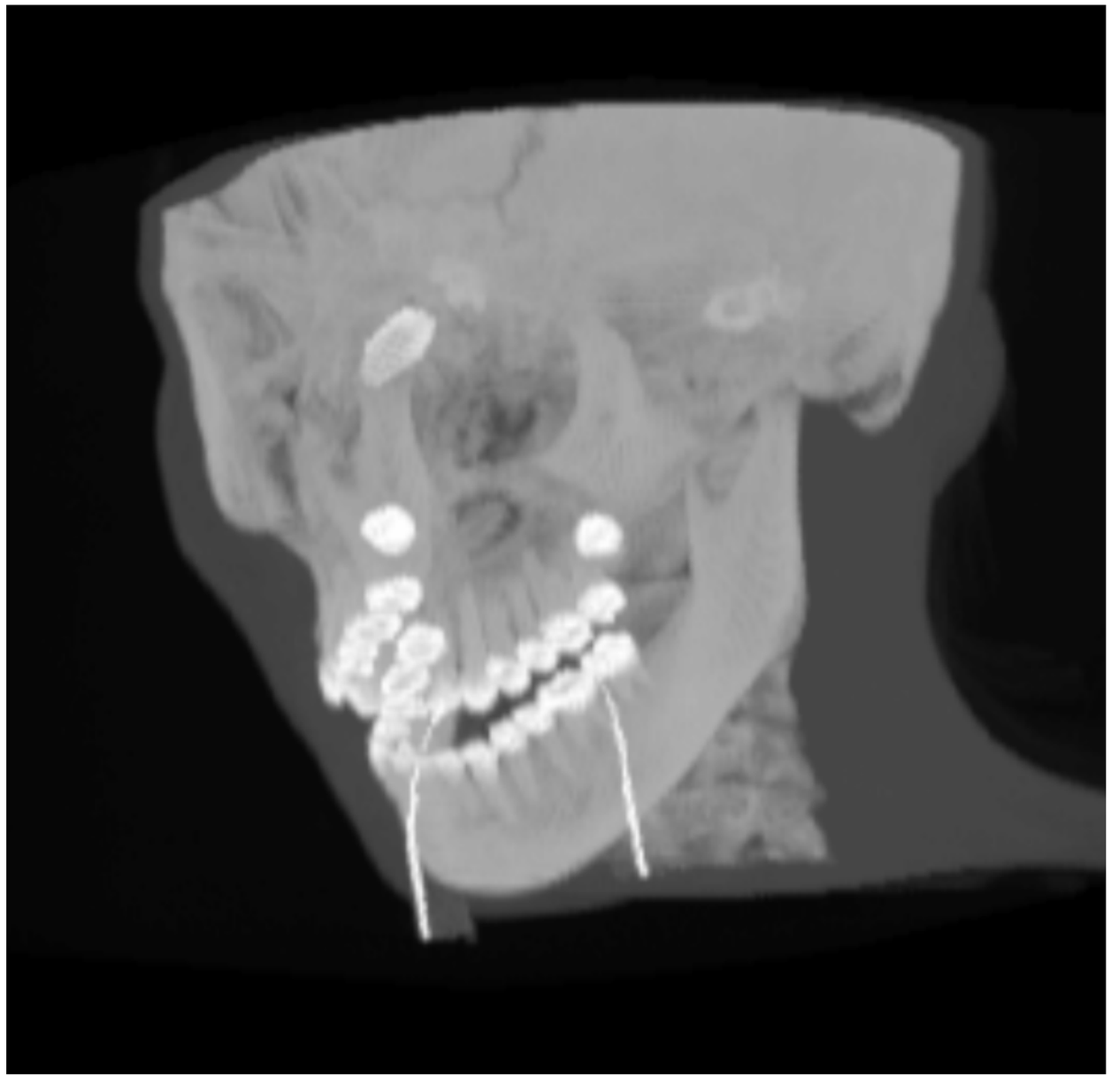
$$
\begin{aligned}
S_X &= (1-\alpha)(1-\beta)S_{X_0} + \alpha(1-\beta)S_{X_1} \\
&\quad +(1-\alpha)\beta S_{X_2} + \alpha\beta S_{X_3}
\end{aligned}
$$

$$
\begin{aligned}
S_X &= (1-\alpha)(1-\beta)(1-\gamma)S_{X_0} + \alpha(1-\beta)(1-\gamma)S_{X_1} \\
&\quad +(1-\alpha)\beta(1-\gamma)S_{X_2} + \alpha\beta(1-\gamma)S_{X_3} \\
&\quad +(1-\alpha)(1-\beta)\gamma S_{X_4} + \alpha(1-\beta)\gamma S_{X_5} \\
&\quad +(1-\alpha)\beta\gamma S_{X_6} + \alpha\beta\gamma S_{X_7}
\end{aligned}
$$

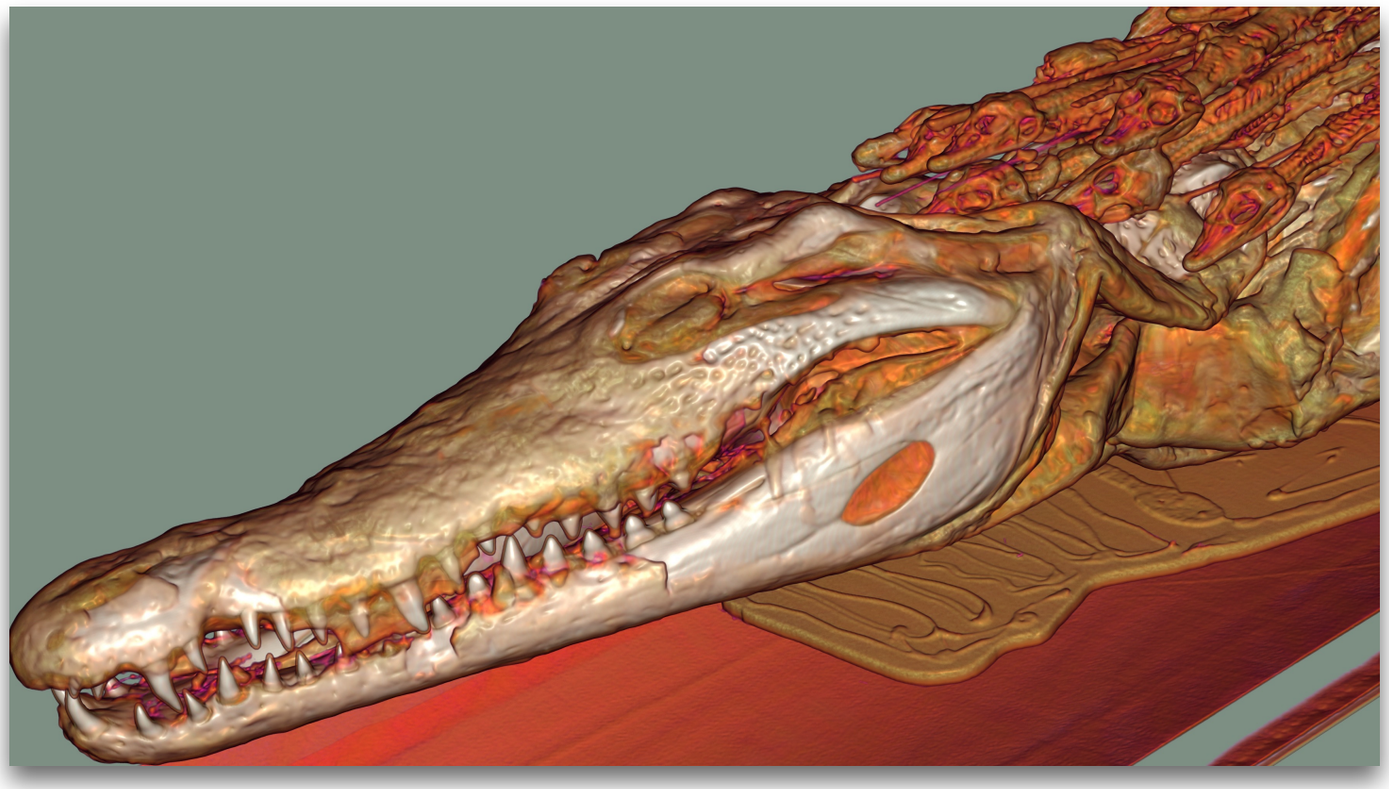
Maximum intensity projection

- Take for F maximum operator: $I(p) = \max_t s_t$

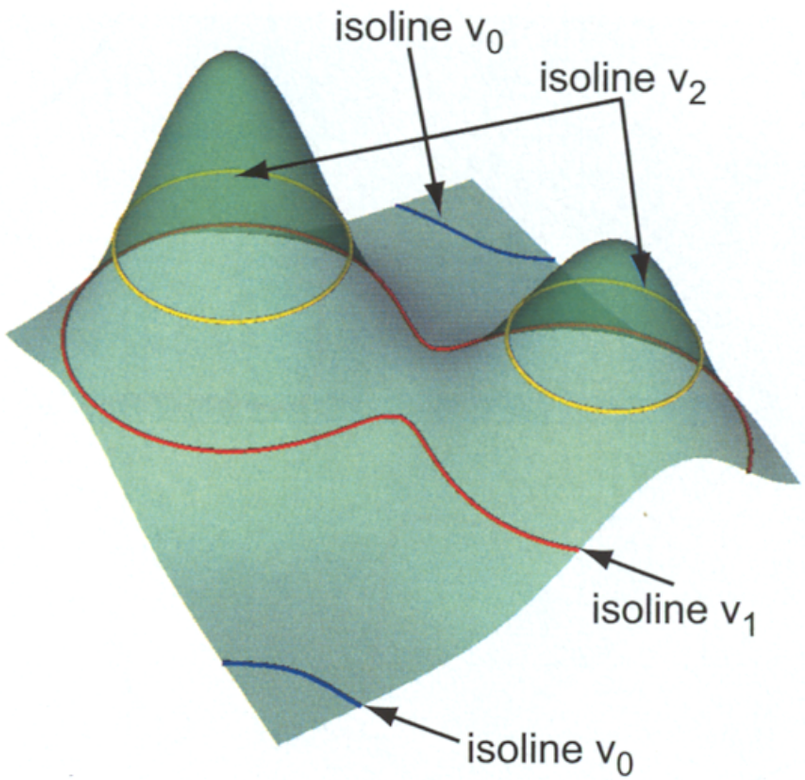




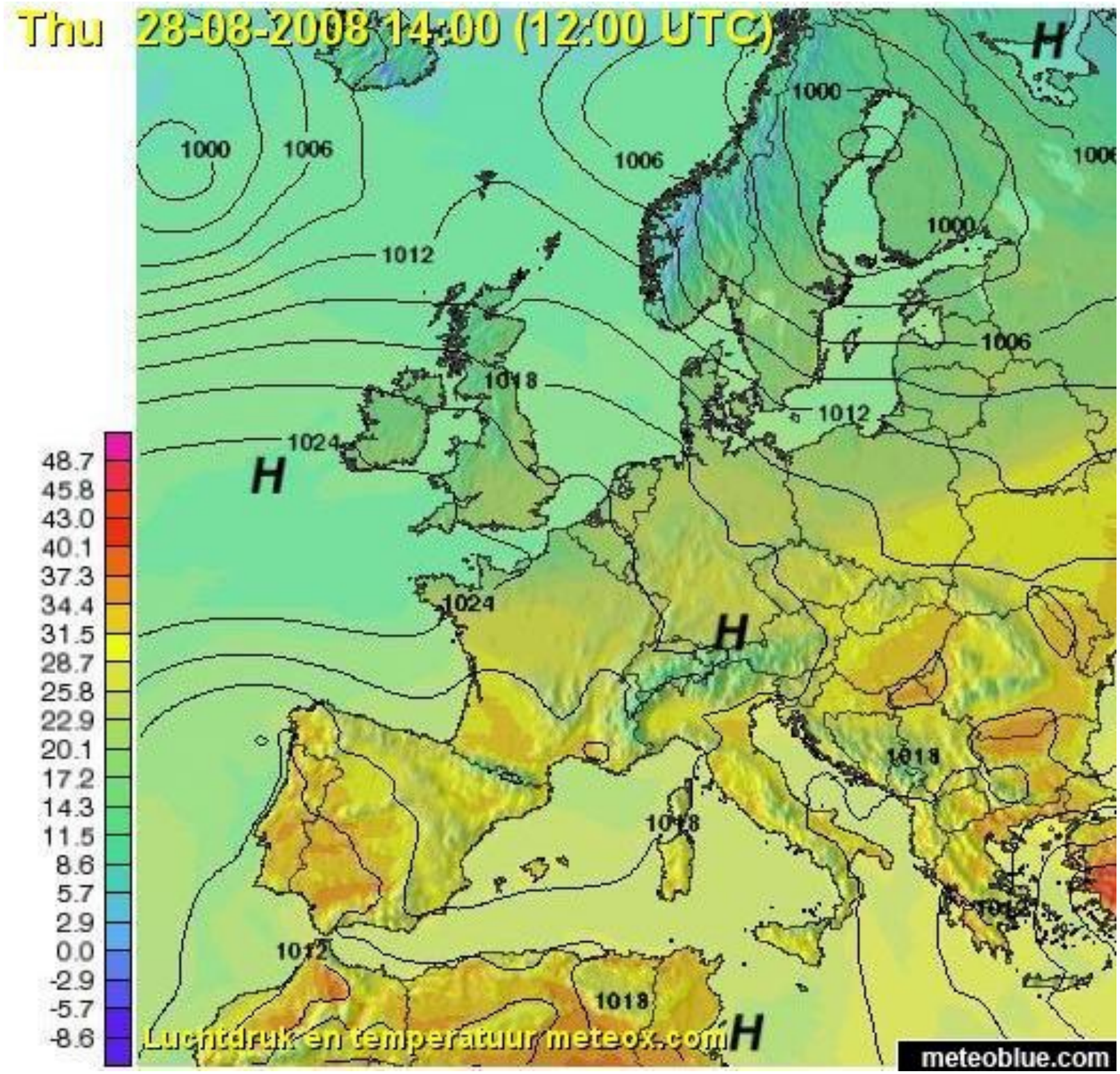
Towards advanced approaches



Contour (iso) lines

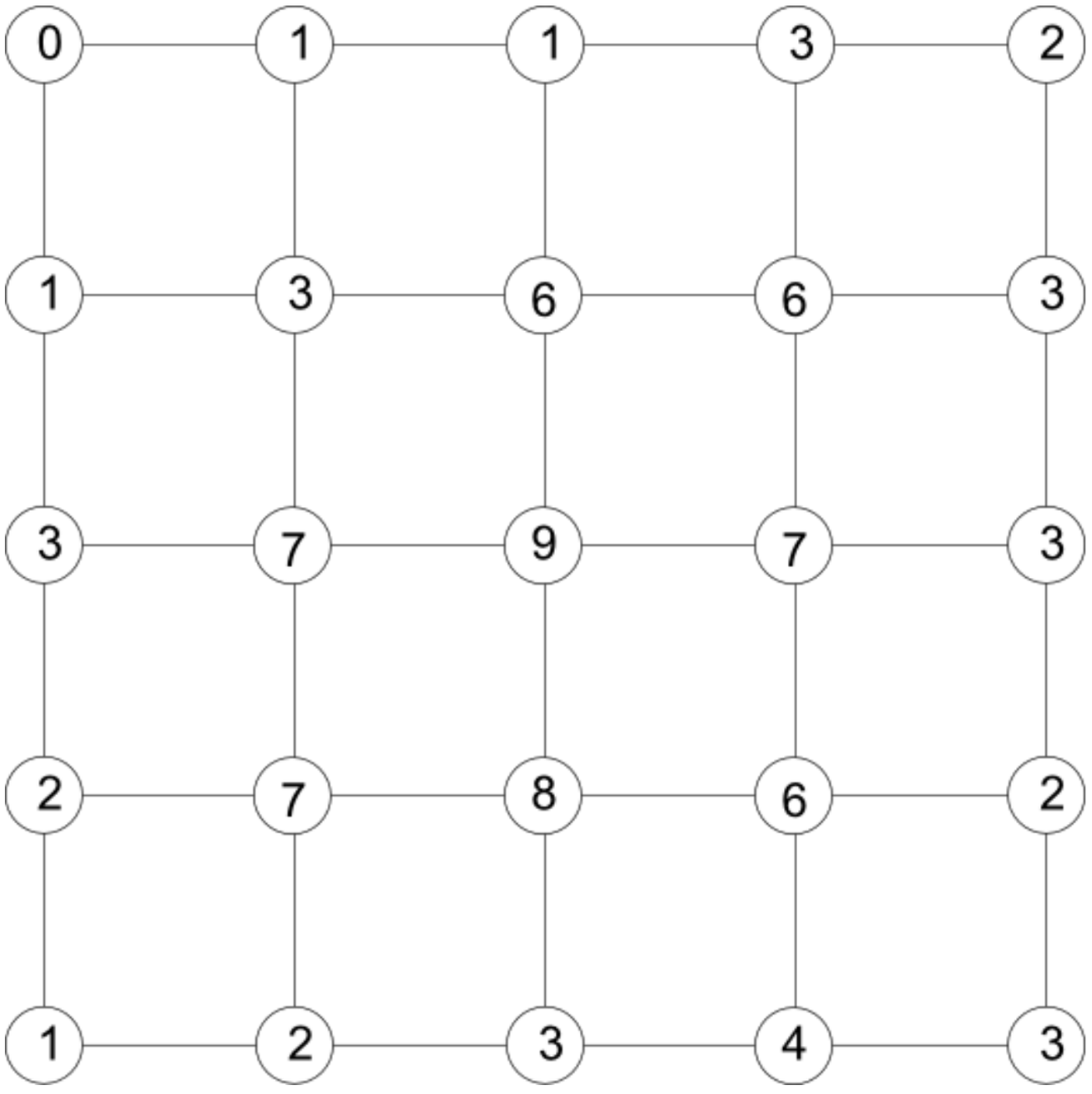


$$f(x, y) = c$$



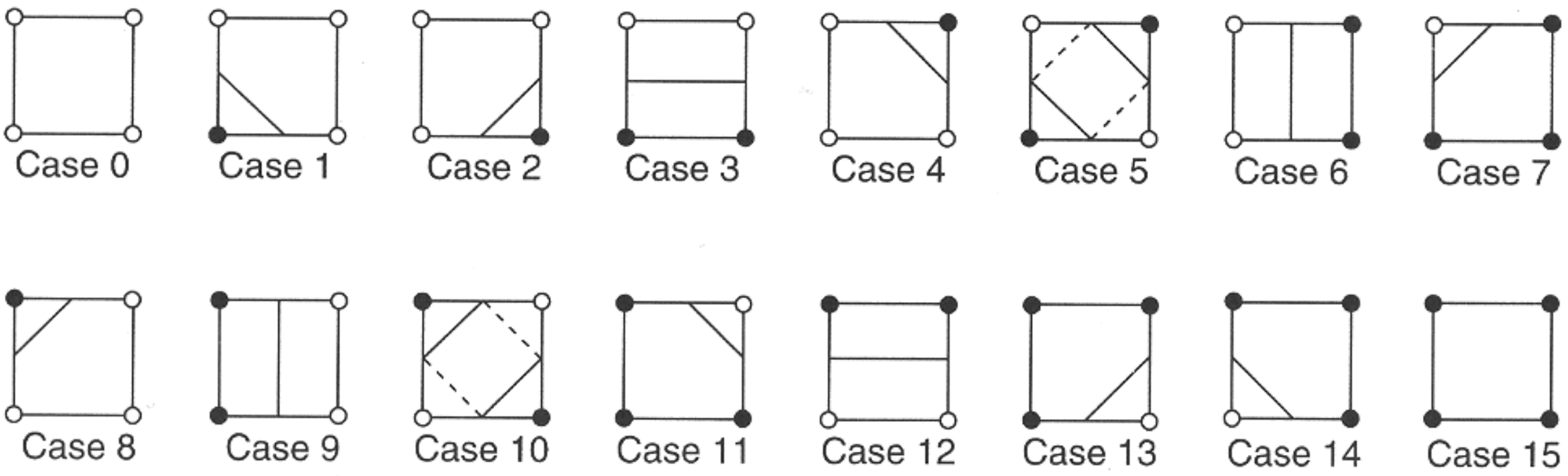
Contour lines in discrete data?

Draw $f(x, y) = 5$



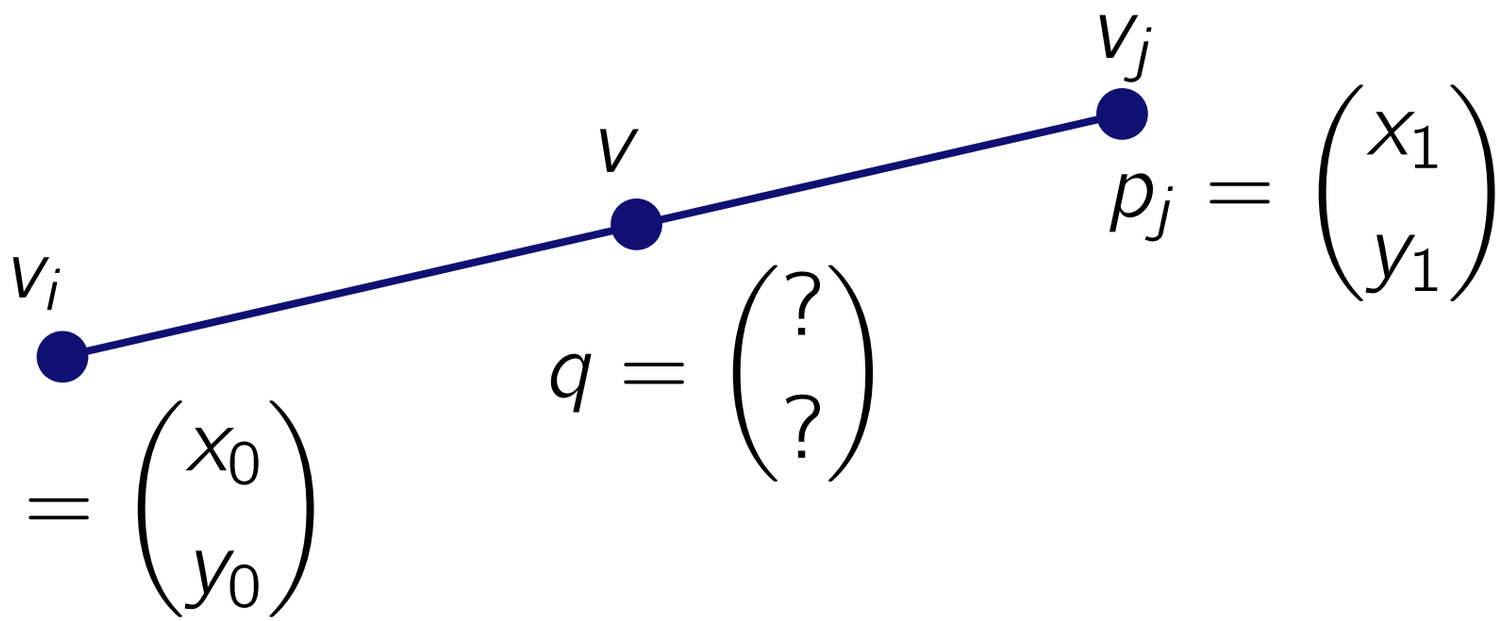
Marching Squares algorithm

- Assumption
  - contour can pass cell in only **finite** number of ways
- Case table
  - enumerate **how** contour passes cell (topology) not **where** (geometry)
  - 4 corner vertices make up $2^4 = 16$ states



Marching squares: exact intersection point

- Compute exact intersection point along edges by linear interpolation

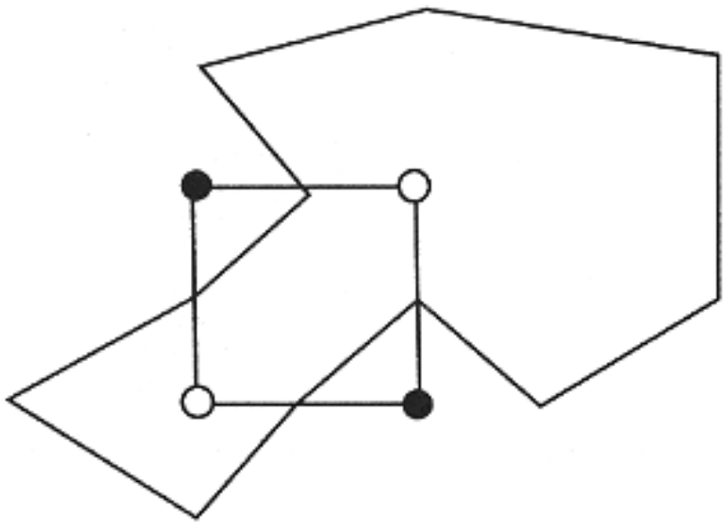


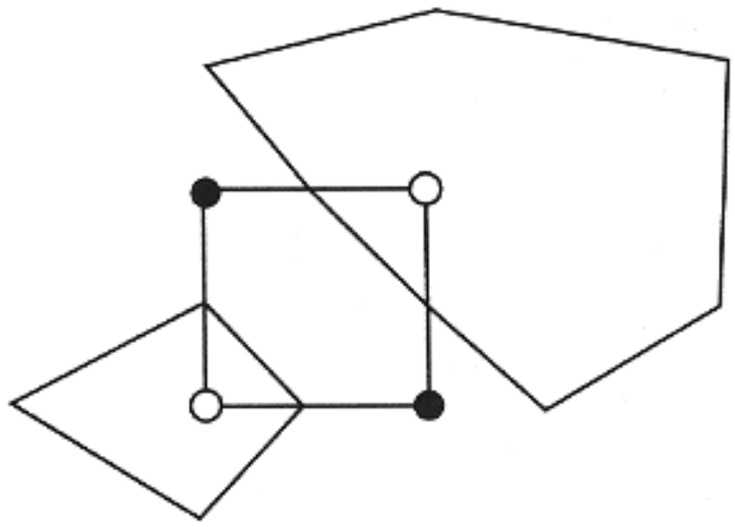
parametric line segment representation: $q = p_i + t(p_j - p_i)$

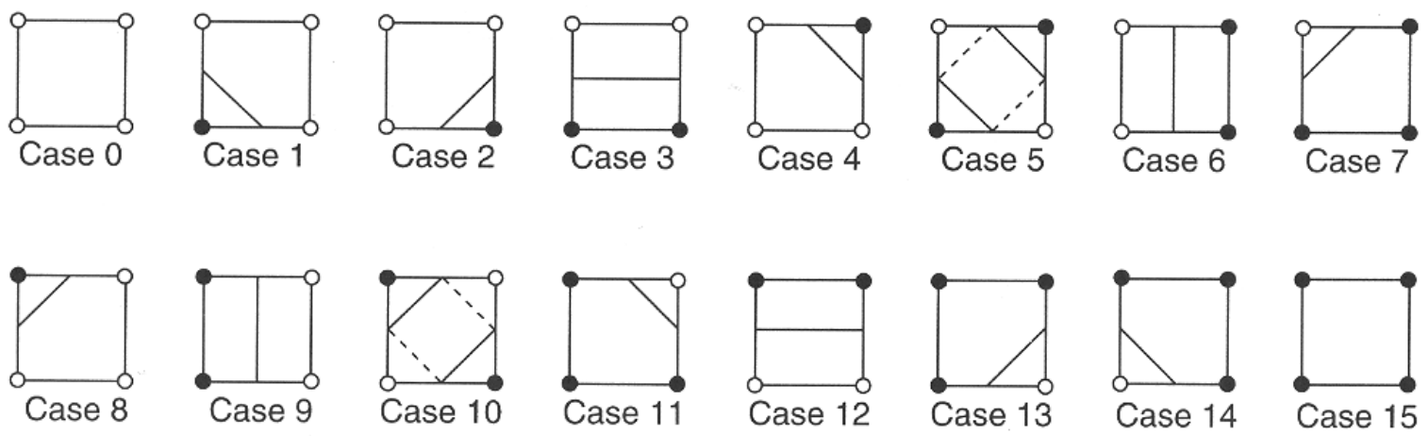
$$t = \frac{v - v_i}{v_j - v_i}$$

Handling ambiguous case: option 1

- Cases 5 and 10 can be contoured in two ways
- Either **extend** or **break** contour

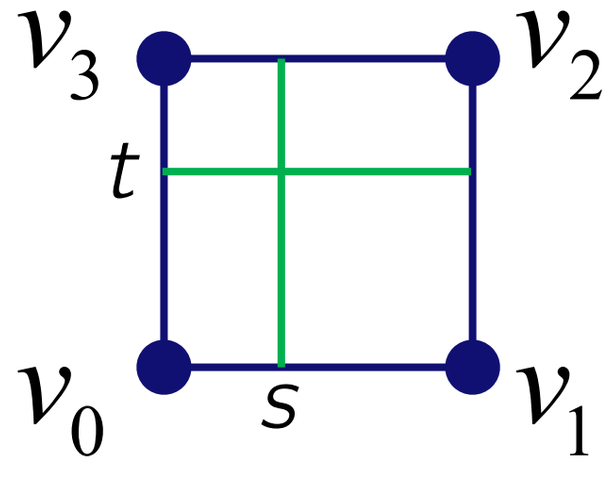






Handling ambiguous case: option 2

- Asymptotic decider
- Assume that scalar value varies bilinearly across cell

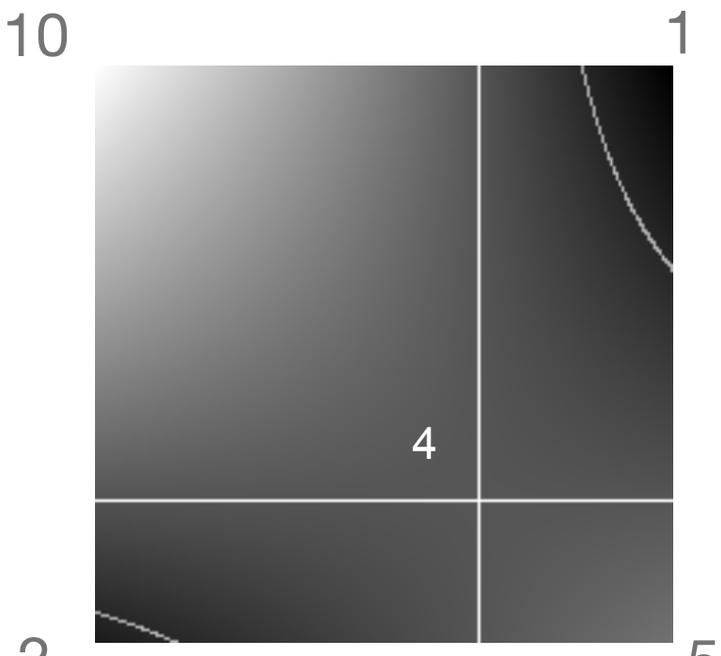


$$F(s,t) = (1-s)(1-t)v_0 + s(1-t)v_1 + stv_2 + (1-s)tv_3$$
$$= (v_0 + v_2 - v_1 - v_3)st + (v_1 - v_0)s + (v_3 - v_0)t + v_0$$
$$= Ast + Bs + Ct + D$$

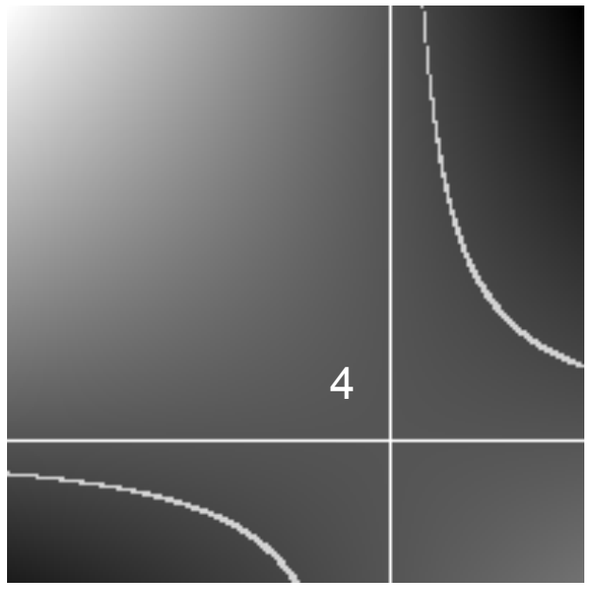
$F(s,t) =$ isovalue is a **hyperbola**

Nielson et al., The asymptotic decider: resolving the ambiguity in marching cubes. Proc. 2nd conf. Visualization, pp. 83-91, 1991.

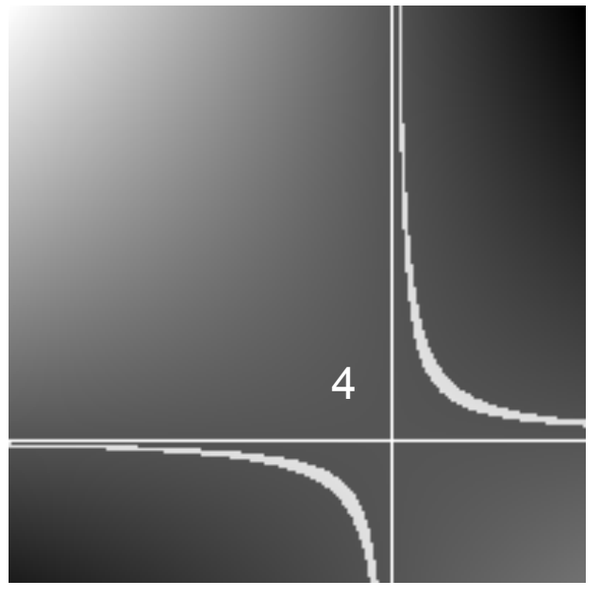
Asymptotic decider: hyperbolas



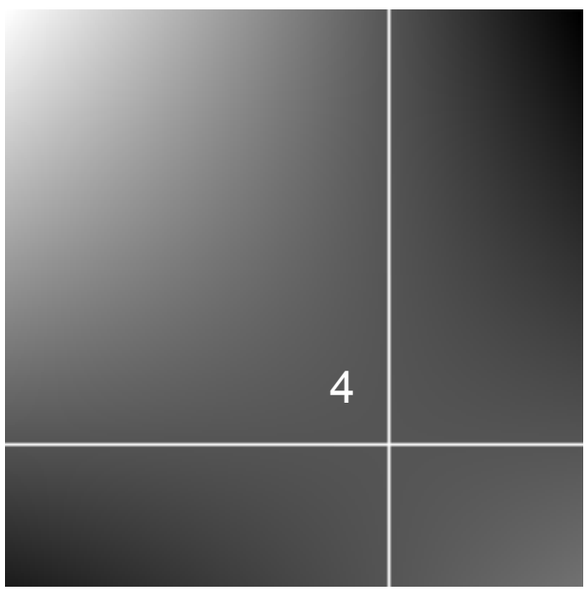
F(s,t) = 2.4



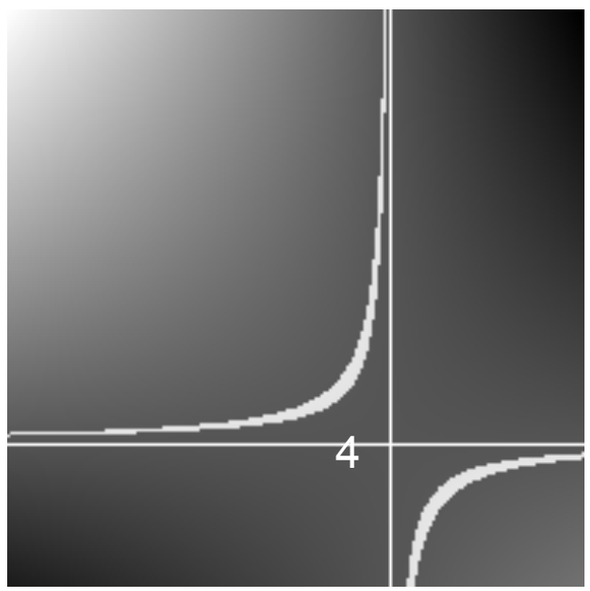
F(s,t) = 3.5



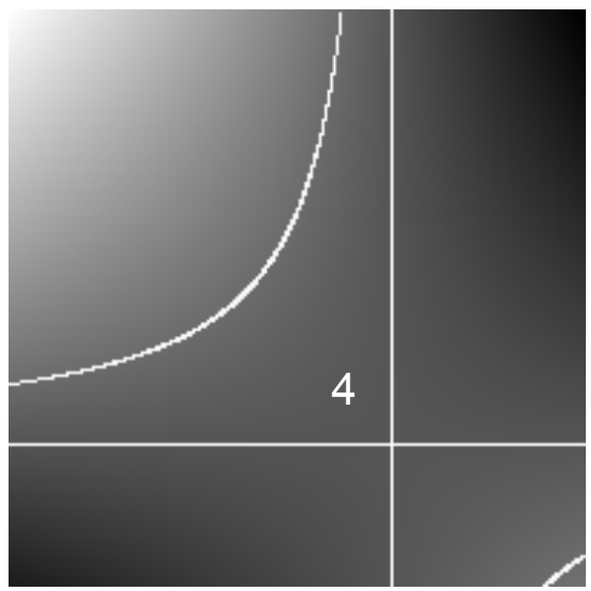
F(s,t) = 3.9



F(s,t) = 4.0



F(s,t) = 4.1



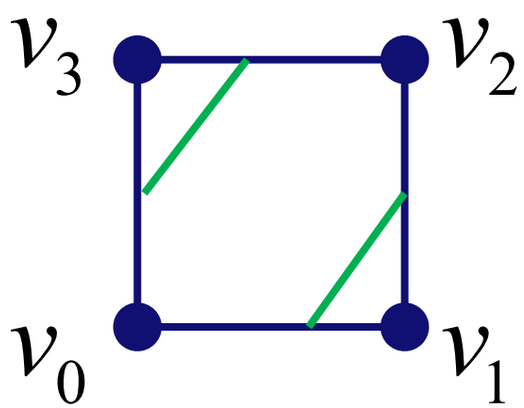
F(s,t) = 4.8

Asymptotic decider

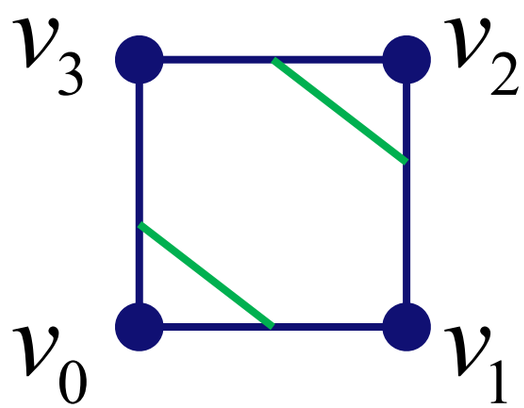
- Compare isovalue against value of interpolant at intersection of asymptotes

$$s_a = -\frac{C}{A} \qquad t_a = -\frac{B}{A}$$

$$F(s_a, t_a) = \frac{AD - BC}{A}$$



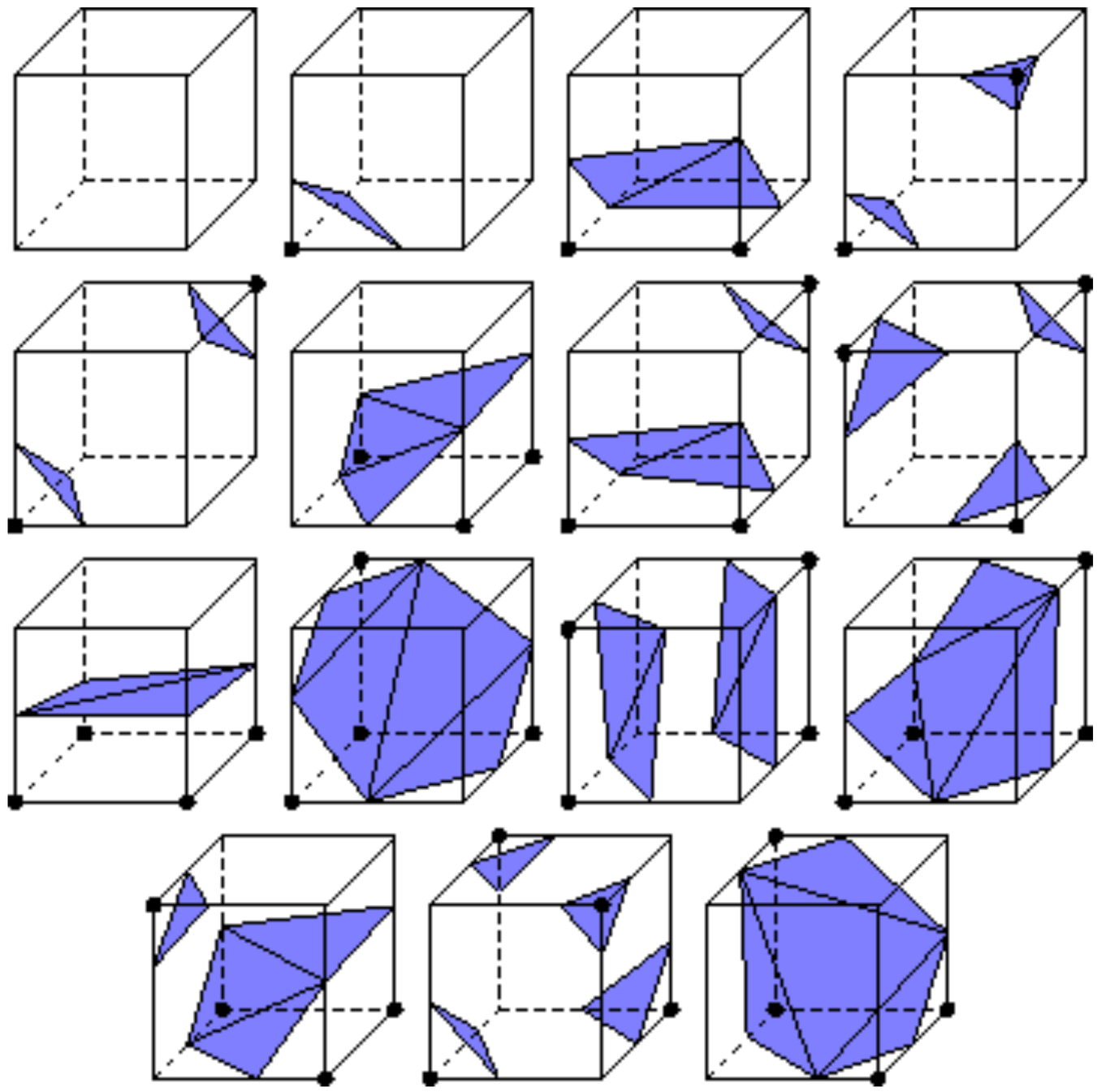
$F(s_a, t_a) <$ isovalue



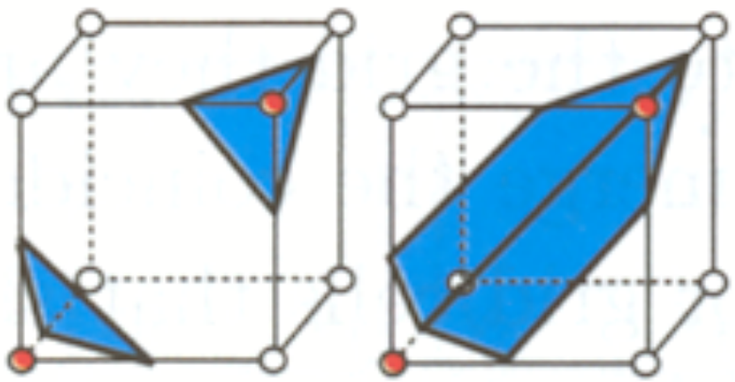
$F(s_a, t_a) \geq$ isovalue

Isosurfaces: Marching Cubes

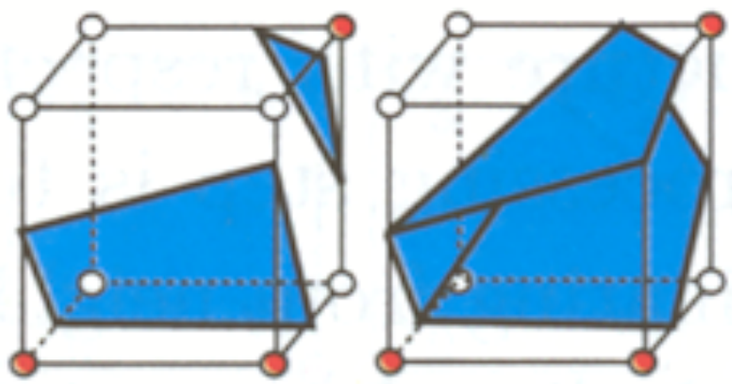
- 3D version of Marching Squares
- Isolines become isosurfaces
- Compute triangle patches instead of line segments
- Compute surface normals to apply lighting model



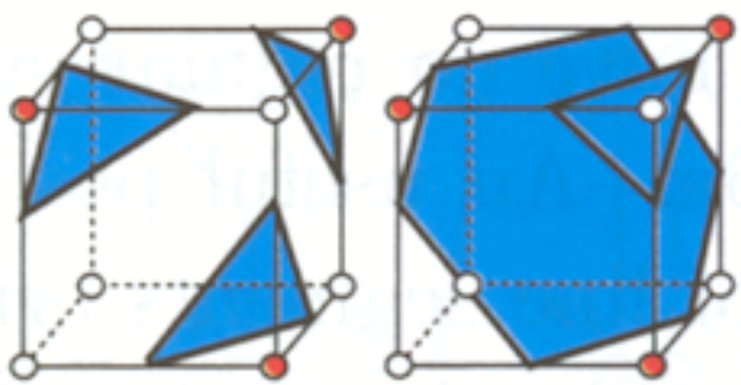
Ambiguous cases



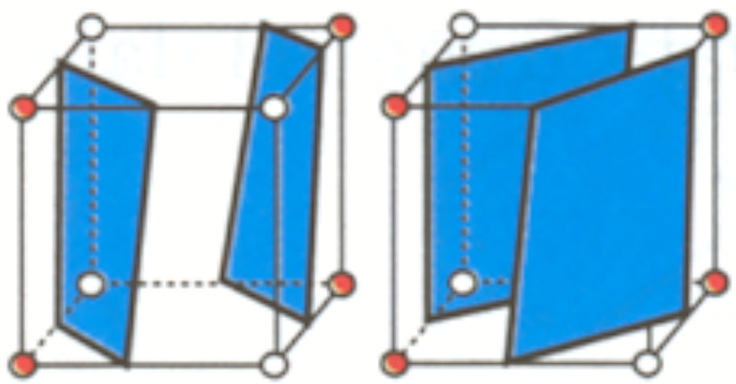
00100001



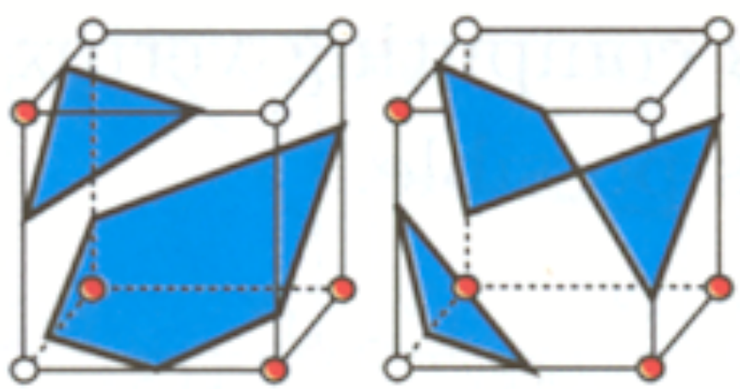
01000011



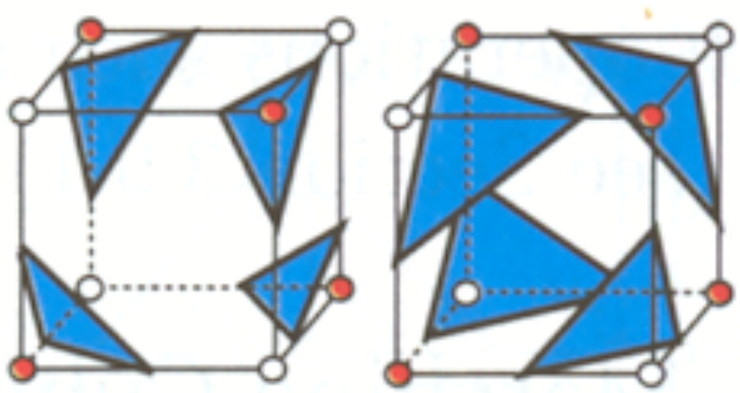
01010010



01010101



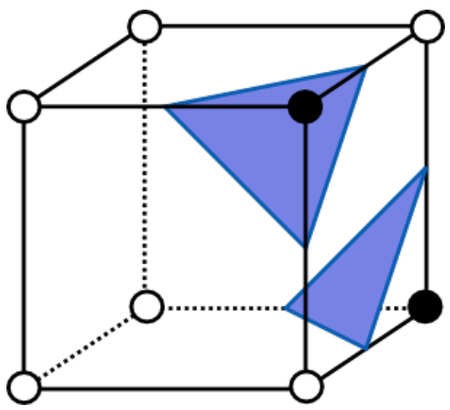
00011110

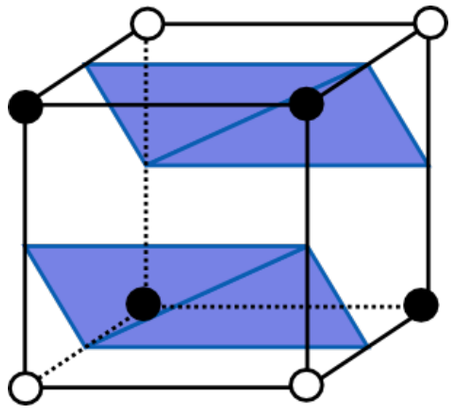


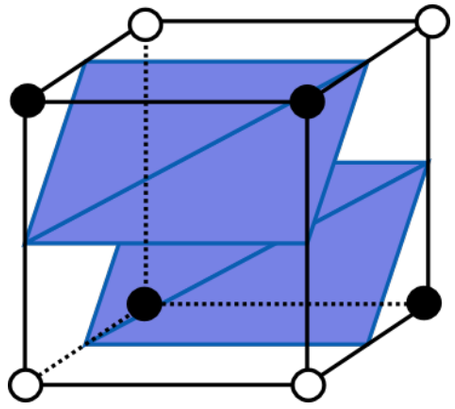
10100101

Ambiguous cases

- Cannot choose any triangulation option per cell



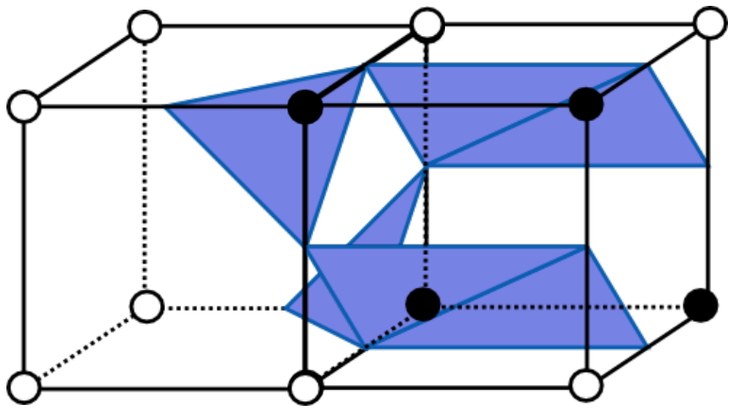


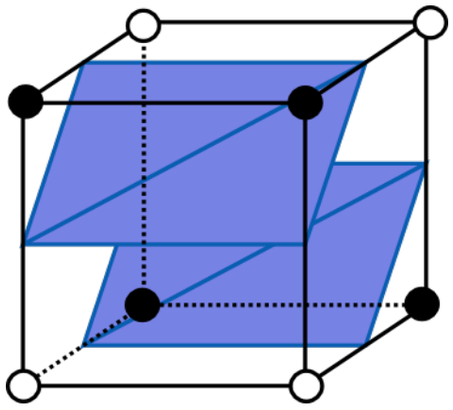


- Use asymptotic decider for each face of cube

Ambiguous cases

- Cannot choose any triangulation option per cell

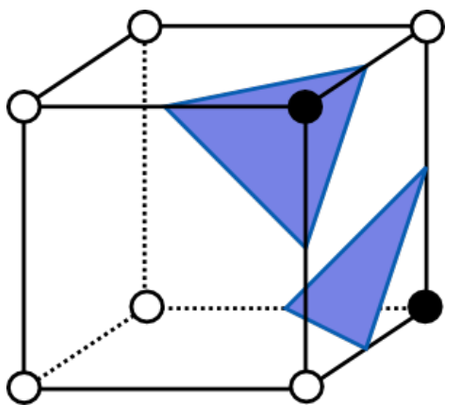


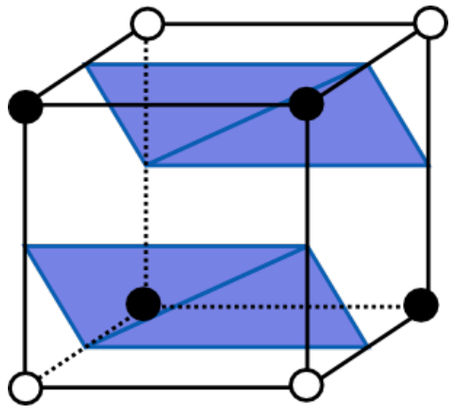


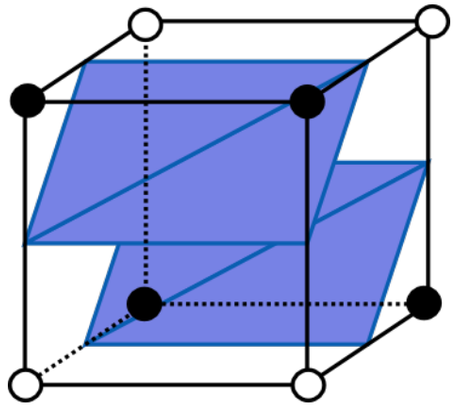
- Use asymptotic decider for each face of cube

Ambiguous cases

- Cannot choose any triangulation option per cell



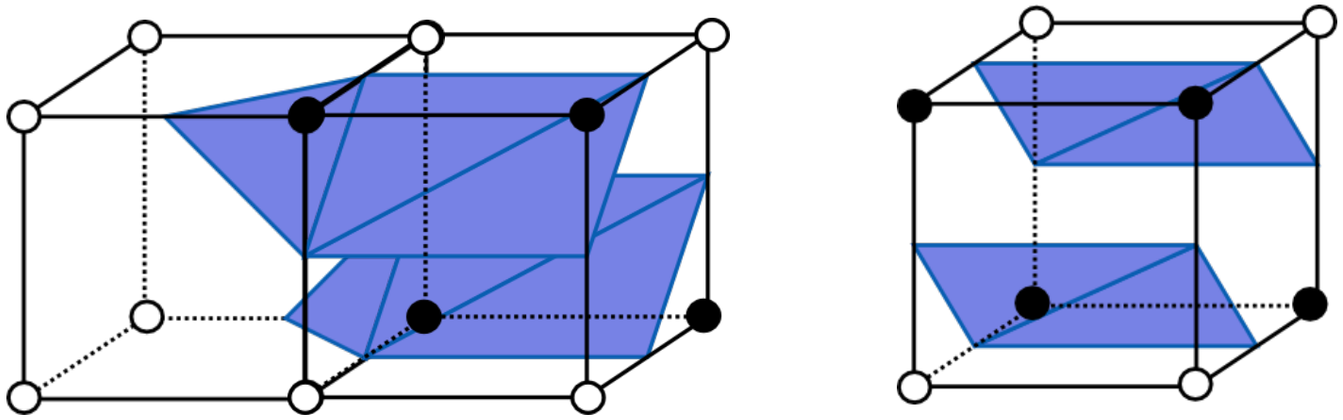




- Use asymptotic decider for each face of cube

Ambiguous cases

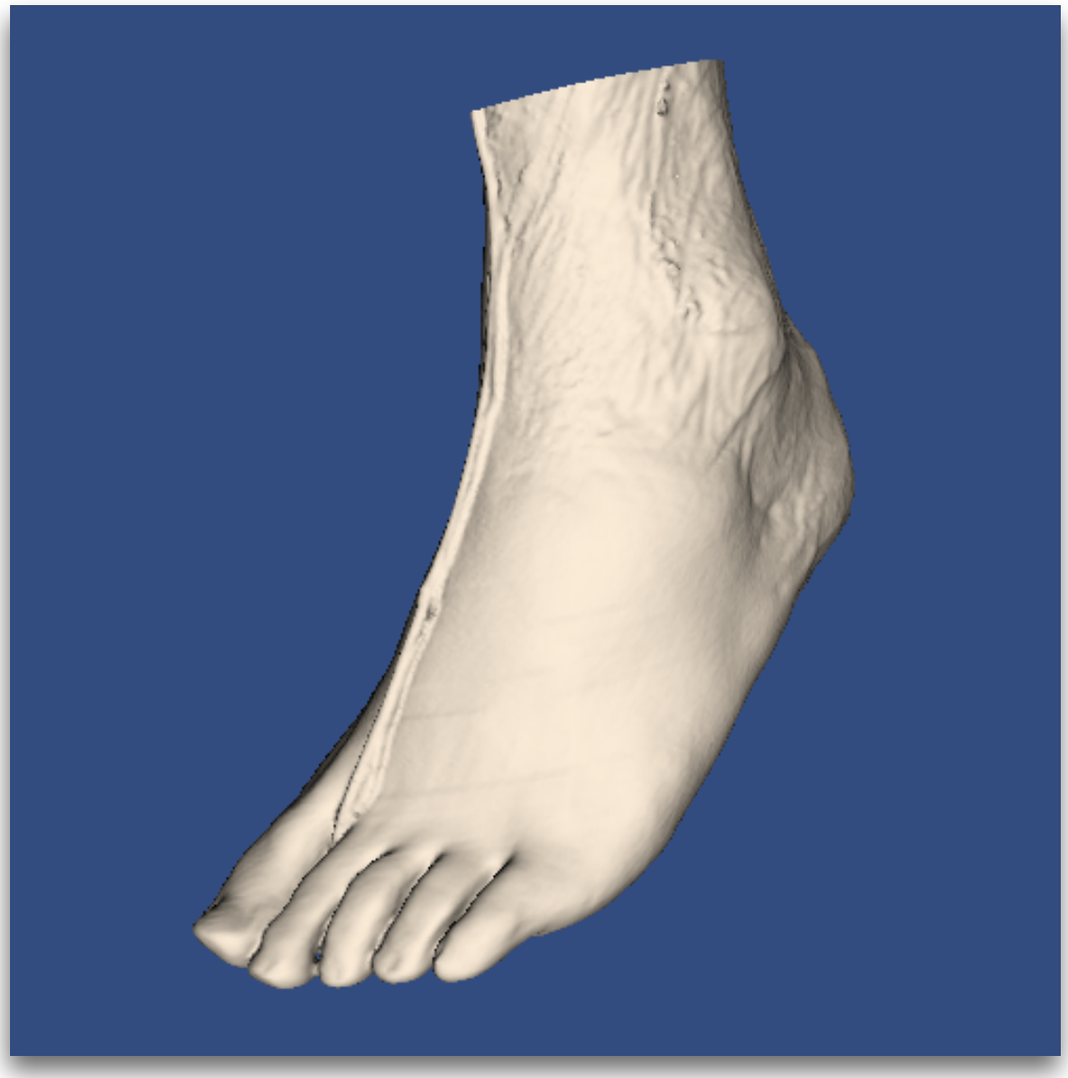
- Cannot choose any triangulation option per cell



- Use asymptotic decider for each face of cube

Example: foot CT

- Dimensions 152 x 261 x 220
- Resolution 0.95 mm x 0.95 mm x 1.0 mm
- Value range 0 - 255

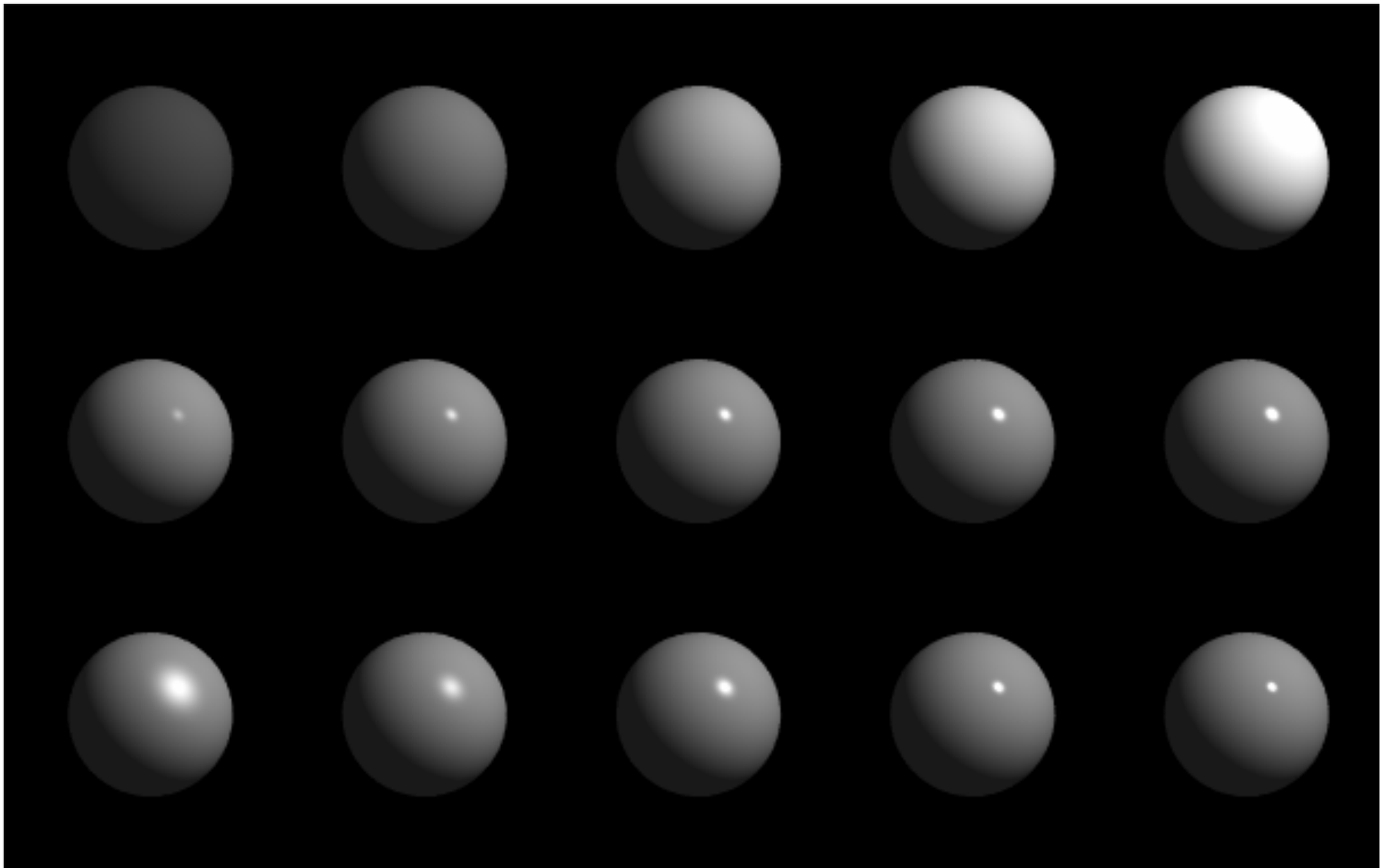


Local illumination

- Phong shading model

http://www.cs.toronto.edu/~jacobson/phong-demo/

$$I = I_a k_{\text{ambient}} + I_l k_{\text{diff}}(\mathbf{L} \cdot \mathbf{N}) + I_l k_{\text{spec}}(\mathbf{V} \cdot \mathbf{R})^{\alpha}$$



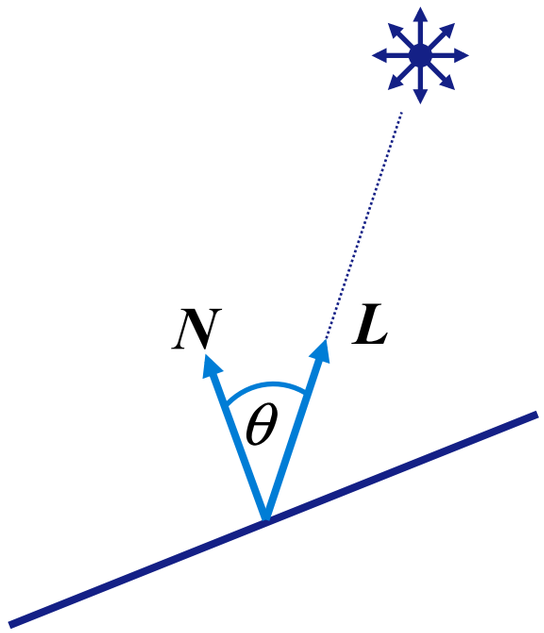
$k_{\text{spec}} = 0$
$\alpha = 0$
$0.2 \le k_{\text{diff}} \le 1$

$k_{\text{diff}} = 0.5$
$\alpha = 120$
$0.2 \le k_{\text{spec}} \le 1$

$k_{\text{diff}} = 0.5$
$k_{\text{spec}} = 0.5$
$10 \le \alpha \le 250$

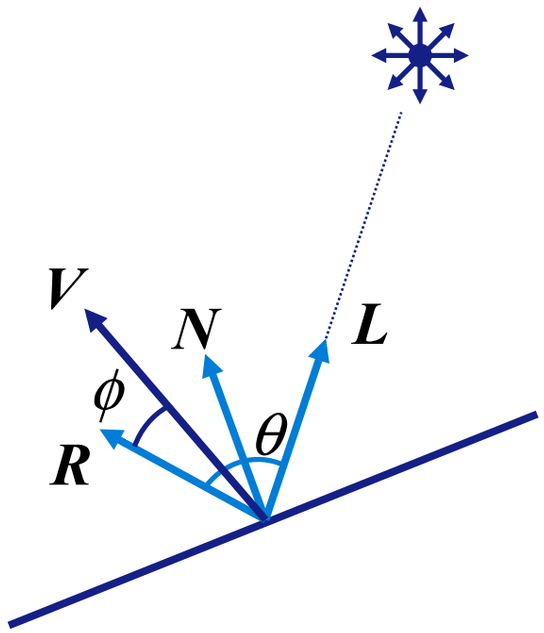
Local illumination: one-slide recap

- Diffuse reflection
- Intensity of reflected light depends on
  - angle of light rays with surface
  - reflectivity of surface
  - intensity of light



$$I_{\text{diff}} = I_l k_{\text{diff}}(\mathbf{L} \cdot \mathbf{N})$$

- Specular reflection
- Intensity of reflected light depends on
  - angle of light rays with surface
  - reflectivity of surface
  - intensity of light
  - shininess of material
  - viewer direction



$$I_{\text{spec}} = I_l k_{\text{spec}}(\mathbf{V} \cdot \mathbf{R})^{\alpha}$$

Computing surface normals

- Lighting models require information of surface orientation
- Obtain smooth shading by estimating the surface normals in the vertices
- Estimation is computed from the normalized local gradient vector

$$N_s = \frac{\nabla s}{|\nabla s|}$$

$$\nabla s \approx \frac{1}{2} \begin{pmatrix} s(i+1,j,k) - s(i-1,i,j) \\ s(i,j+1,k) - s(i,j-1,k) \\ s(i,j,k+1) - s(i,j,k-1) \end{pmatrix}$$

Derivatives and finite differences

- Approximating derivatives of discrete data
  - forward differencing

$$\Delta x = x_{i+1} - x_i$$

  - backward differencing

$$\Delta x = x_i - x_{i-1}$$

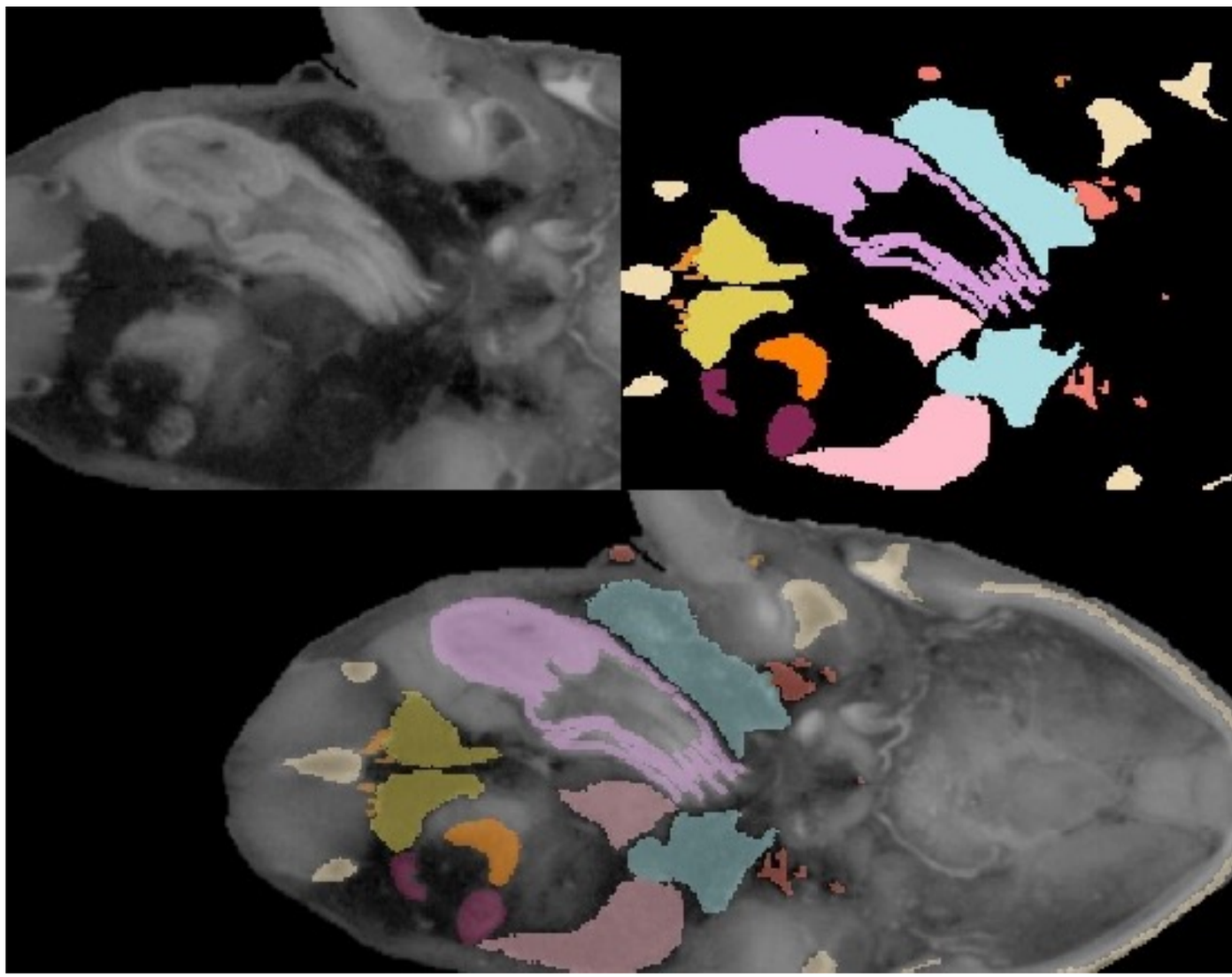
  - central differencing

$$\Delta x = \frac{1}{2}(x_{i+1} - x_{i-1})$$

- High-dimensional data
  - compute partial derivatives per dimension

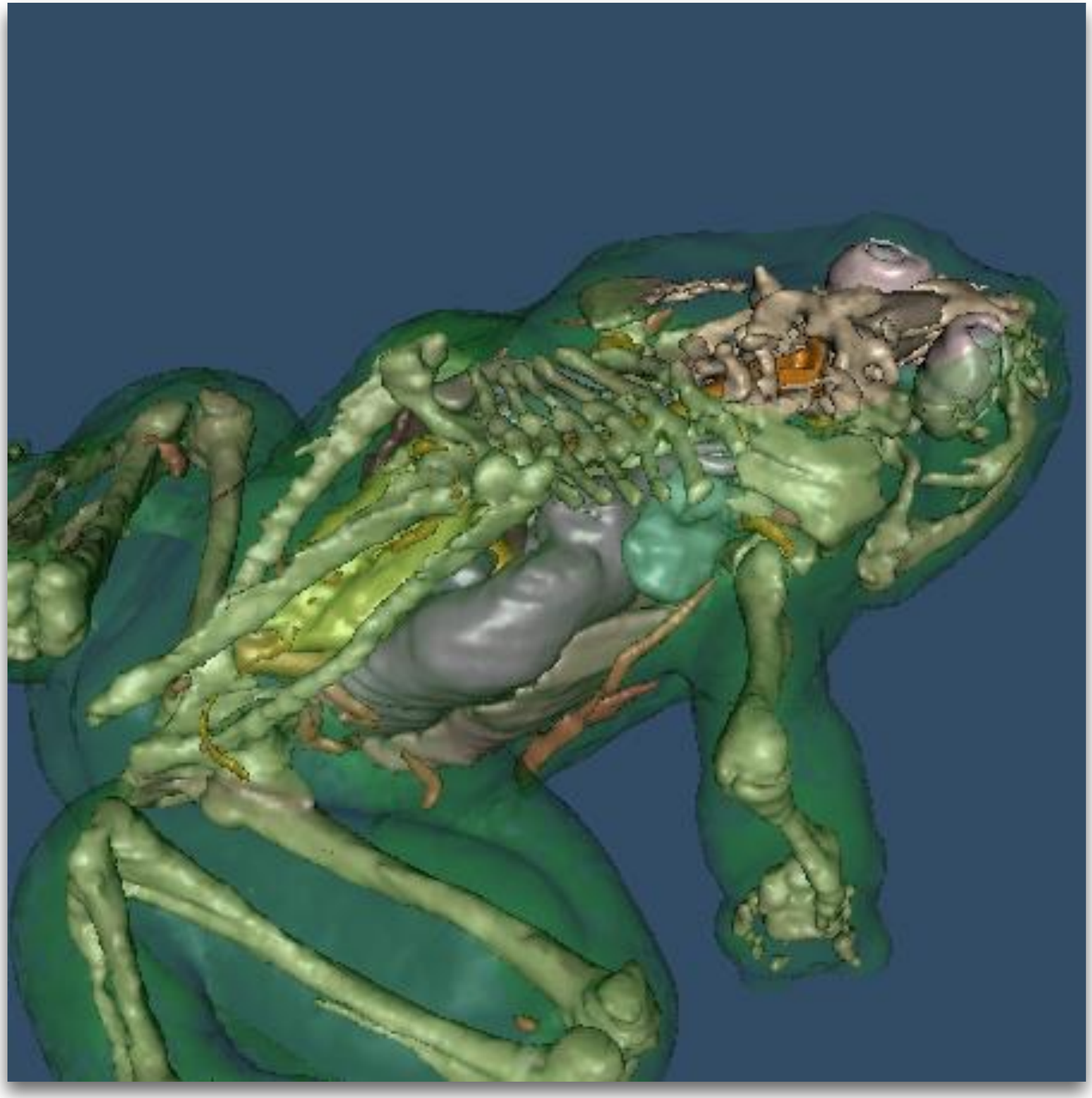
Example: virtual frog

- Data
  - MRI scan
  - physically slicing the frog and photographing the slices
- Dimensions 470 x 500 x 136
- Each pixel labeled manually with a tissue number



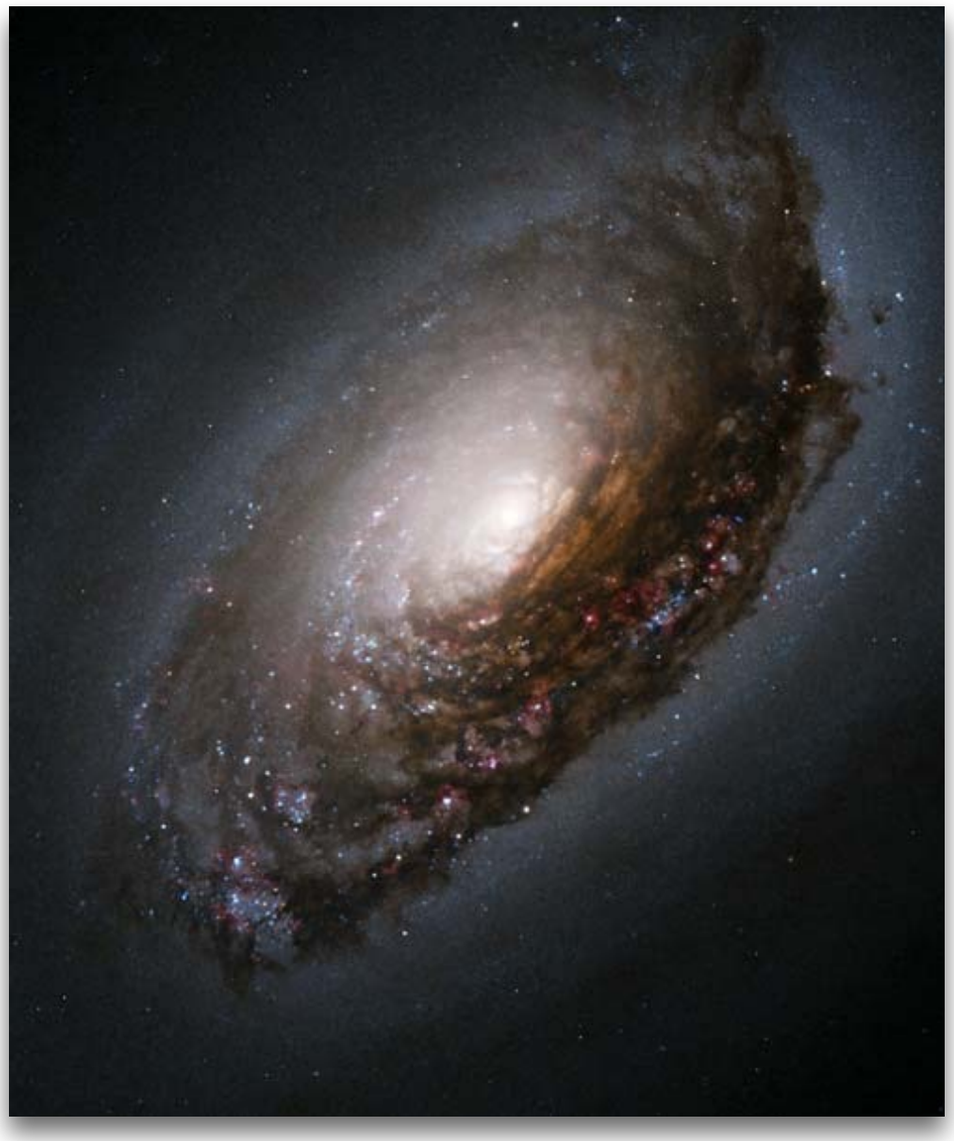
0	no tissue	8	kidney
1	blood	9	large intestine
2	brain	10	liver
3	duodenum	11	lung
4	eye retina	12	nerve
5	eye white	13	skeleton
6	heart	14	spleen
7	ileum	15	stomach

Virtual frog visualization



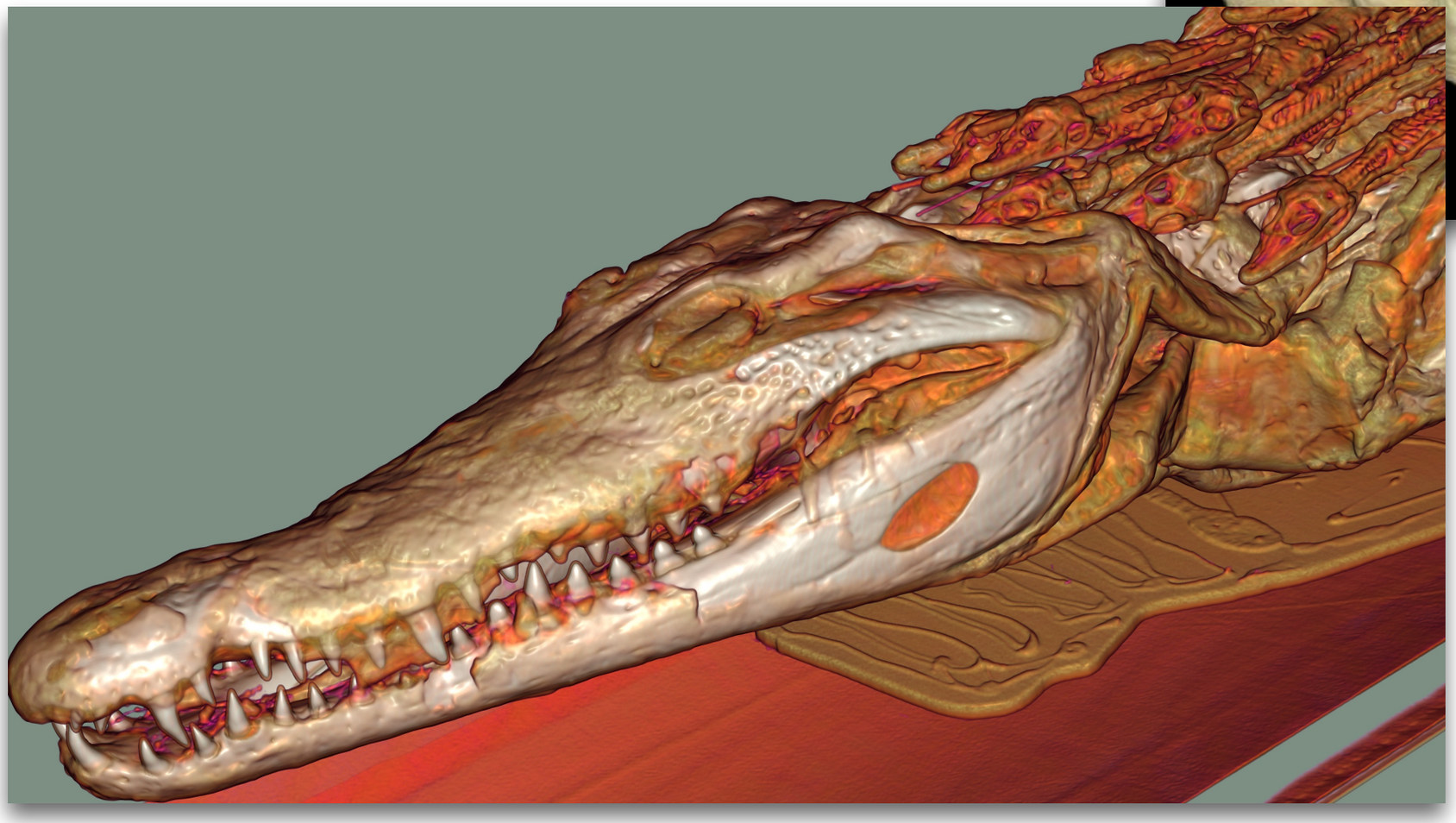
Isosurfacing drawbacks

- Provides only approximation of a surface
- Loss of information if only one or few surfaces are shown
- Amorphous phenomena have no surfaces



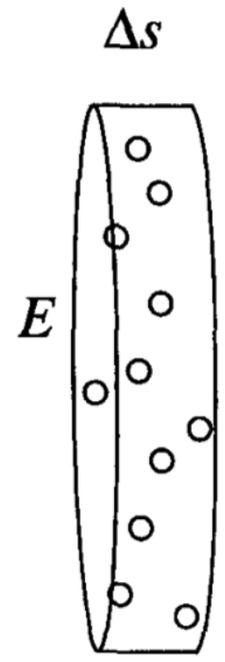
Direct volume rendering

- No derived geometry
- Scalar values
  - used directly (MIP)
  - mapped through a **transfer function** to obtain colors and opacities (DVR)



Compositing and the optical model

- Optical model
  - continuous medium with very small particles
  - particles **absorb** light, and **emit** light themselves
- Interaction of light with medium described by



$$\frac{dI}{dt} = c(t) - \tau(t)I(t)$$

$$I(t) = I_0 e^{-\int_0^t \tau(u)du} + \int_0^t c(s)e^{-\int_s^t \tau(u)du} ds$$

Max, **Optical models for direct volume rendering**. IEEE Trans. Visualization and Computer graphics 1(2):99-108,1995.

Volume rendering equation

- Discretize and simplify

$$I(t) = I_0 e^{-\int_0^t \tau(u)du} + \int_0^t c(s) e^{-\int_s^t \tau(u)du} ds$$

$$I(p) = \sum_{i=0}^{n-1} c_i \prod_{j=i+1}^{n-1} (1 - \tau_j)$$

color **opacity**

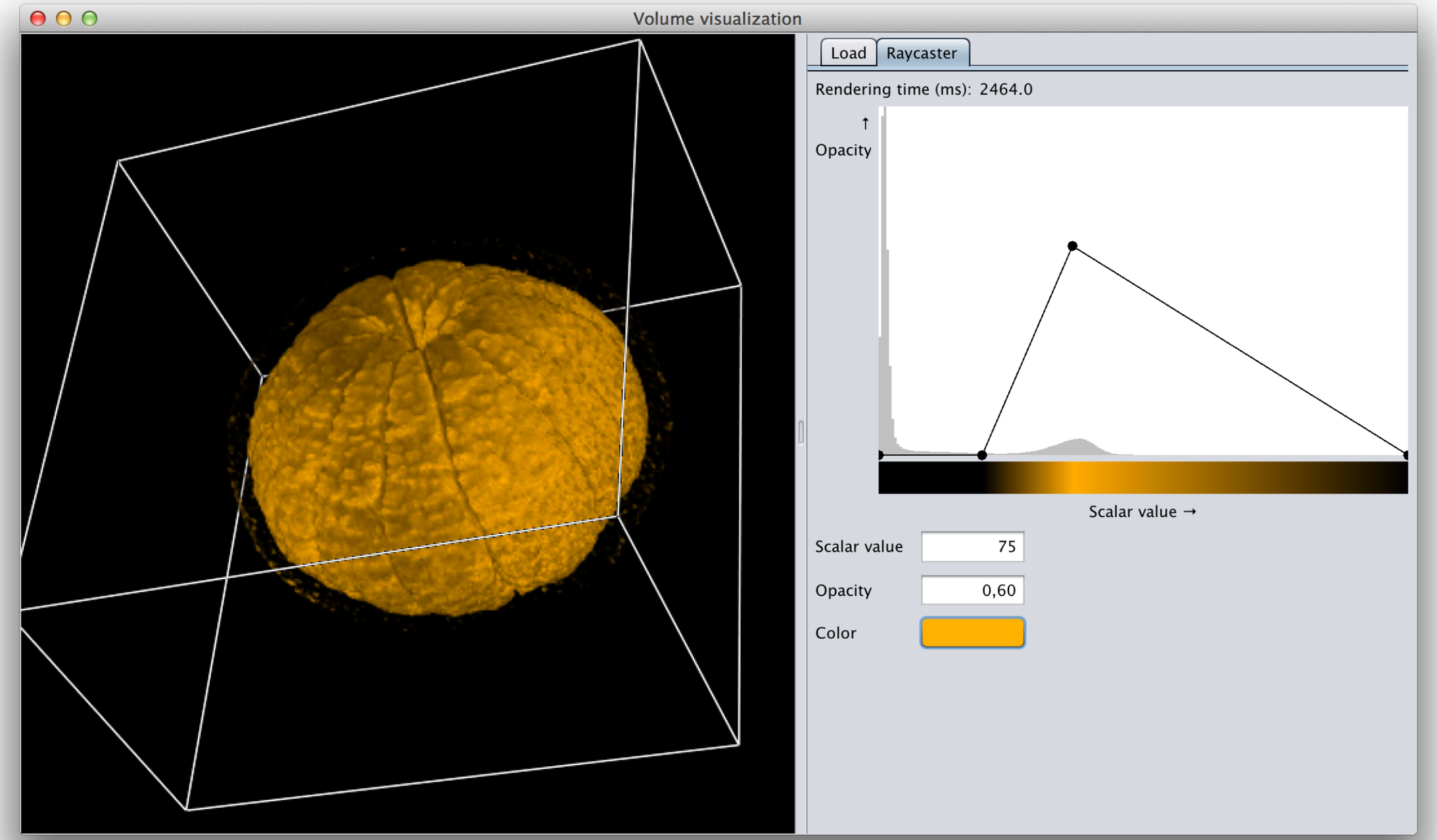
- Compositing order (implementation aspect of above equation)
  - back-to-front

$$C_i = \tau_i c_i + (1 - \tau_i) C_{i-1}$$

  - front-to-back

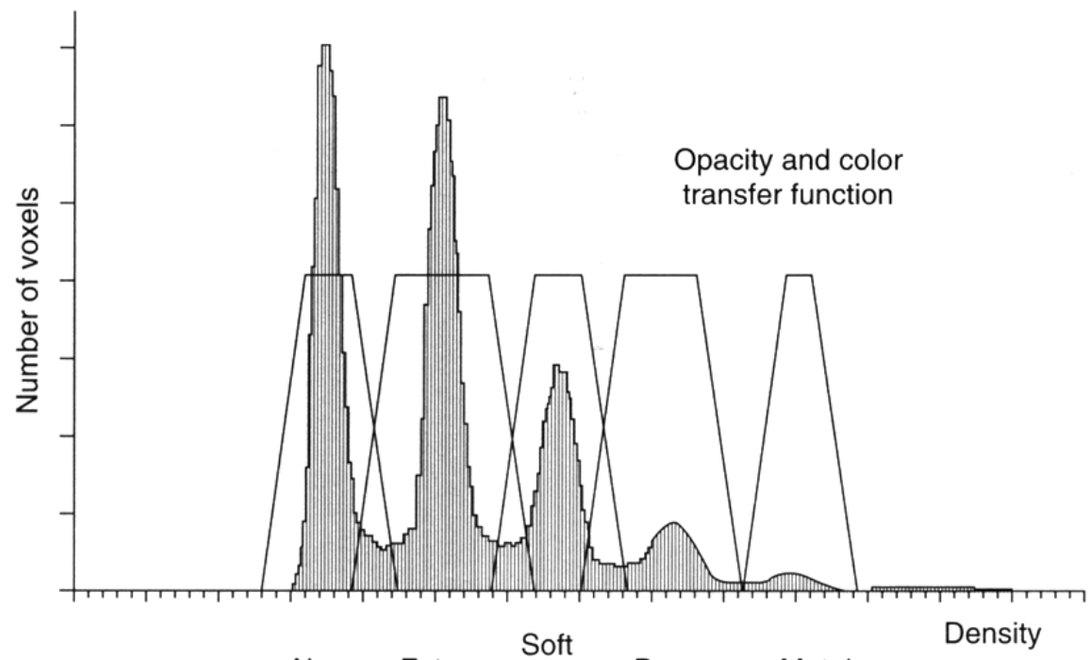
Transfer functions

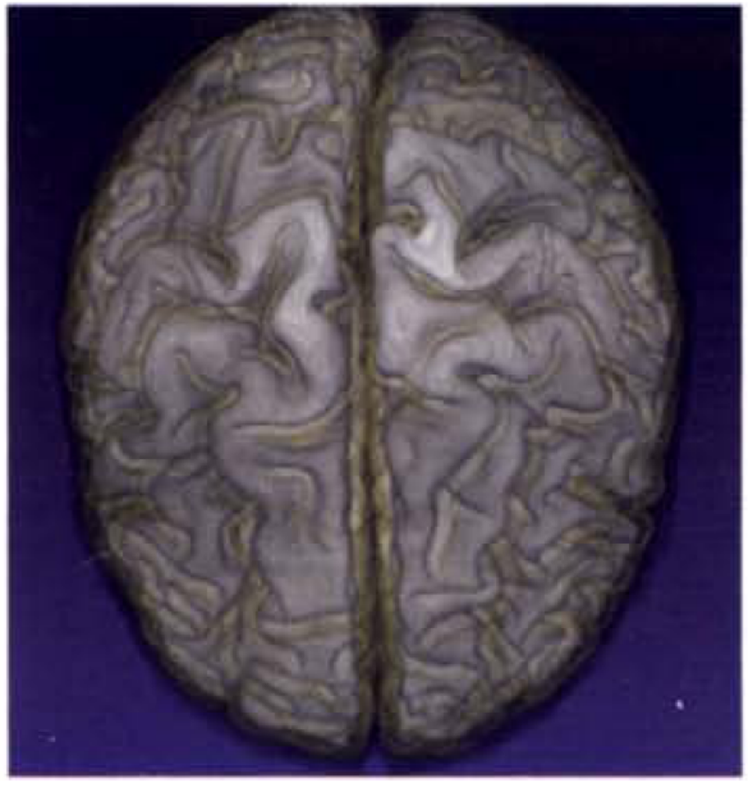
- Map data values to color and opacity



Transfer function design strategy

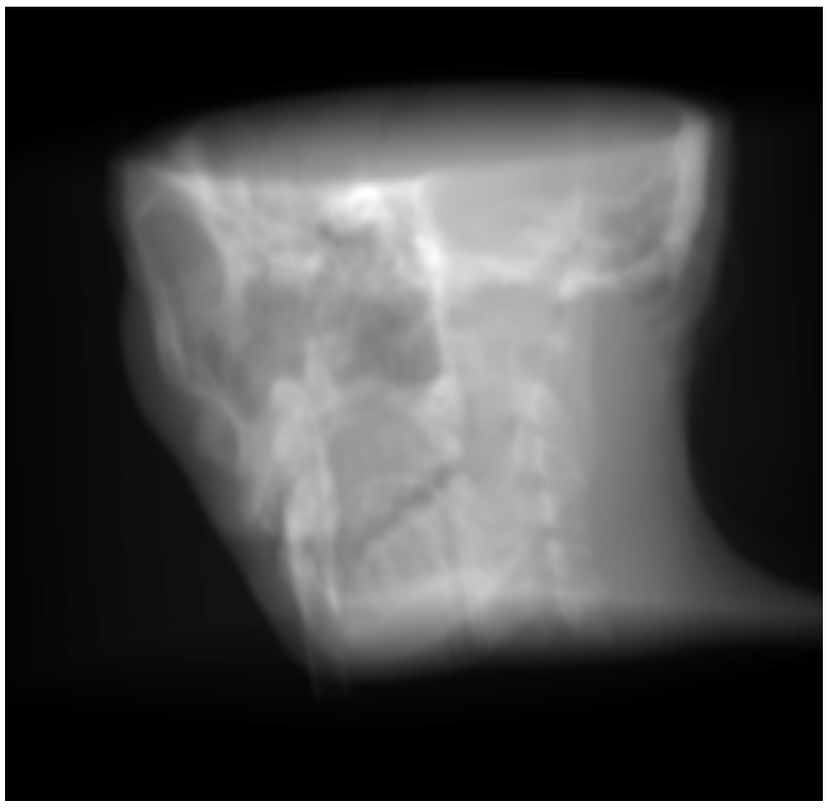
- Trial-and-error: repeat until satisfied
- Make use of density histogram
- Try to emphasize boundaries





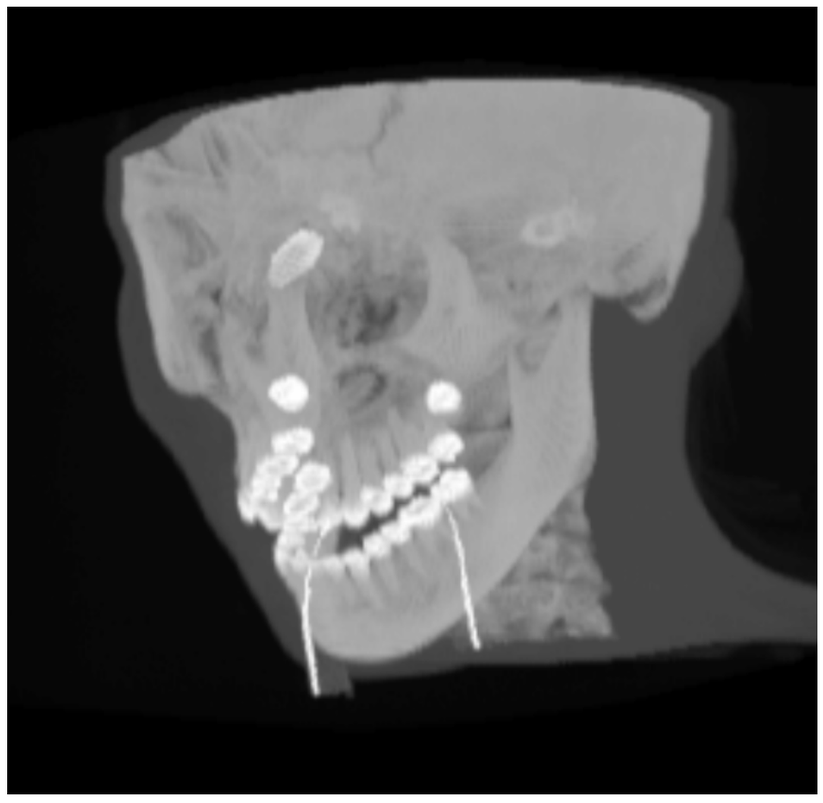
Ray functions summary

X-ray



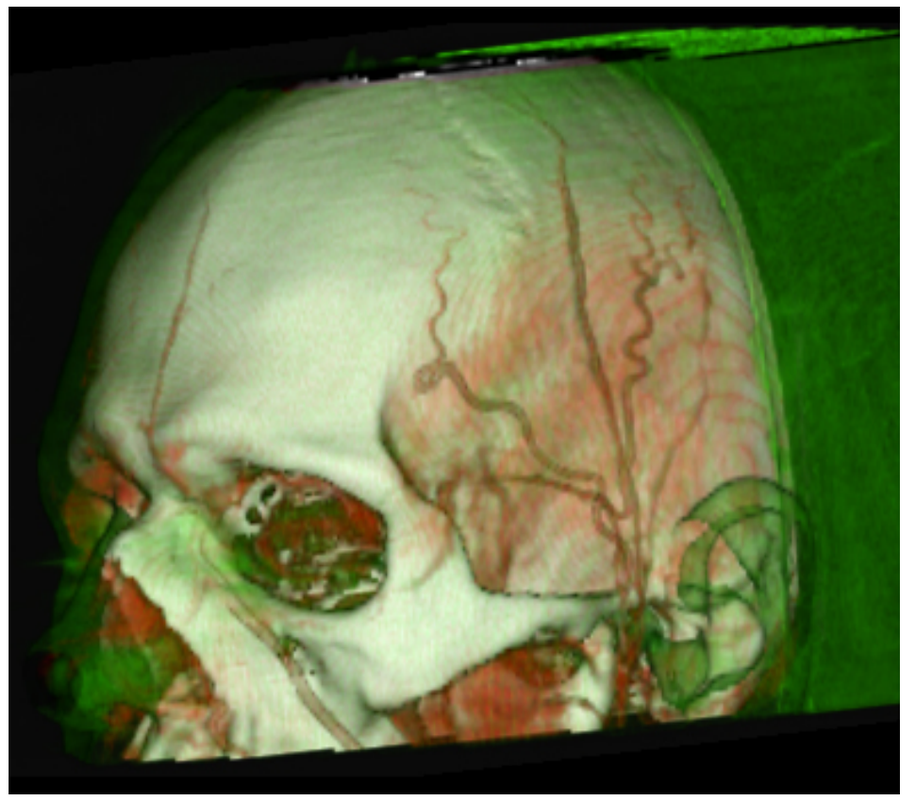
$$I(p) = \sum_t s_t$$

MIP



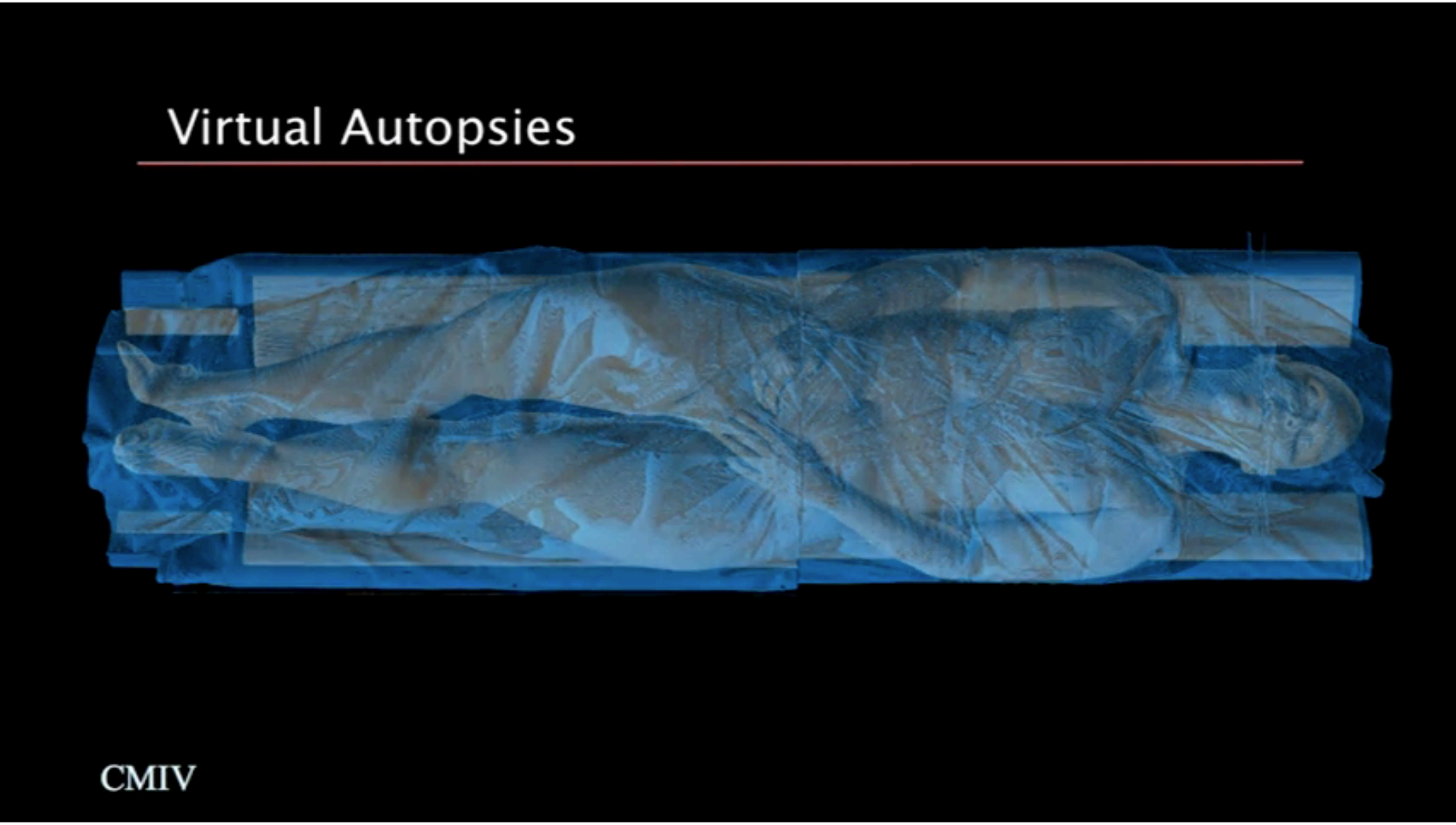
$$I(p) = \max_t s_t$$

Composite



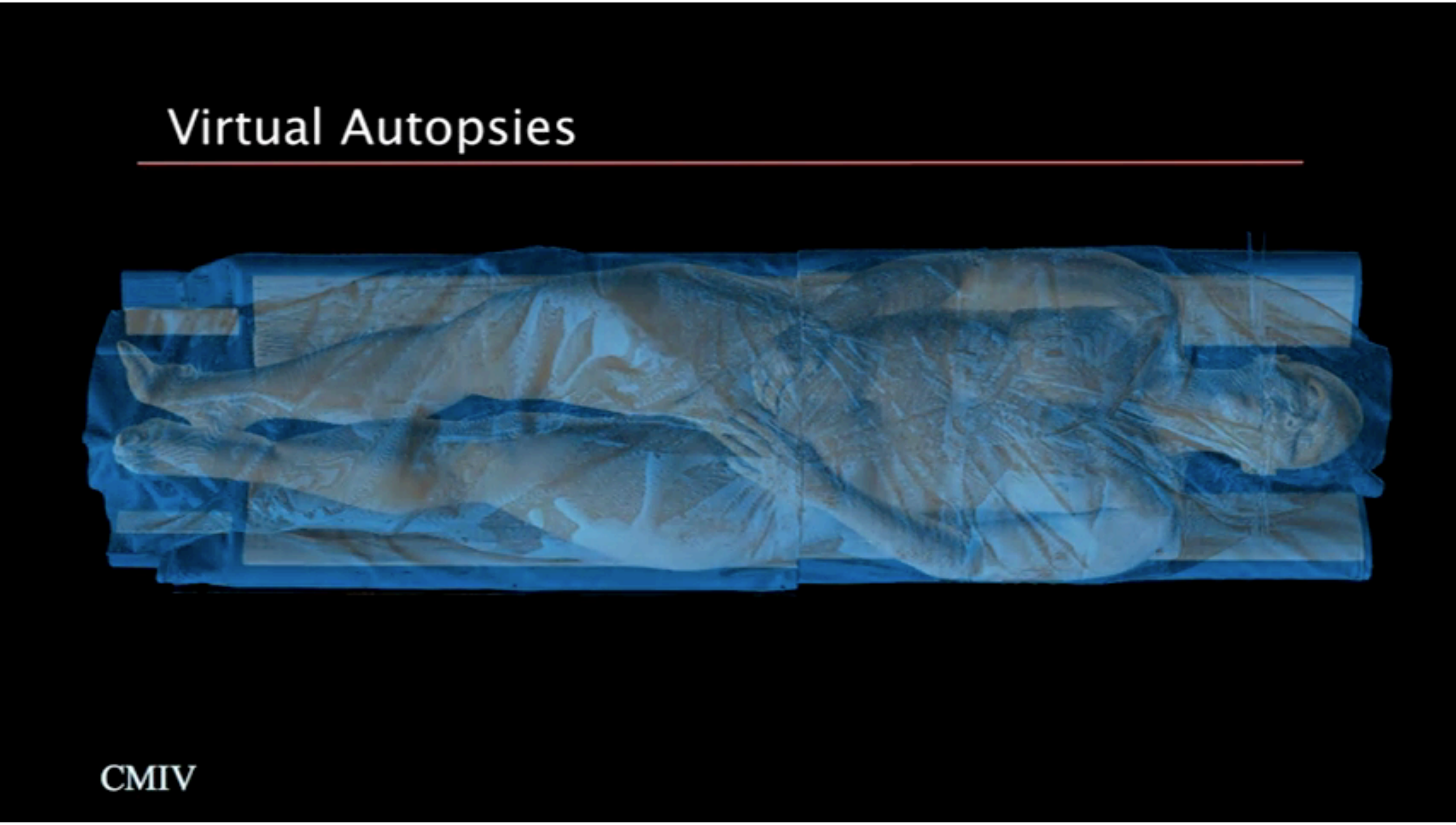
$$I(p) = \sum_{i=0}^{n-1} c_i \prod_{j=i+1}^{n-1} (1 - \tau_j)$$

Virtual autopsy



Anders Ynnerman: Visualizing the medical data explosion. Part of TEDx talk 2010

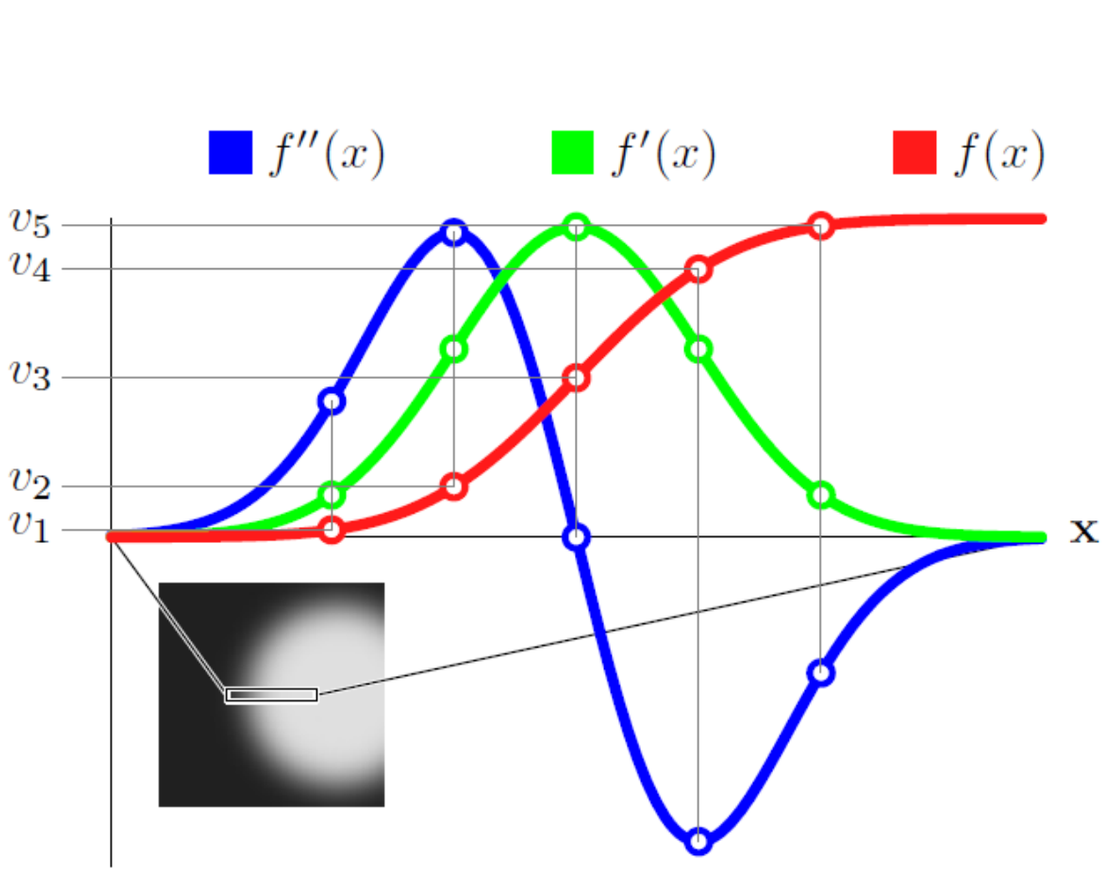
Virtual autopsy



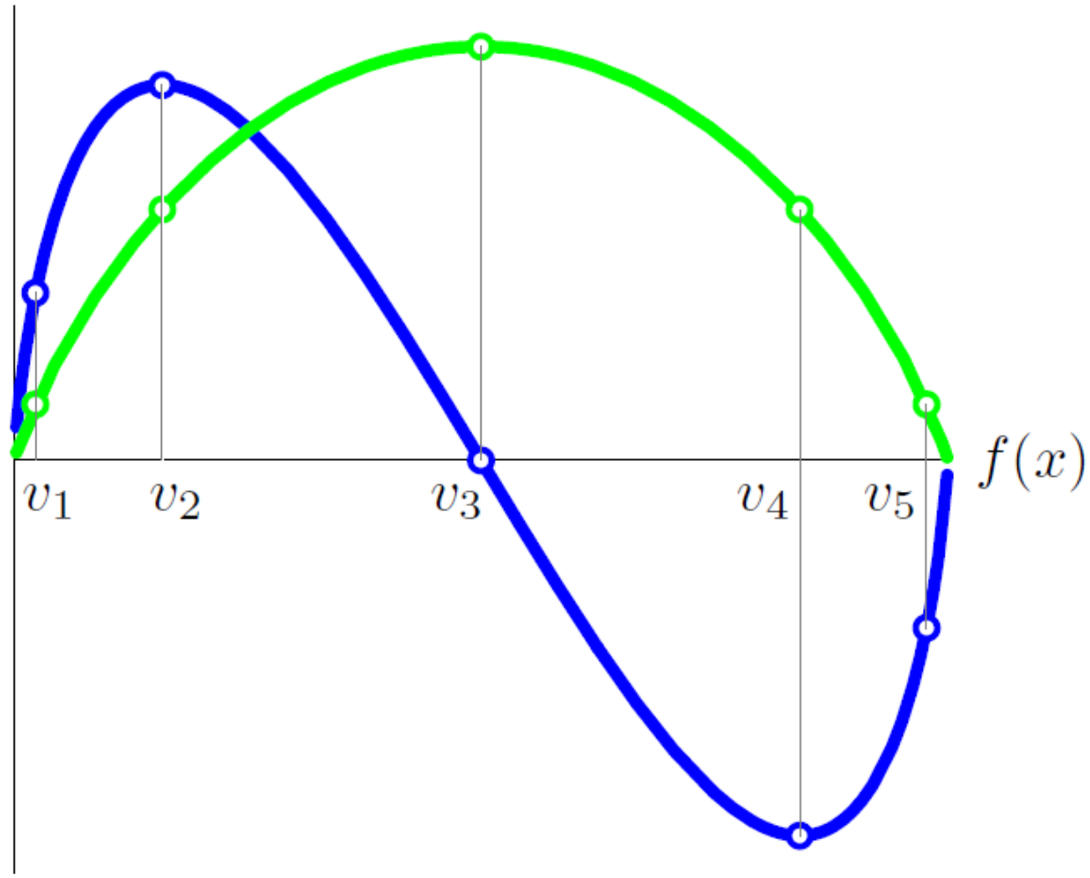
Anders Ynnerman: Visualizing the medical data explosion. Part of TEDx talk 2010

Data-driven transfer function design

- Boundary exists where
  - maximum in first-order derivative
  - zero crossing in second-order derivative

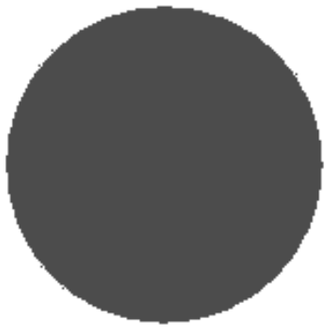


function of position



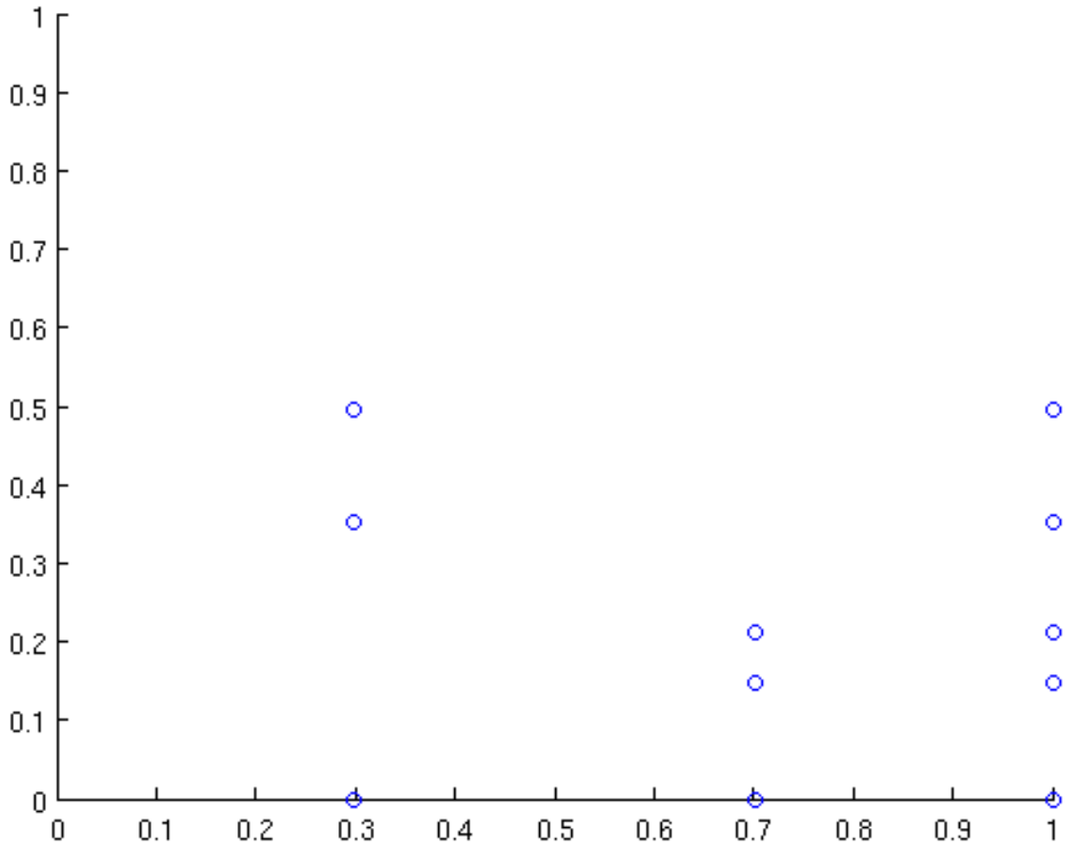
function of value

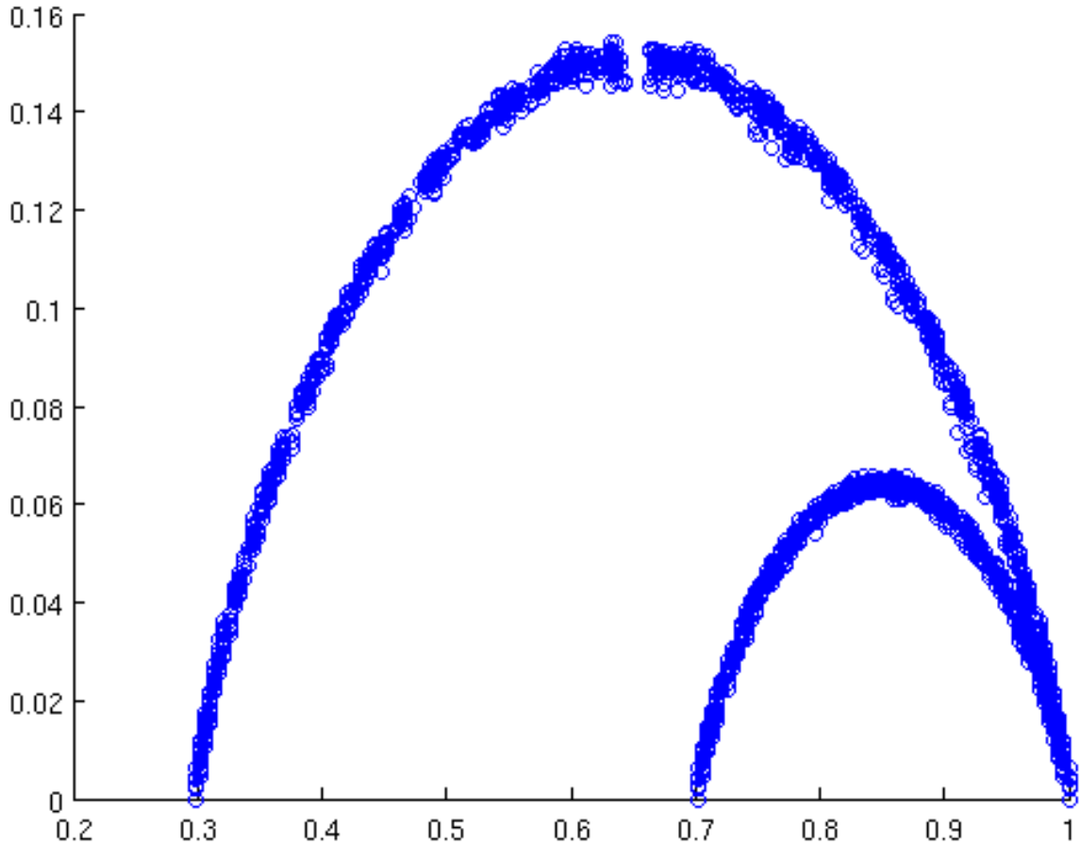
Intensity/gradient magnitude plot



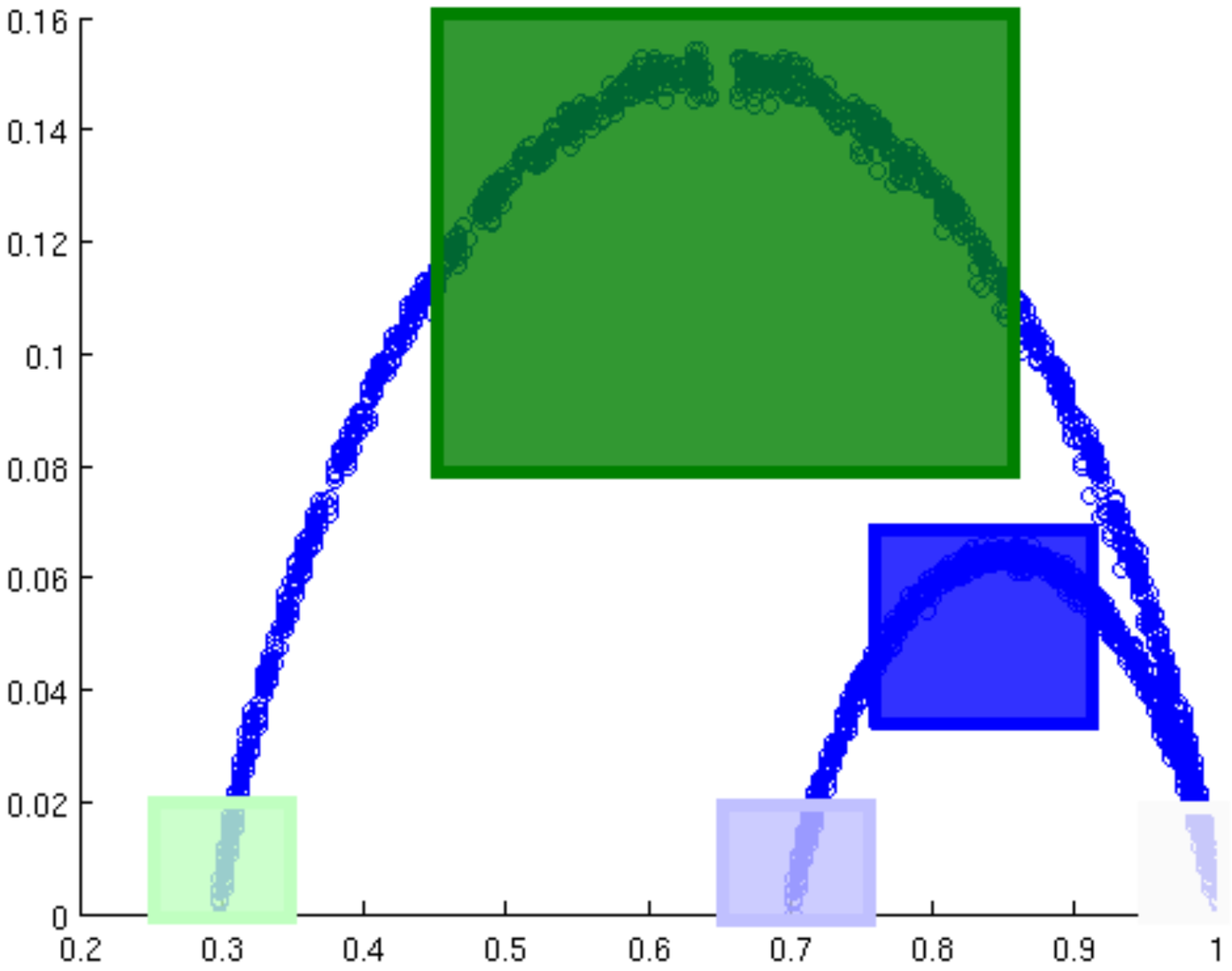
step edges

blurred step edges

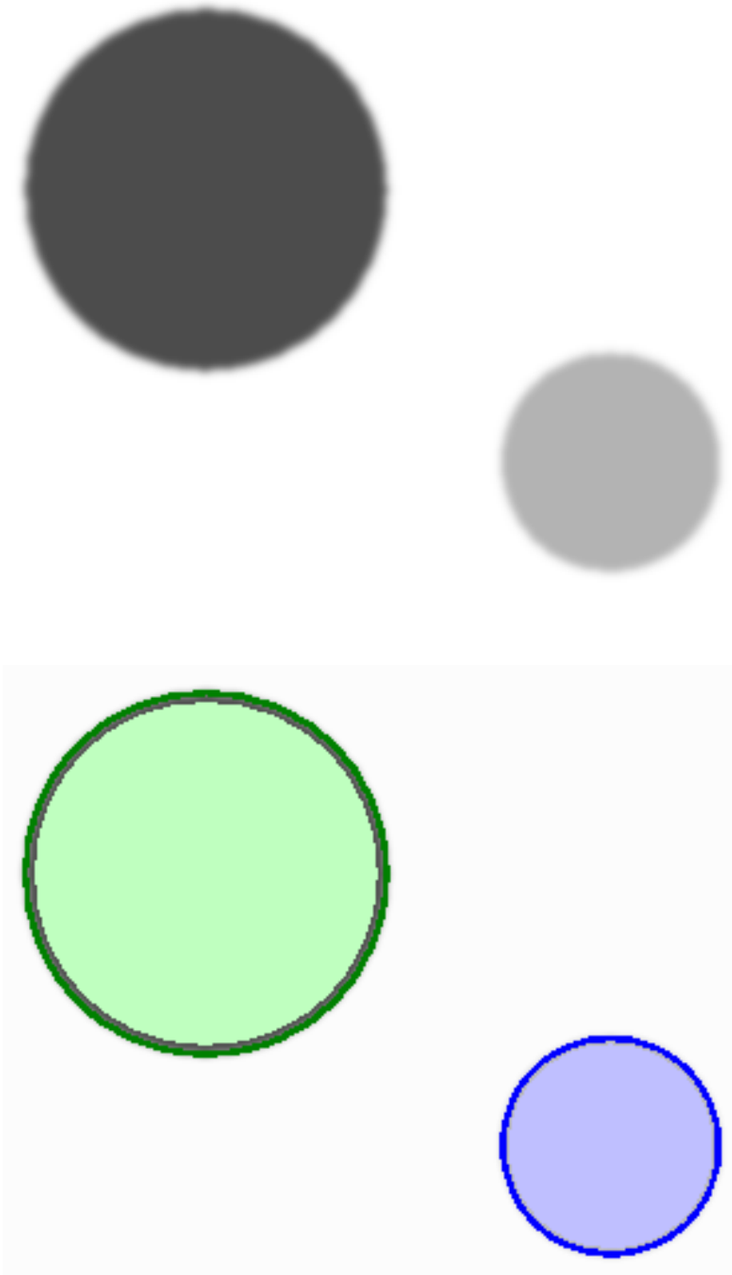




Transfer function



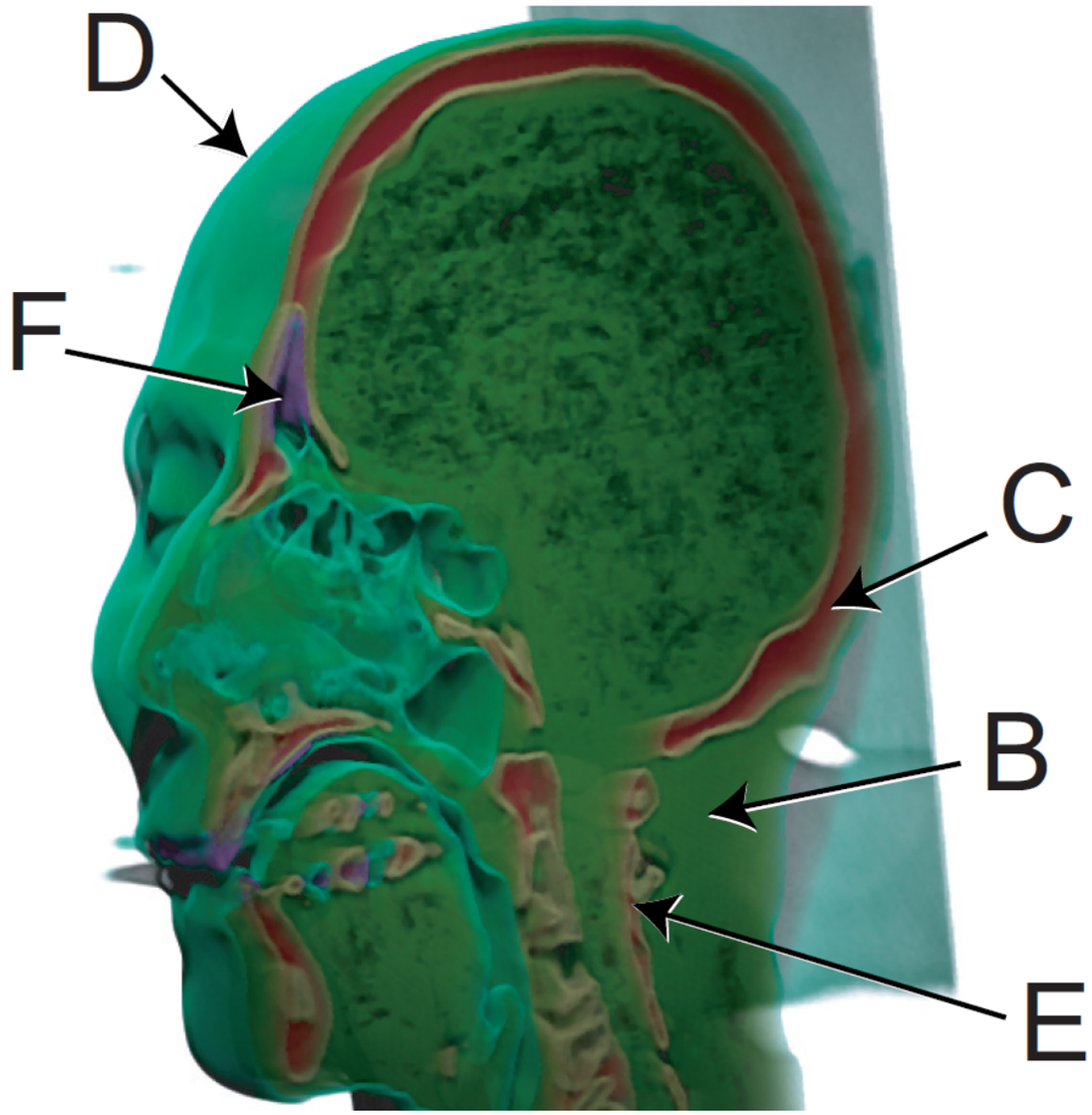
TF design

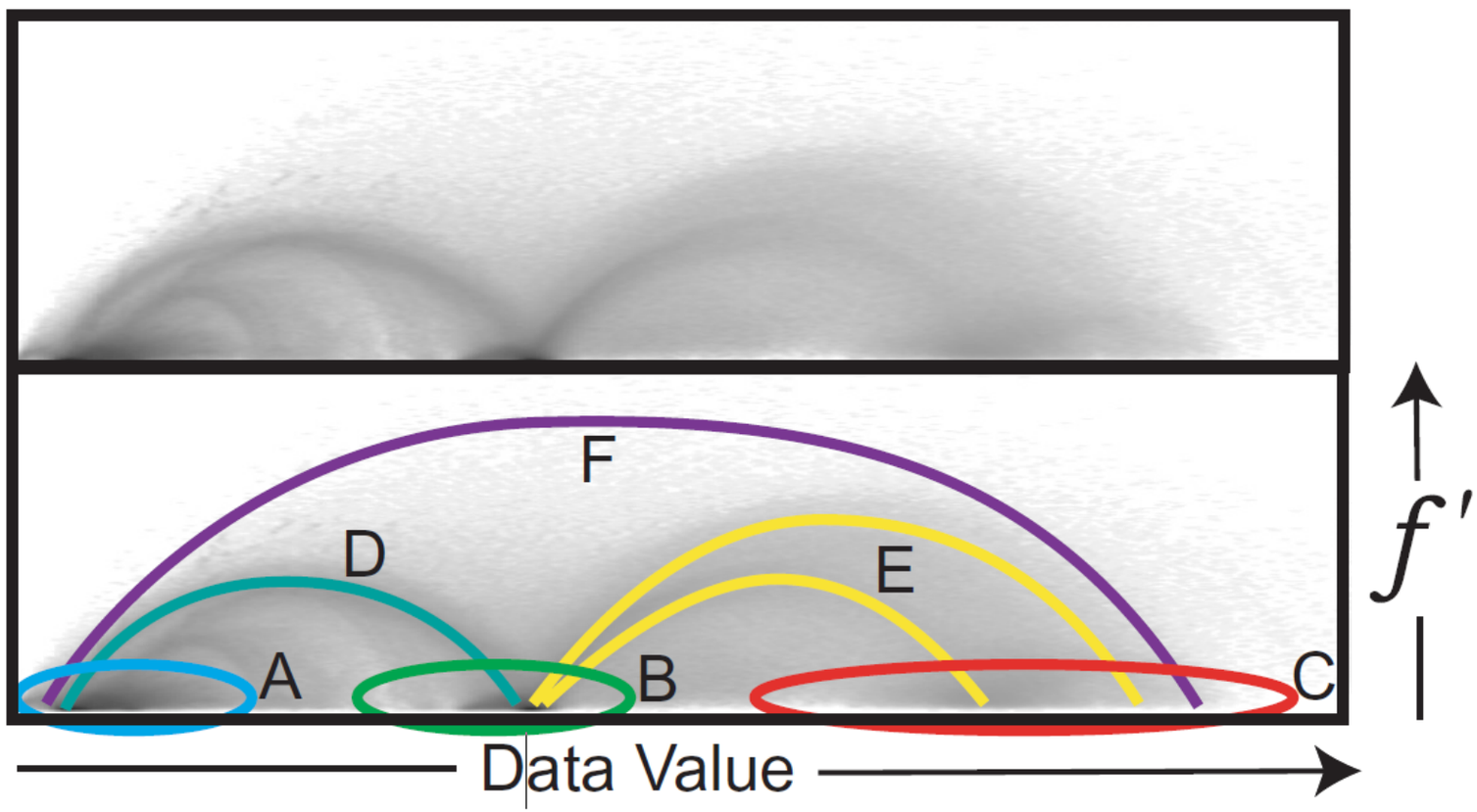


TF applied

2-D transfer functions

- Histogram of voxel intensity and gradient magnitude
- Identify arcs





Local illumination in volume rendering

- Use standard Phong model $I = I_a k_{\text{ambient}} + I_l k_{\text{diff}}(\mathbf{L} \cdot \mathbf{N}) + I_l k_{\text{spec}}(\mathbf{V} \cdot \mathbf{R})^{\alpha}$
- Estimate normal vector **N** by computing gradient in each voxel

Reading this week

Volume Rendering

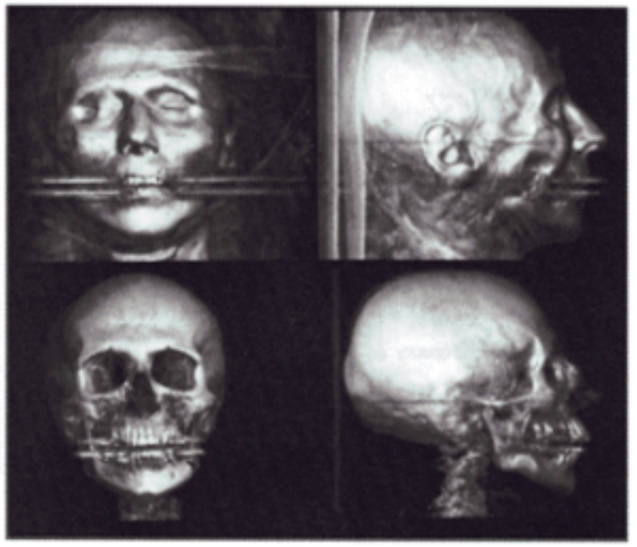
Display of Surfaces from Volume Data

Marc Levoy, University of North Carolina

In this article we will explore the application of *volume rendering* techniques to the display of surfaces from sampled scalar functions of three spatial dimensions. It is not necessary to fit geometric primitives to the sampled data. Images are formed by directly shading each sample and projecting it onto the picture plane.

Surface-shading calculations are performed at every voxel with local gradient vectors serving as surface normals. In a separate step, surface classification operators are applied to compute a partial opacity for every voxel. We will look at operators that detect isovalue contour surfaces and region boundary surfaces. Independence of shading and classification calculations ensure an undistorted visualization of 3D shape. Nonbinary classification operators ensure that small or poorly defined features are not lost. The resulting colors and opacities are composited from back to front along viewing rays to form an image.

The technique is simple and fast, yet displays surfaces exhibiting smooth silhouettes and few other aliasing artifacts. We will also describe the use of selective blurring and supersampling to further improve image quality. Examples from two applications are given: molecular graphics and medical imaging.



Visualization of scientific computations is a rapidly growing application of computer graphics. A large subset of these applications involves sampled functions of

surfaces from volume data consist of applying a surface detector to the sample array, fitting geometric primitives to the detected surfaces, then rendering these primitives using conventional surface-rendering algorithms. The techniques differ from one another mainly in the choice of primitives and the scale at which they are defined.

In the medical imaging field, a common approach is to apply thresholding to the volume data. The resulting binary representation can be rendered by treating