Max-tree visualization

ASCI course Advanced Morphological Filters

Michel Westenberg

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

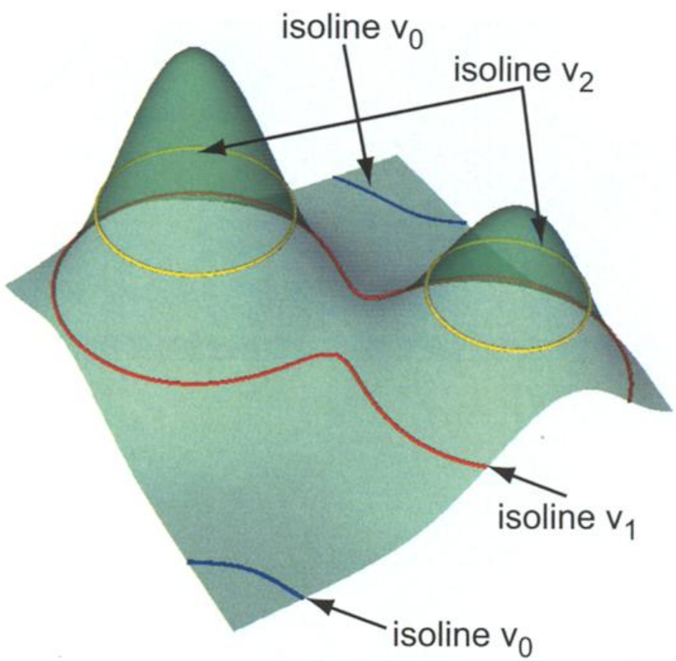
Volume data

- Samples taken at regularly spaced intervals along three orthogonal axes
- Isotropic: same constant spacing for all axes (uniform grid)
- Anisotropic: different constant spacing for each axis (rectilinear grid)
- Samples are called voxels

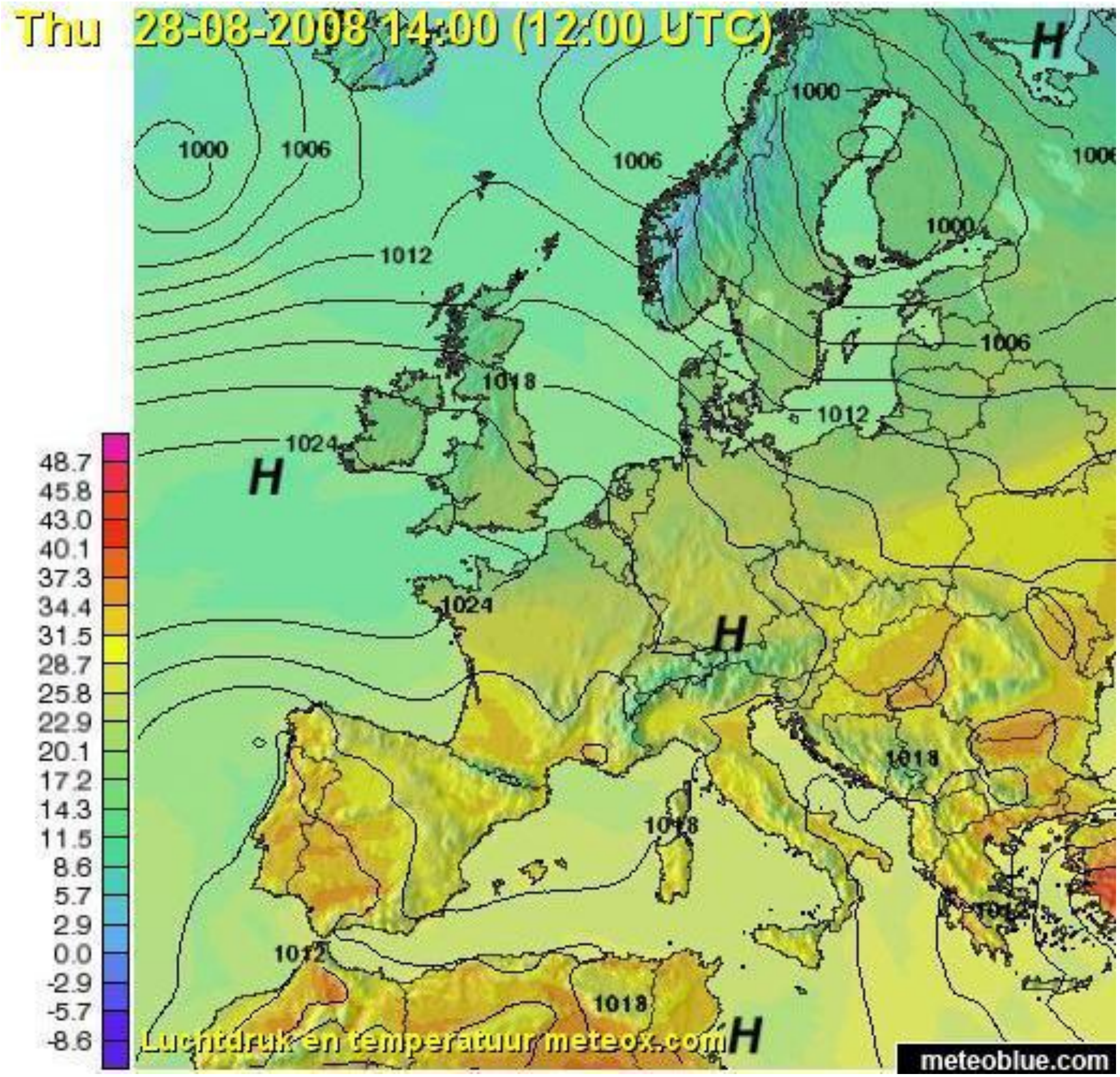
Volume visualization

- Surface fitting algorithms
  - marching cubes
- Direct volume rendering
  - ray casting
  - voxel projection
  - texture-based rendering

Contour lines

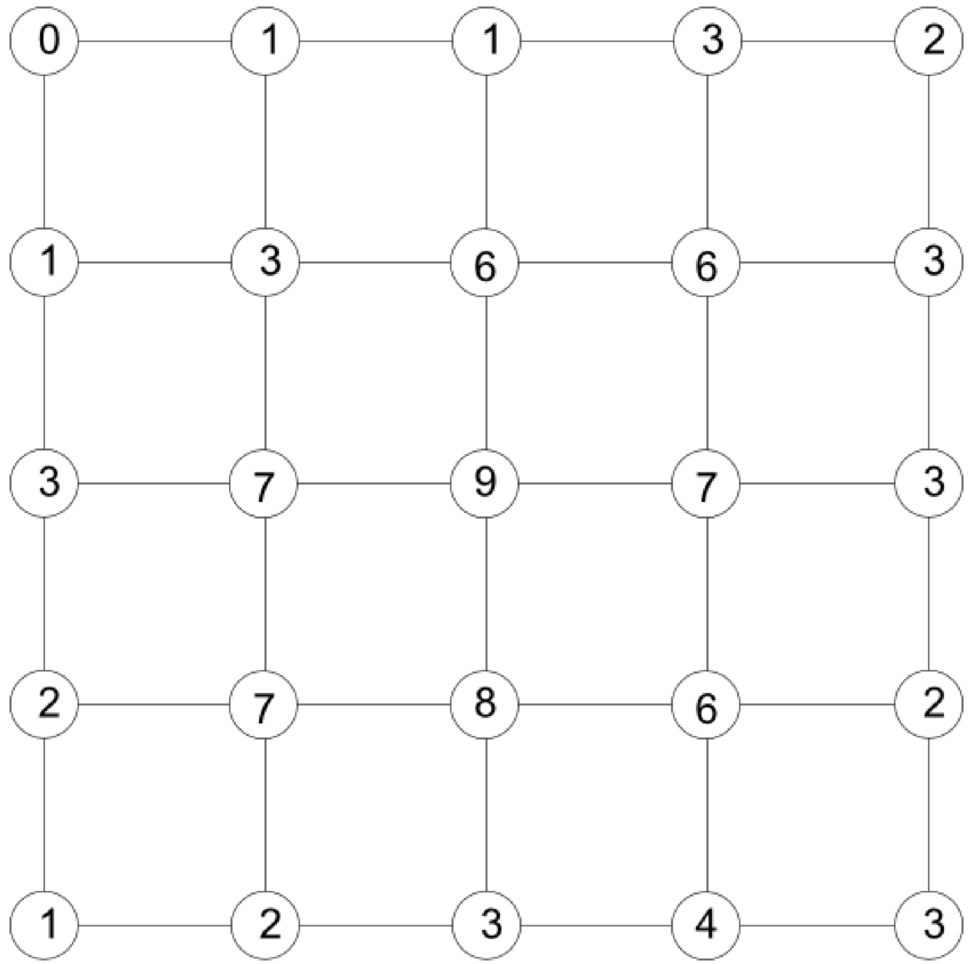


$$f(x, y) = c$$



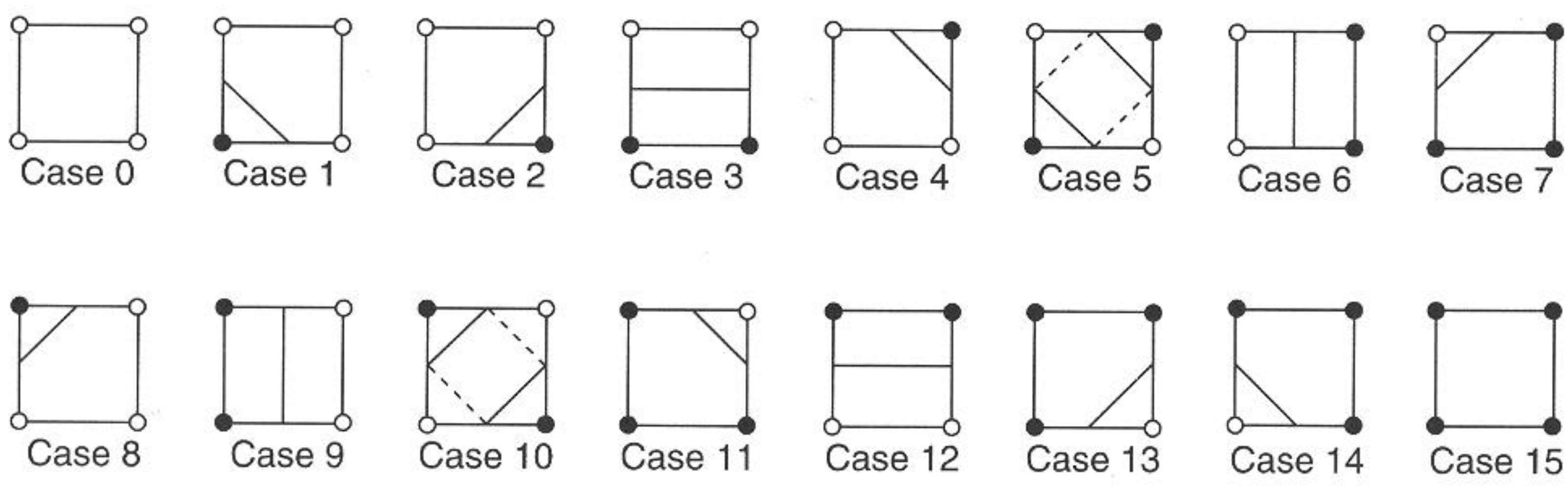
Isolines

Draw $f(x, y) = 5$



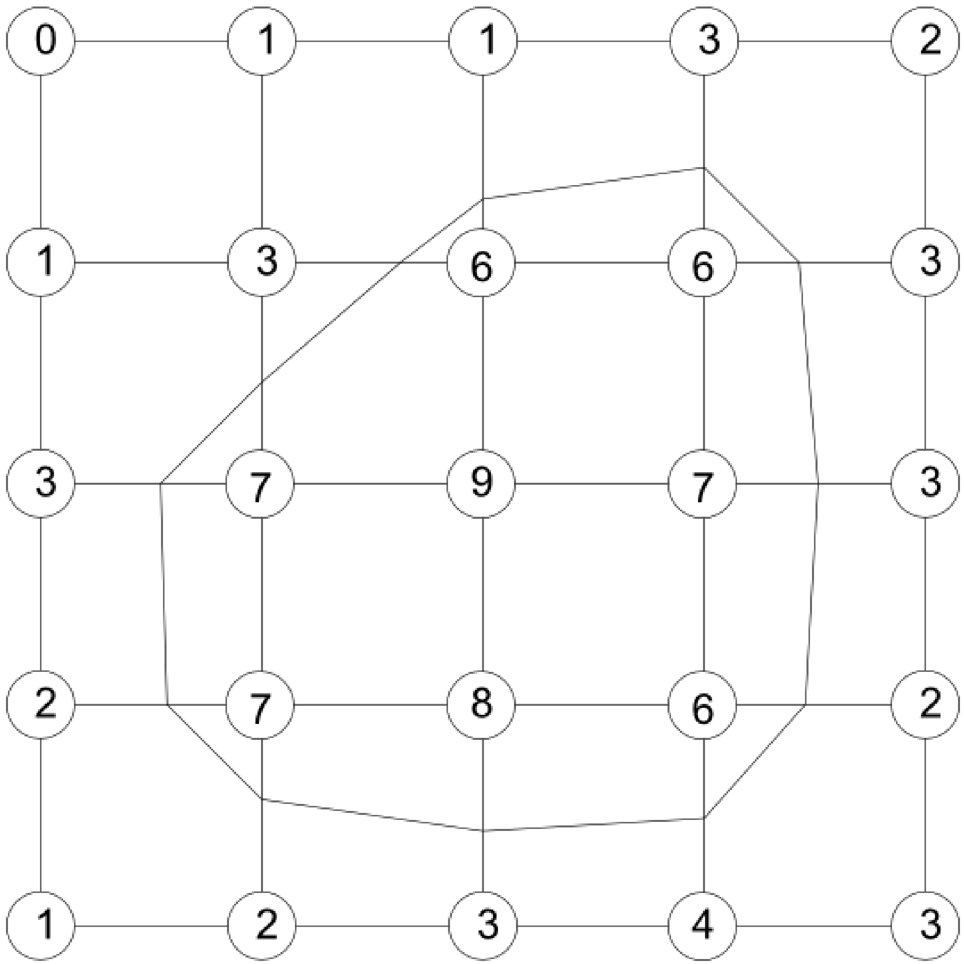
Marching Squares

- **Basic assumption: contour can pass cell in only finite number of ways**
- **4 vertices make up $2^4 = 16$ states**
- **Construct case table enumerating all possible topological states**
- **Case table considers how contour passes through cell (topology), not where it passes (geometry)**



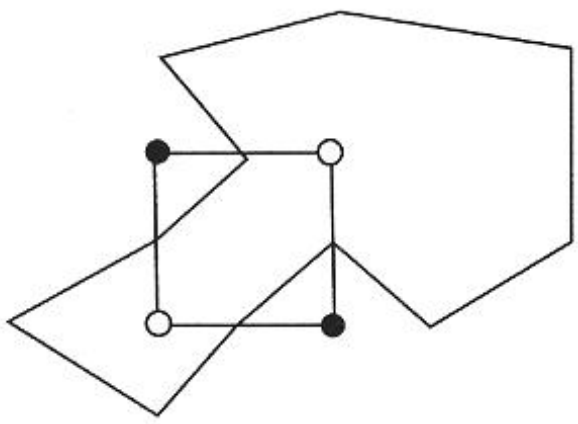
Isolines

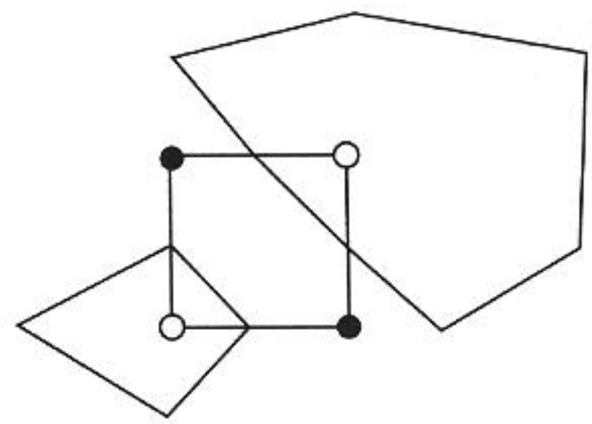
Draw $f(x, y) = 5$



Marching Squares: ambiguity

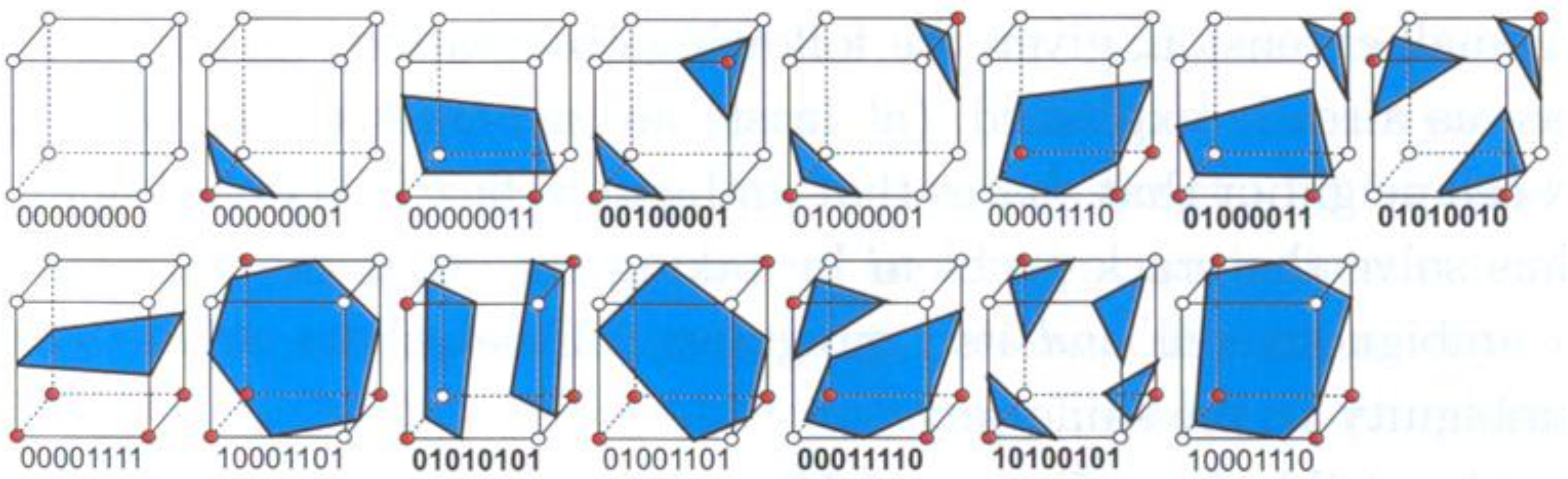
- Some cases can be contoured in more than one way (cases 5 and 10)
- Implement one of the two possibilities
- Choice leads to extending or breaking of a contour



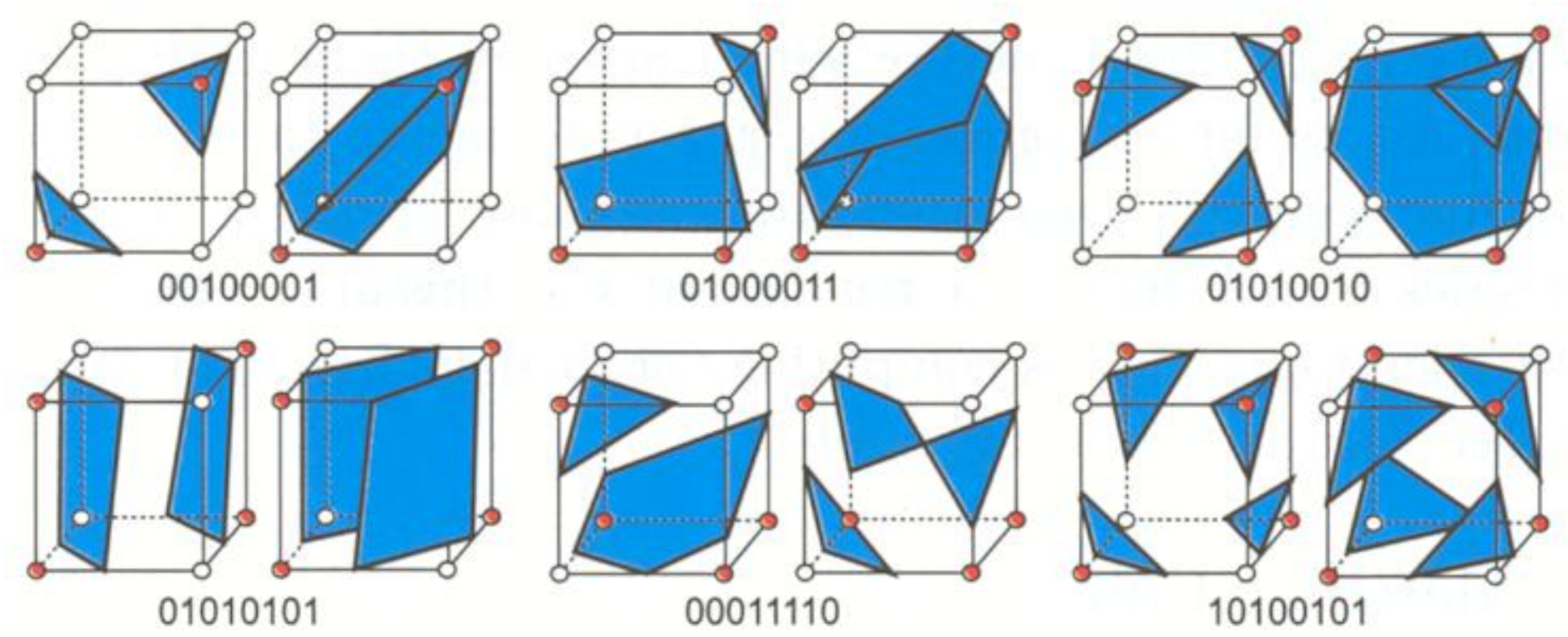


Marching Cubes

- **3D version of Marching Squares**
- **Computes contour surfaces**
- **Generates triangles instead of lines**
- **Extra step: generate surface normals**

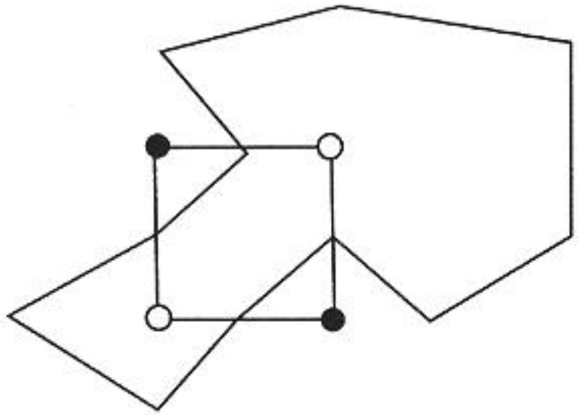


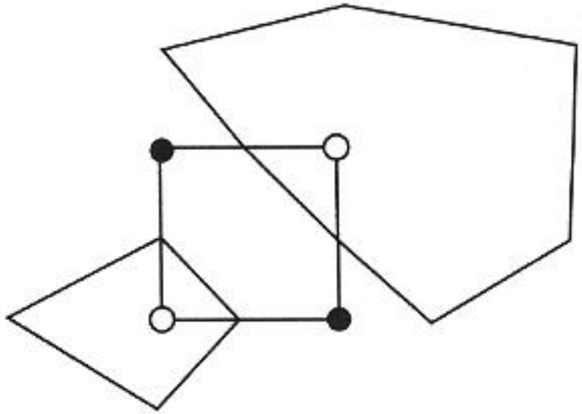
Marching Cubes: ambiguities



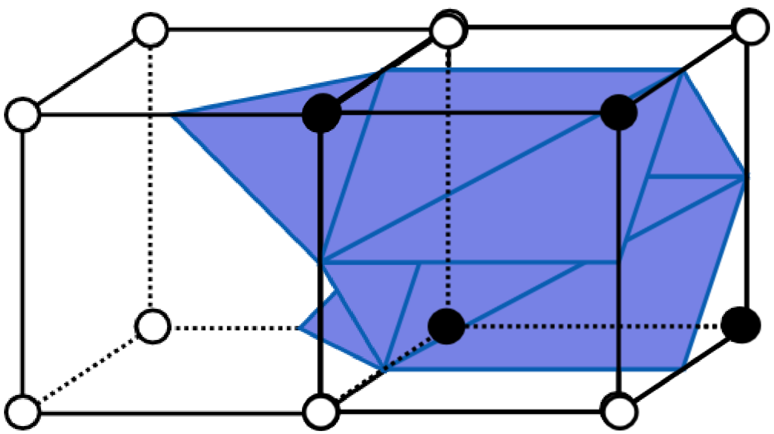
Marching Cubes: ambiguity

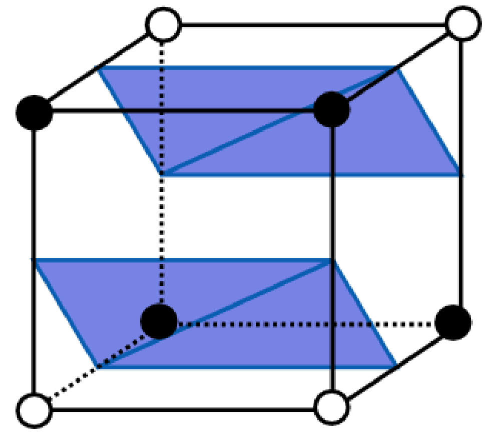
- **In 2-D, each of the two possibilities was equally acceptable**
- **Choice leads to extending or breaking of a contour**
- **In 3-D, however, …**

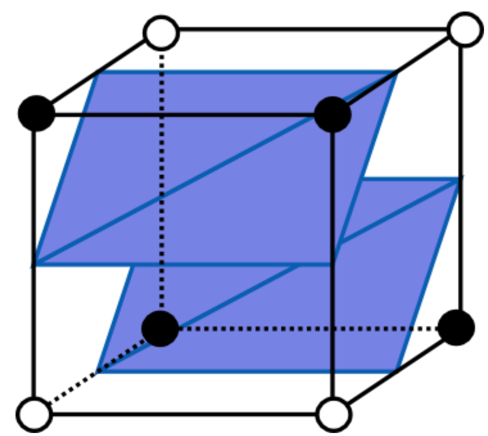




Marching Cubes: ambiguity example

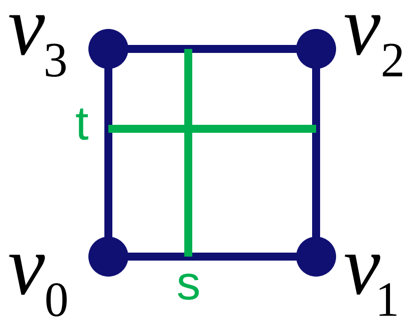






Asymptotic decider

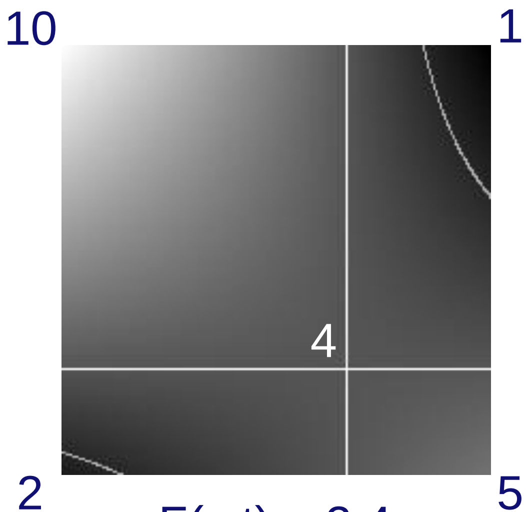
- **Bilinear interpolation over cell**



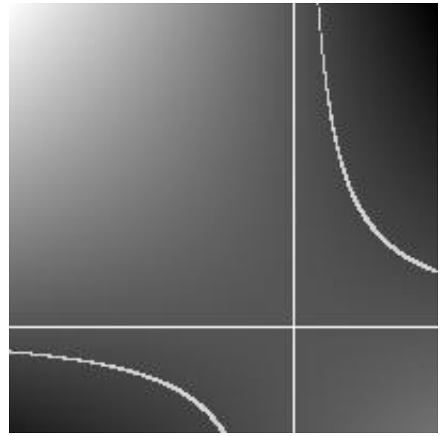
$$
\begin{aligned}
F(s,t) &= (1-s)(1-t)v_0 + s(1-t)v_1 + stv_2 + (1-s)tv_3 \\
&= (v_0 + v_2 - v_1 - v_3)st + (v_1 - v_0)s + (v_3 - v_0)t + v_0 \\
&= Ast + Bs + Ct + D
\end{aligned}
$$

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

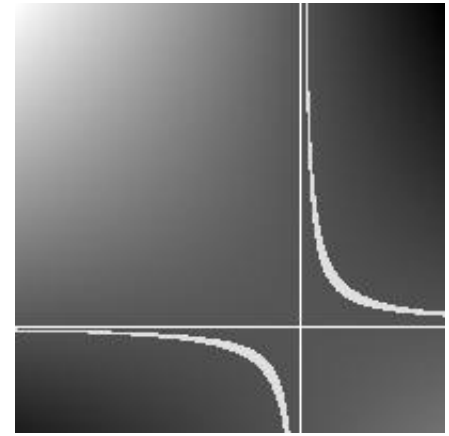
Asymptotic decider: hyperbolas F(s,t)=c



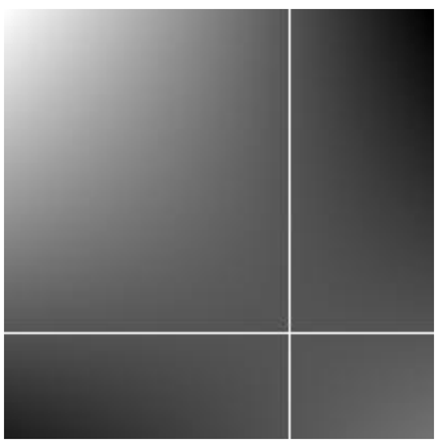
F(s,t) = 2.4



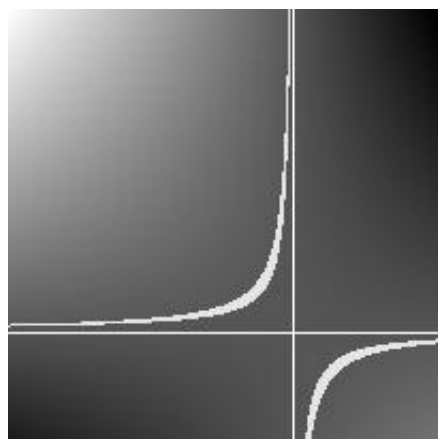
F(s,t) = 3.5



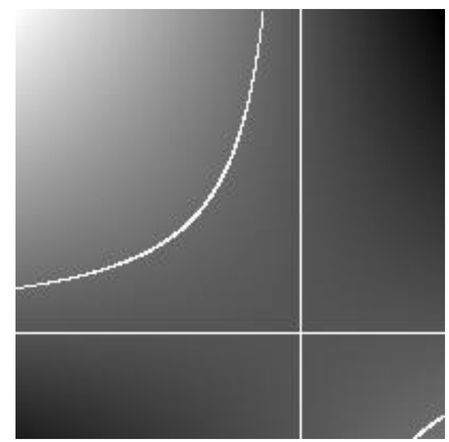
F(s,t) = 3.9



F(s,t) = 4.0



F(s,t) = 4.1



F(s,t) = 4.8

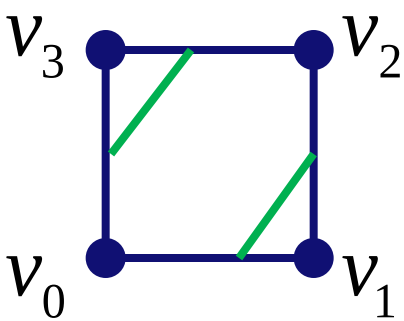
Asymptotic decider

- **Compare isovalue against value of interpolant at intersection of asymptotes**

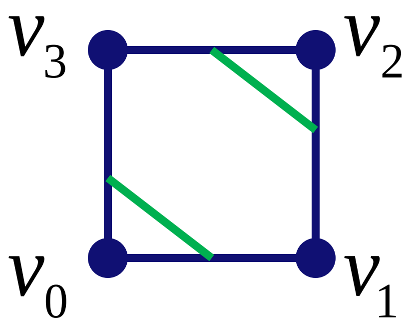
$$F(s,t) = Ast + Bs + Ct + D$$

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

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



$F(s_a, t_a) < isovalue$



$F(s_a, t_a) >= isovalue$

Marching Cubes: surface normals

- **The surface normals are determined by linear interpolation of vertex normals**
- **Each vertex normal is computed by central differences:**

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

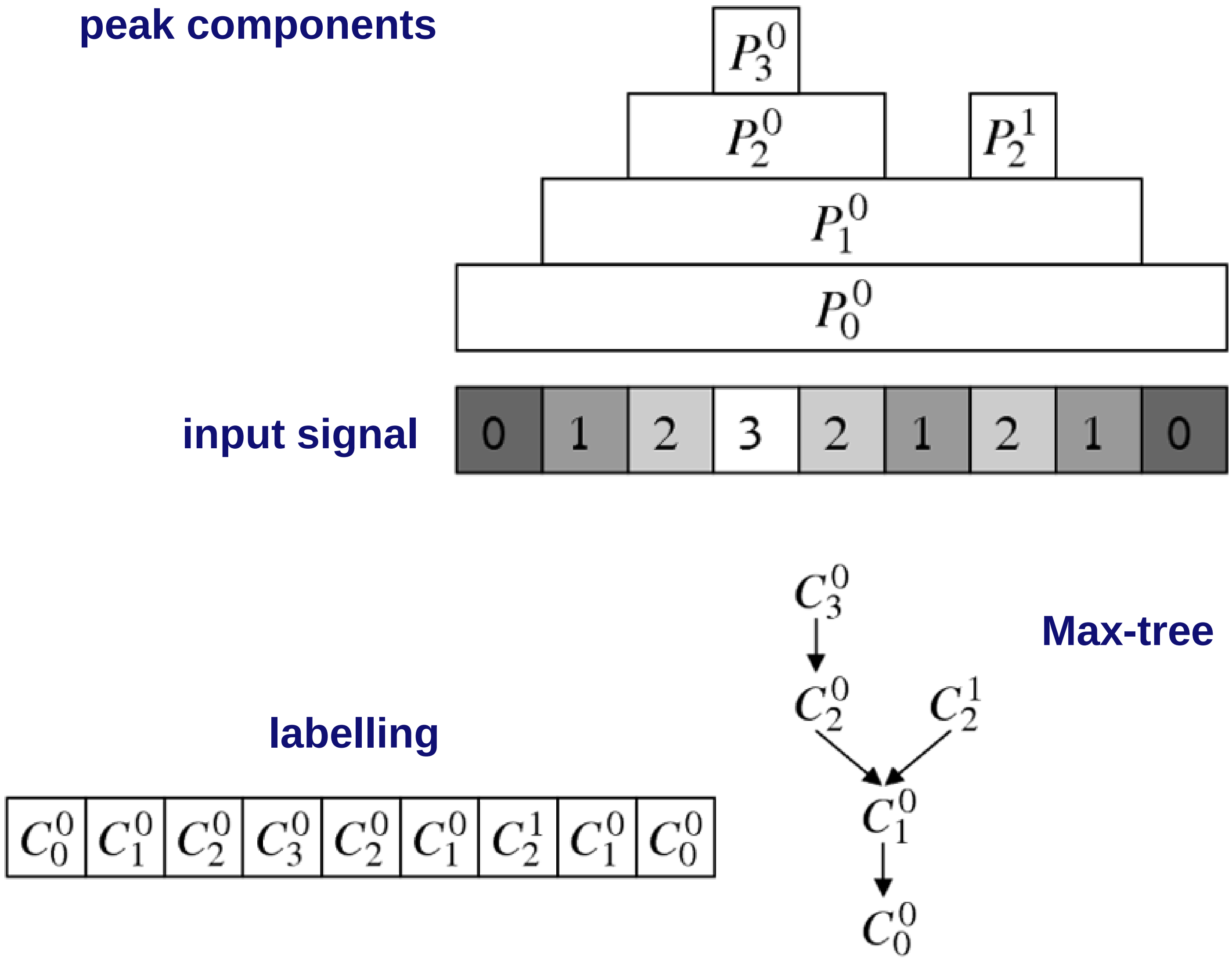
$$\nabla s \approx \frac{1}{2}\begin{pmatrix} s(i+1, j, k) - s(i-1, j, k) \\ s(i, j+1, k) - s(i, j-1, k) \\ s(i, j, k+1) - s(i, j, k-1) \end{pmatrix}$$

Isosurface extraction from the Max-tree

Max-tree

- **Construction**
  - **initial partitioning in regions from regional maxima**
  - **merge regions by nesting of peak components at successive gray levels**
- **Tree encodes nesting of flat zones in peak components**
  - **region model: gray level h of flat zone L_h corresponding to region R_i**
  - **merging order: dictated by nesting of peak components**

Max-tree representation



Filtering and simplification

- Attribute filtering
  - compute size/shape attribute per node
  - label nodes with attribute < threshold
  - tree pruning rules
    - max: prune from leaves up to first ancestor to be preserved
    - direct: remove node and merge members with first ancestor to be preserved
    - subtractive: as direct, but lowers of descendants of removed nodes

Augmenting the Max-tree

- Definition: the root path of a node *C* contains all nodes encountered on the descent from *C* to the root
- In a 26-connected neighborhood
  - all 8 corner voxels of cell are part of same root path
  - filtering does not change ordering of nodes along path
- Therefore
  - one node defines cell’s minimum
  - one other node defines cell’s maximum
- Important: after filtering, the same nodes still define cell’s minimum and maximum

Augmented Max-tree

0	2	1	0
0	3	1	0
0	2	1	0
0	2	1	0

original image

C_0^0	C_2^0	C_1^0	C_0^0
C_0^0	C_3^0	C_1^0	C_0^0
C_0^0	C_2^0	C_1^0	C_0^0
C_0^0	C_2^0	C_1^0	C_0^0

label image

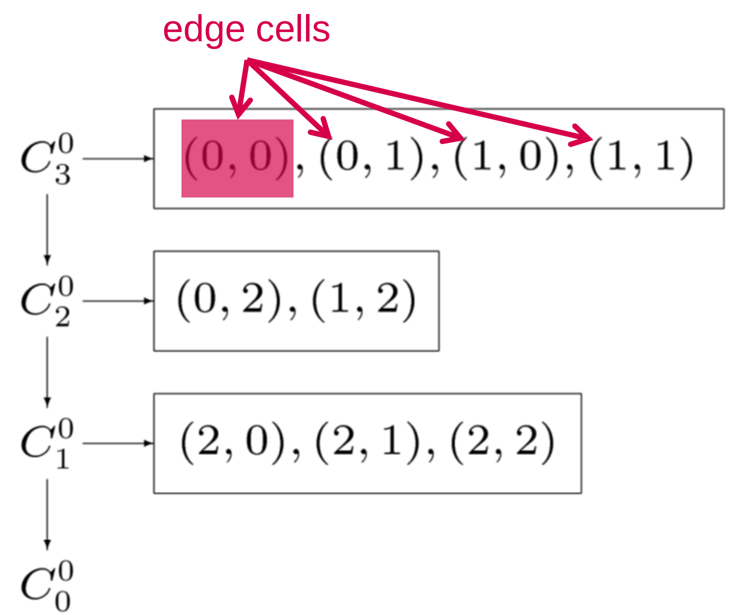
C_0^0	C_1^0	C_0^0	
C_0^0	C_1^0	C_0^0	
C_0^0	C_1^0	C_0^0	

V_{min}

C_3^0	C_3^0	C_1^0	
C_3^0	C_3^0	C_1^0	
C_2^0	C_2^0	C_1^0	

V_{max}

- c is edge cell when $V_{\text{min}} \neq V_{\text{max}}$
- store c in tree node corresponding to V_{max}
- sort edge cells in ascending order of V_{min}

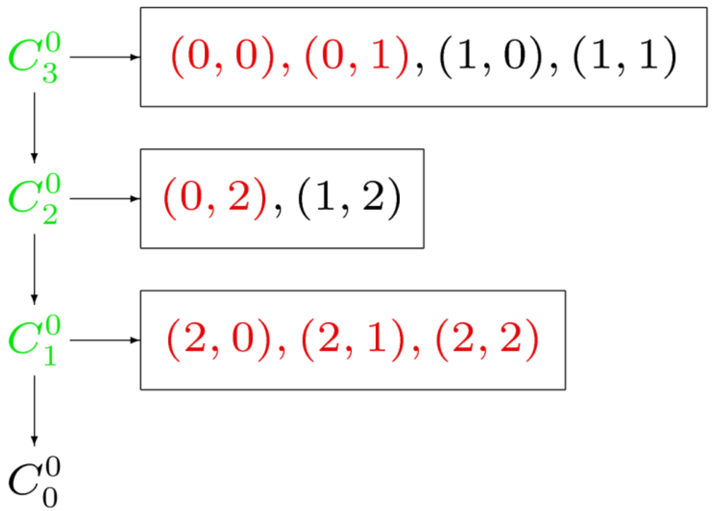


Augmented Max-tree

Algorithm

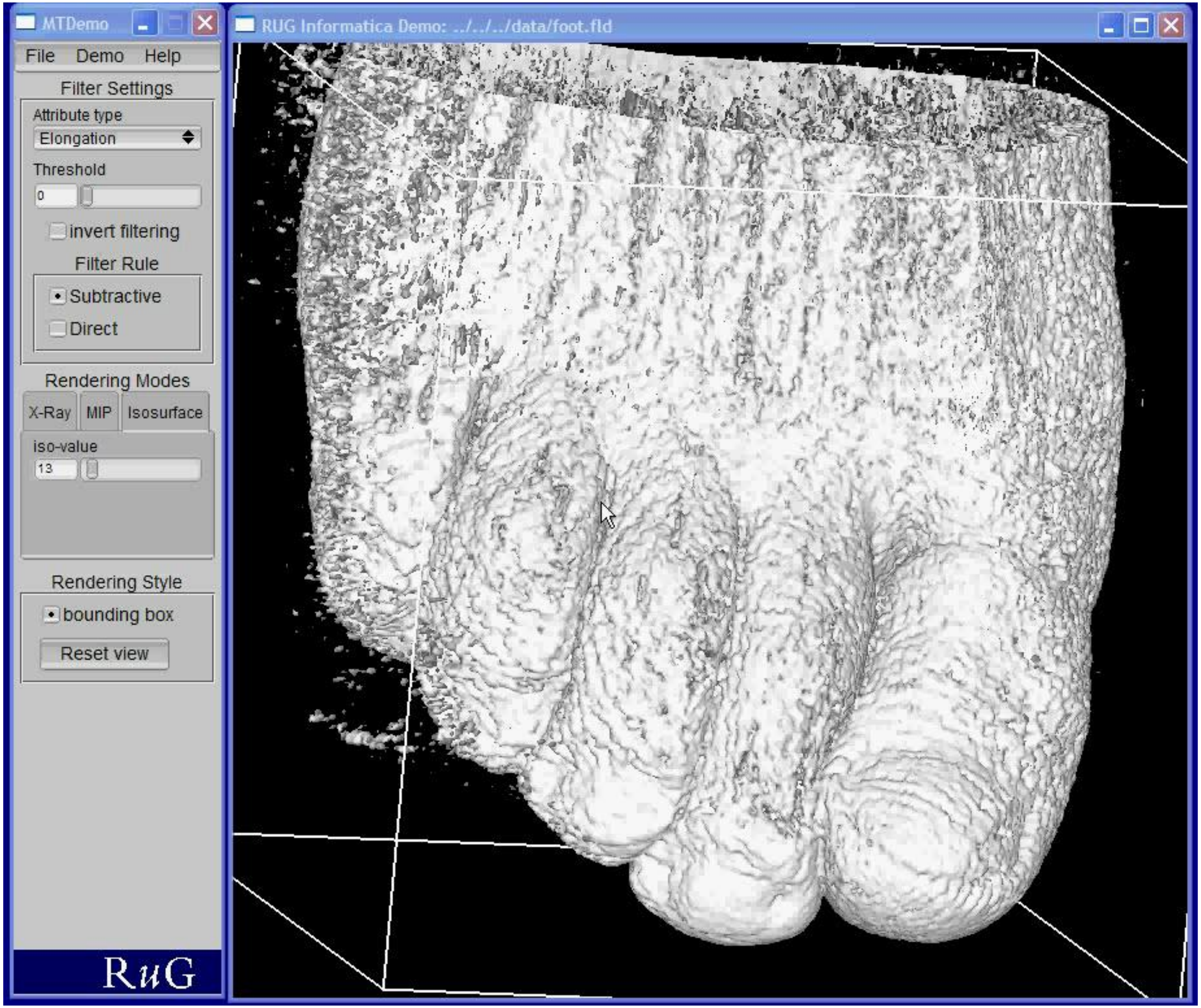
Pseudo-code

for all nodes p **do**
  p.processed $\leftarrow$ false
end for
root.processed $\leftarrow$ true
for all leaves q **do**
  p $\leftarrow$ q
  while (not p.processed) and $(g(\mathrm{p}) \geq t)$ **do**
    i $\leftarrow$ 0
    while (i<p.numEdges) and $(g(V_{\mathrm{min}}(c_i^p)) \leq t)$
    do
      mark c_i^p as active
      i $\leftarrow$ i + 1
    end while
    p.processed $\leftarrow$ true
    p $\leftarrow$ p.parent
  end while
end for



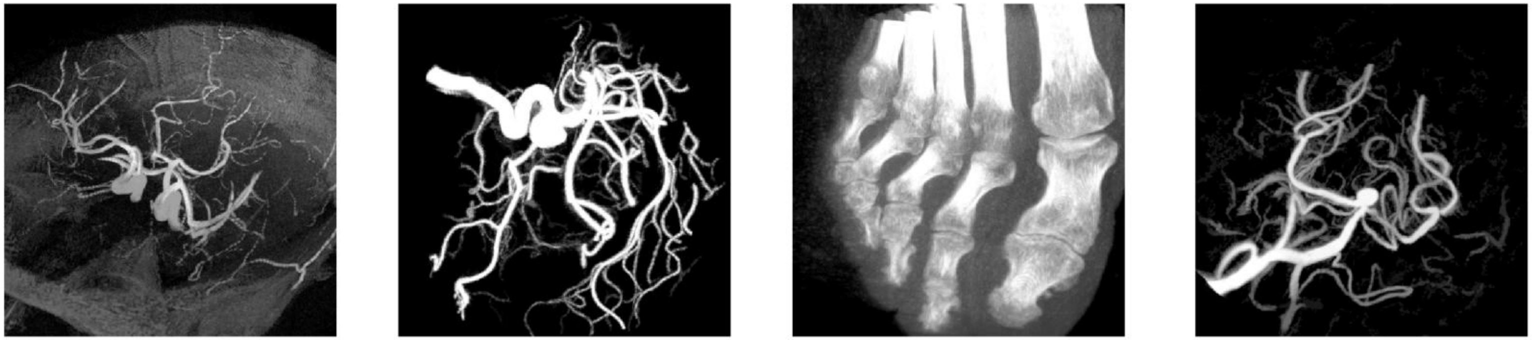
Visited nodes and **cells** for $t = 0.5$.

Isosurface demo (movie)



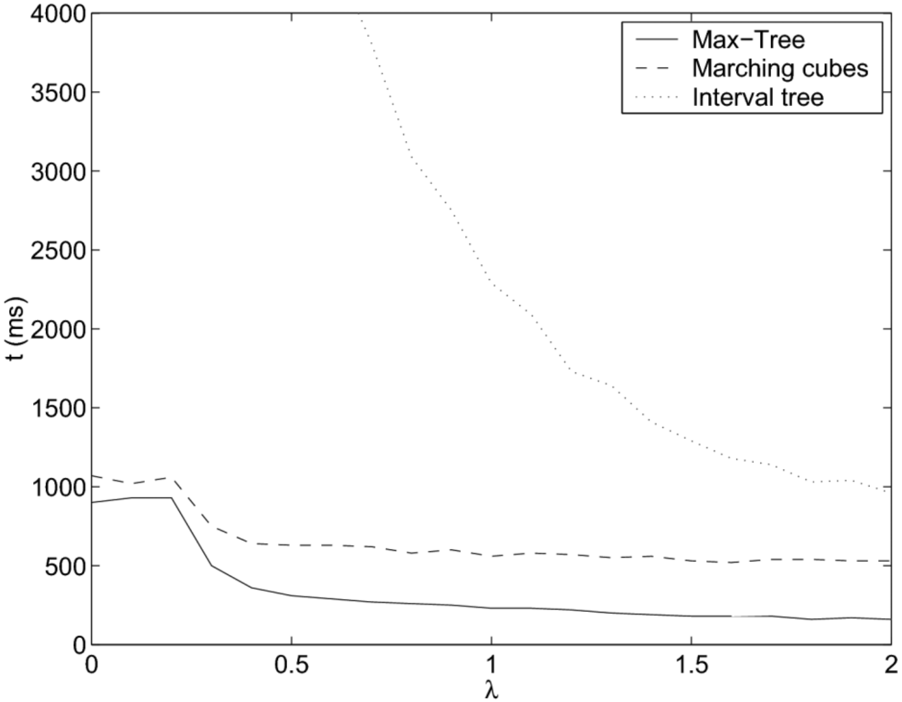
Data sets

Name	Data set size	Dynamic range	Max-Tree nodes	M-T build time (s)
Angiogram	$23,855,104$	10-bits	1,554,454	16.2
Aneurism I	$16,777,216$	8-bits	38,868	11.1
Foot	$16,777,216$	8-bits	279,513	33.8
Aneurism II	$66,846,720$	8-bits	505,952	100.0

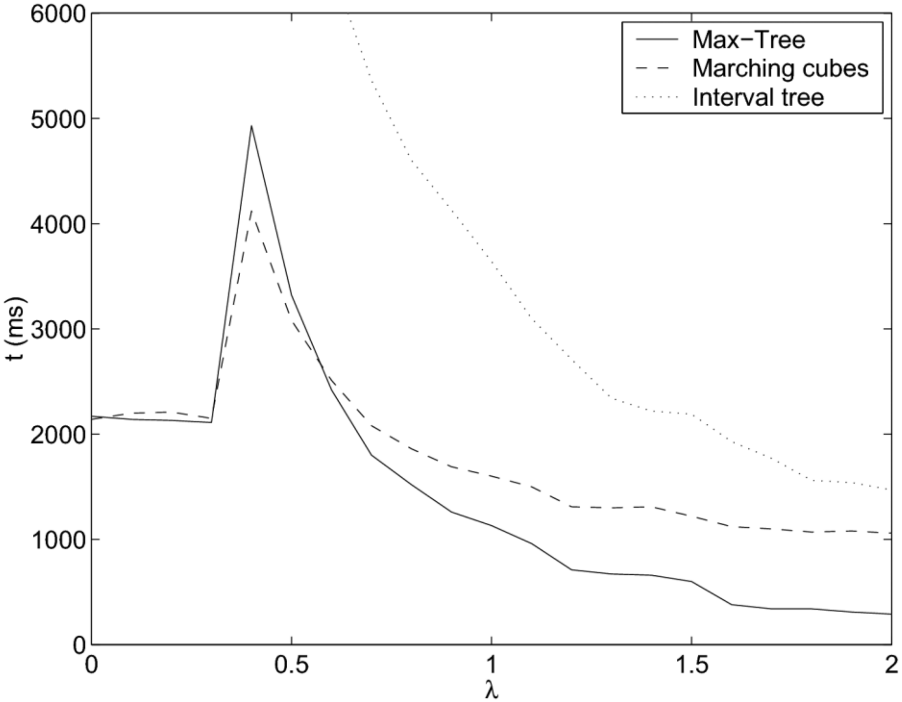


(a) Angiogram (b) Aneurism I (c) Foot (d) Aneurism II

Performance – filter threshold browsing

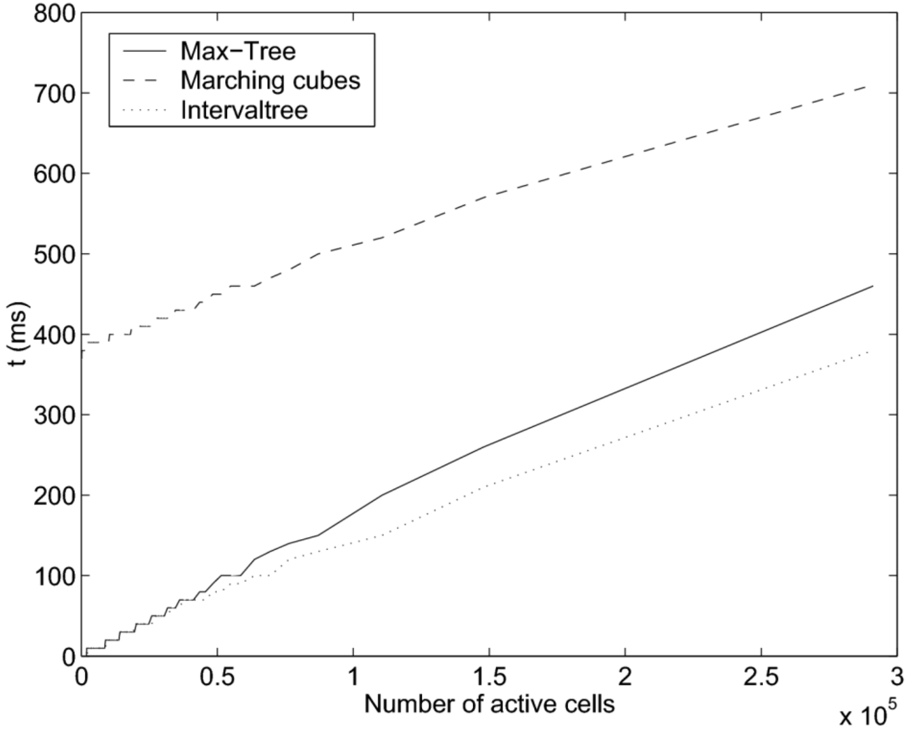


Angiogram, $t = 50$

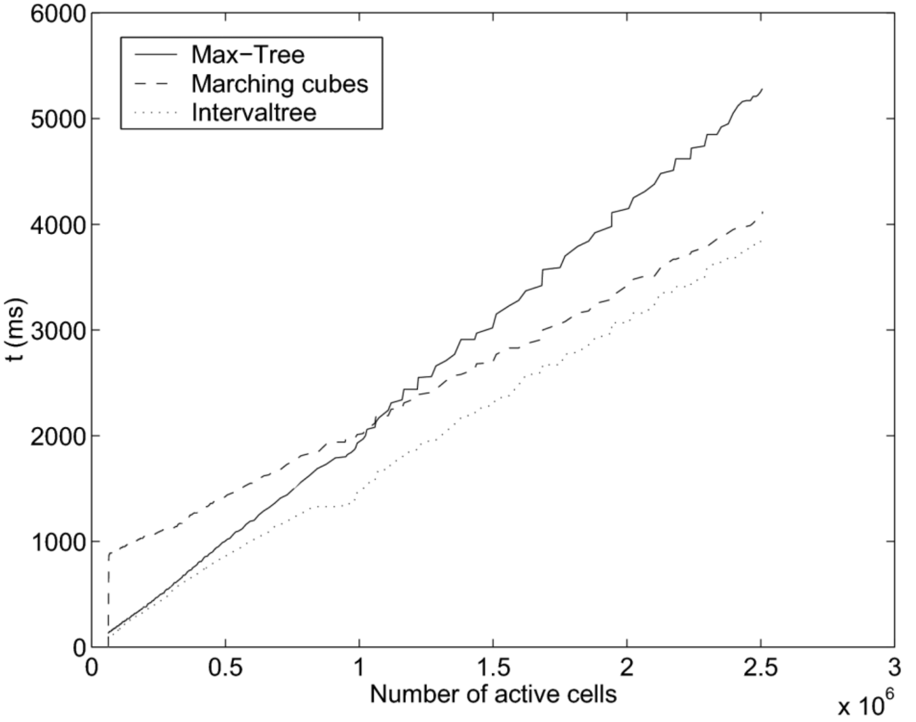


Foot, $\lambda = 10$

Performance – iso-surface browsing



Angiogram, $\lambda = 2.0$



Foot, $\lambda = 0$

Direct volume rendering from the Max-tree

Drawbacks of isosurfacing

- Only approximation of a surface
- Loss of information
- Amorphous phenomena have no surface, e.g. clouds

Direct volume rendering

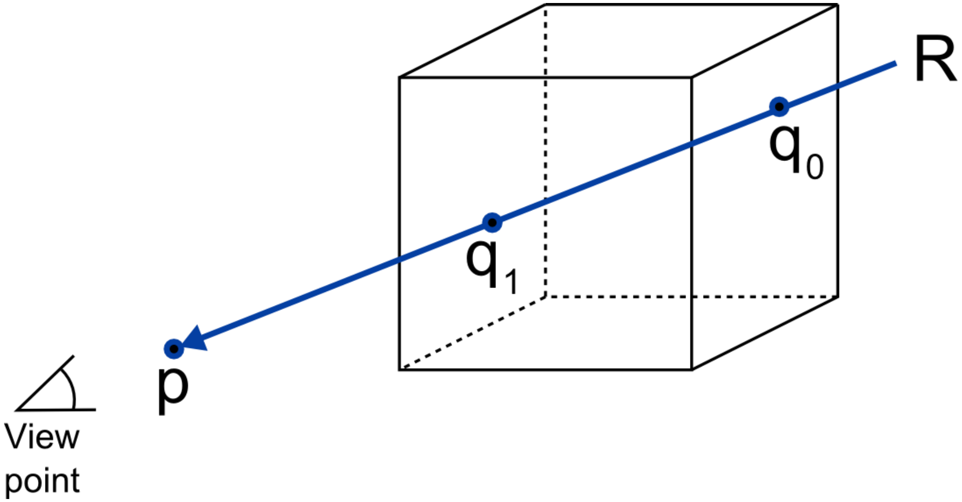
- **Principle: rendering of scalar volume data with cloud-like, semi-transparent effects**

surface rendering

direct volume rendering

Ray casting

- **Consider ray R perpendicular to image plane ending in pixel p**



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

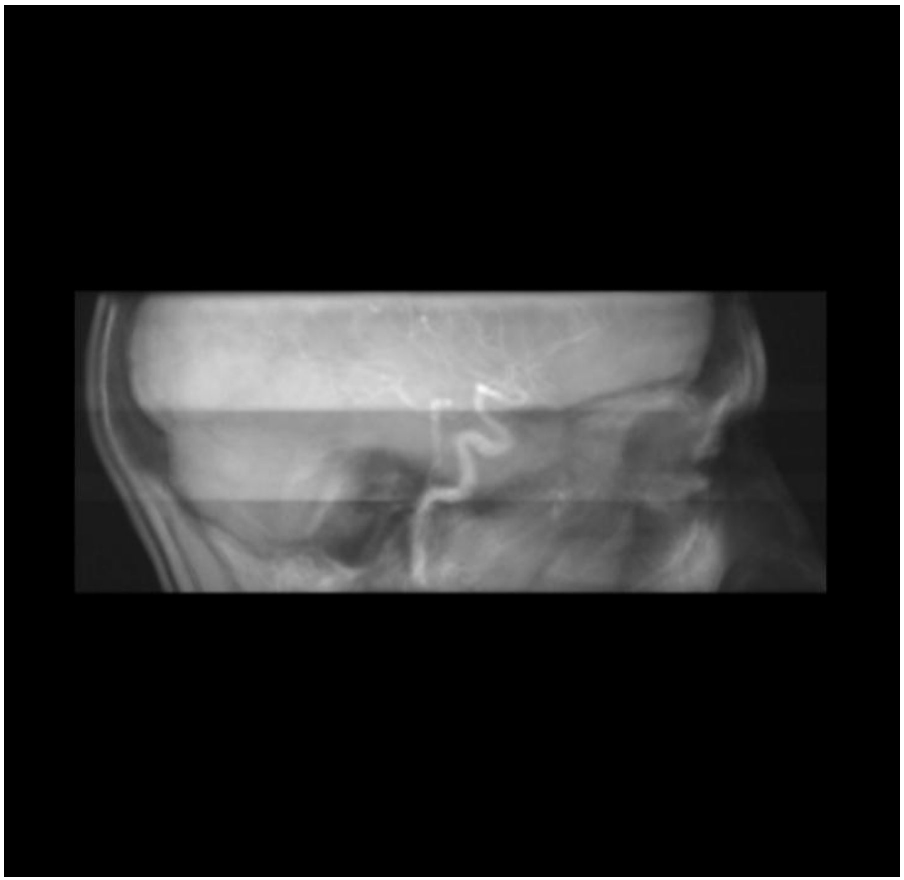
scalar values along R: $s(t) = f(q(t))$

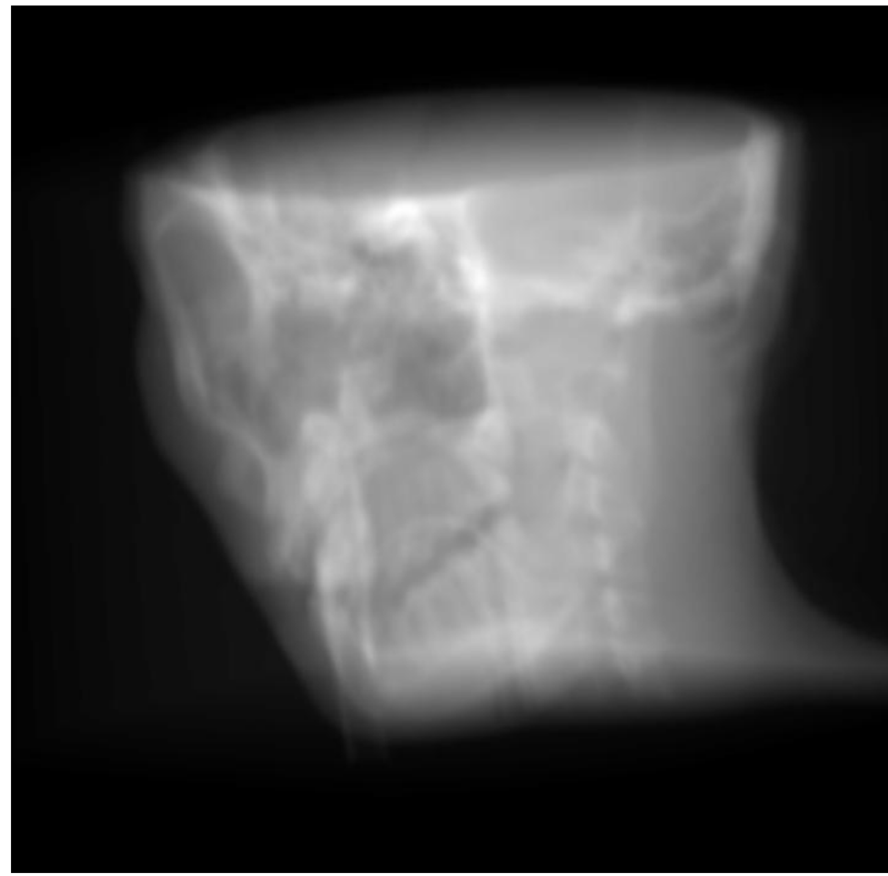
pixel value in p: $I(p) = F(s(t))$

ray function

X-ray projection

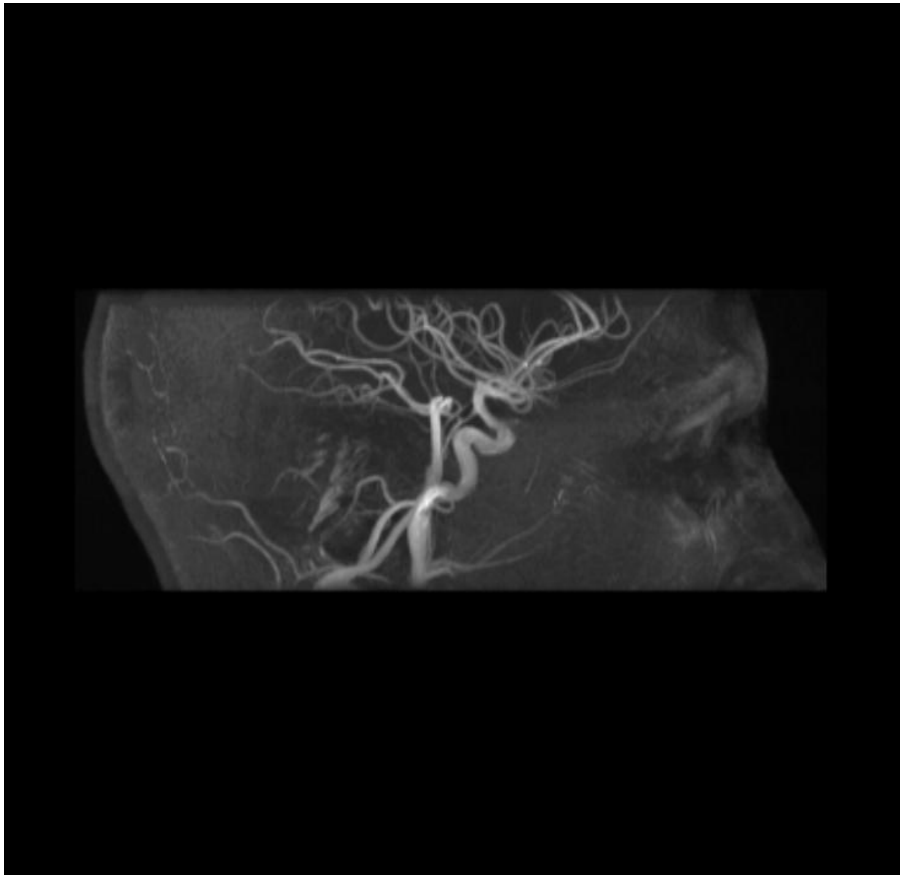
- **Integrate along ray:** $$I(p) = \int_0^1 s(t)\,dt$$

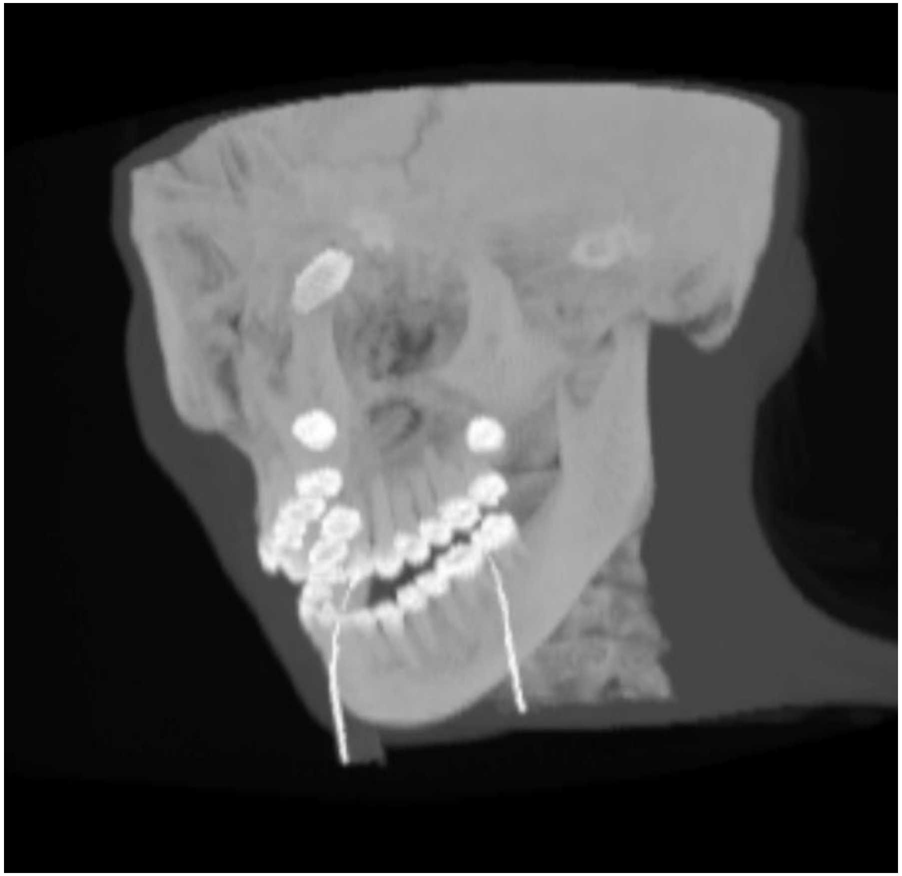




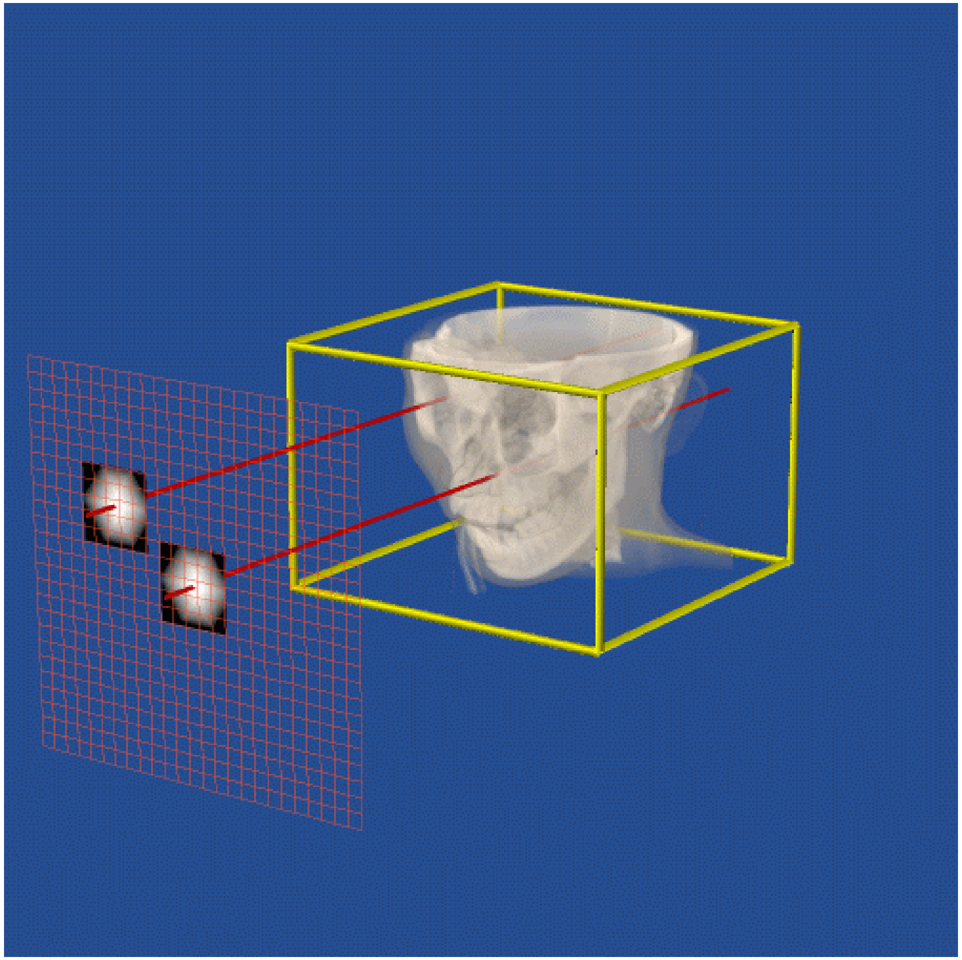
Maximum intensity projection (MIP)

- **Take F = max:** $I(p) = \max(s(t))$





Splatting



Reconstruction from samples

- **Given discrete function, reconstruct continuous function**

$$f(x,y,z) = \sum_i \sum_j \sum_k \tilde{f}(i,j,k)\,h(x-i,y-j,z-k)$$

Footprints

- **Integrate *f* along one of its dimensions**

$$
\begin{aligned}
I(x,y) &= \int f(x,y,z)\,dz \\
&= \int \sum_i \sum_j \sum_k \tilde{f}(i,j,k)\,h(x-i,y-j,z-k)\,dz \\
&= \sum_i \sum_j \sum_k \tilde{f}(i,j,k) \int h(x-i,y-j,z-k)\,dz
\end{aligned}
$$

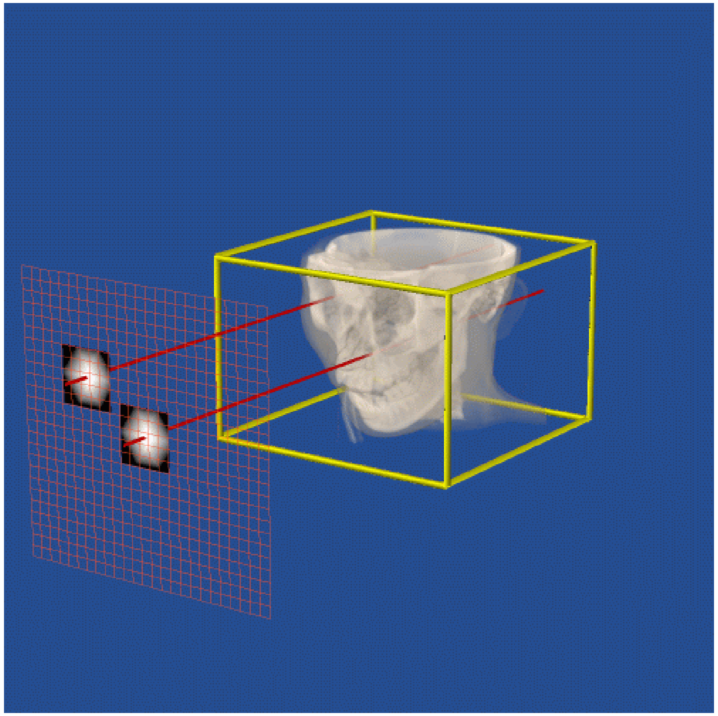
- **Define footprint *H***

$$
H(x,y) = \int h(x,y,z)\,dz
$$

Accumulation

- **Final image is an accumulation of weighted footprints**

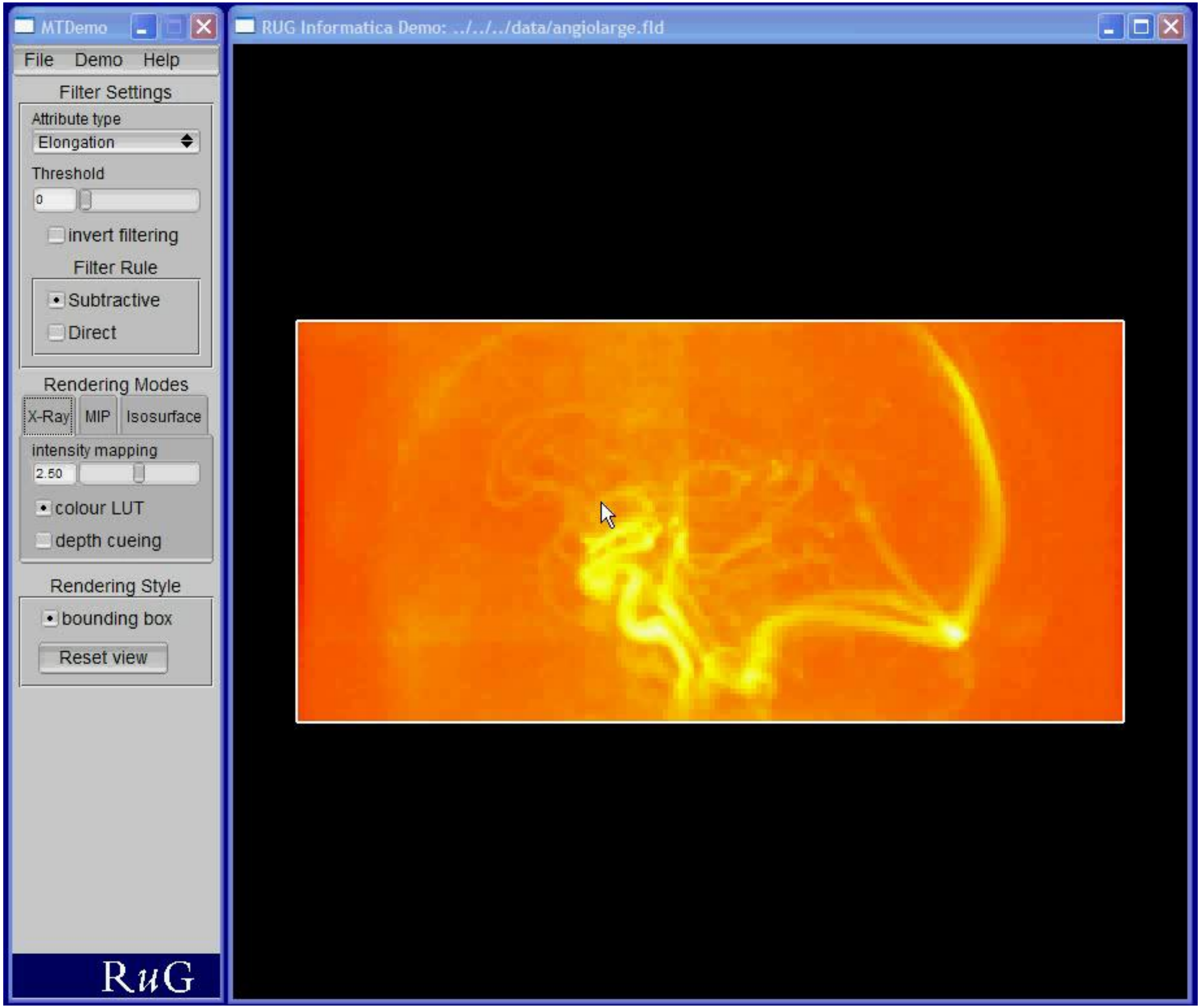
$$I(x,y) = \sum_i \sum_j \sum_k \tilde{f}(i,j,k) H(x-i, y-j)$$



Splatting from a Max-tree

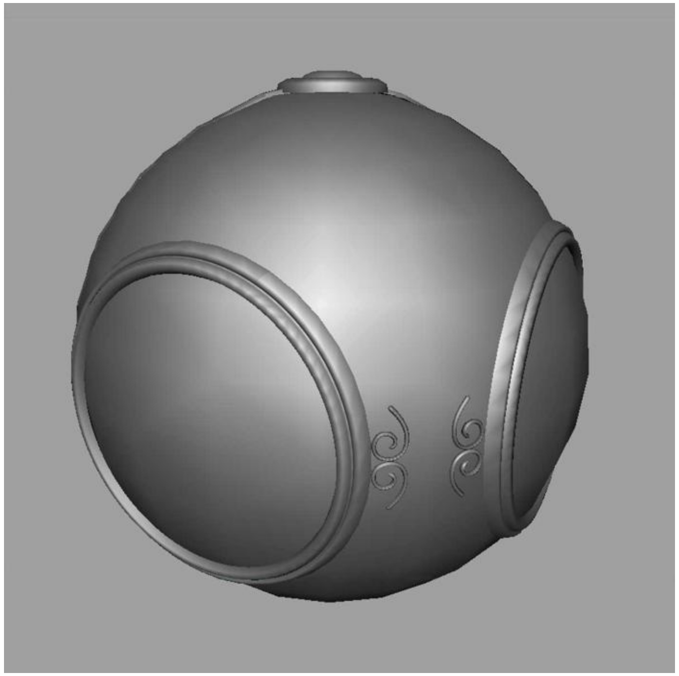
- Volume reconstruction not desirable
  - time consuming
  - non-zero voxels do not contribute to final image; visiting these wastes time
- Adapt Max-tree representation
  - add voxel list to each node
- Rendering algorithm
  - start at leaves, and splat voxels of nonzero nodes

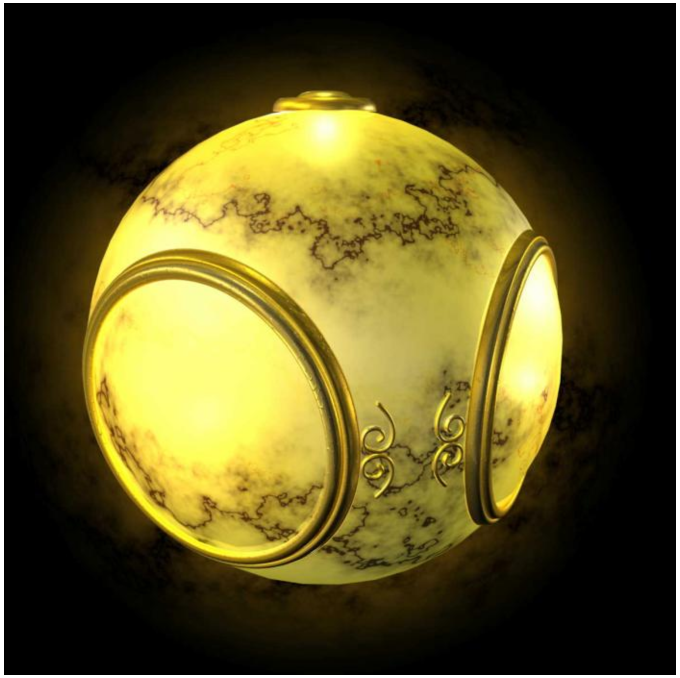
Splatting demo (movie)



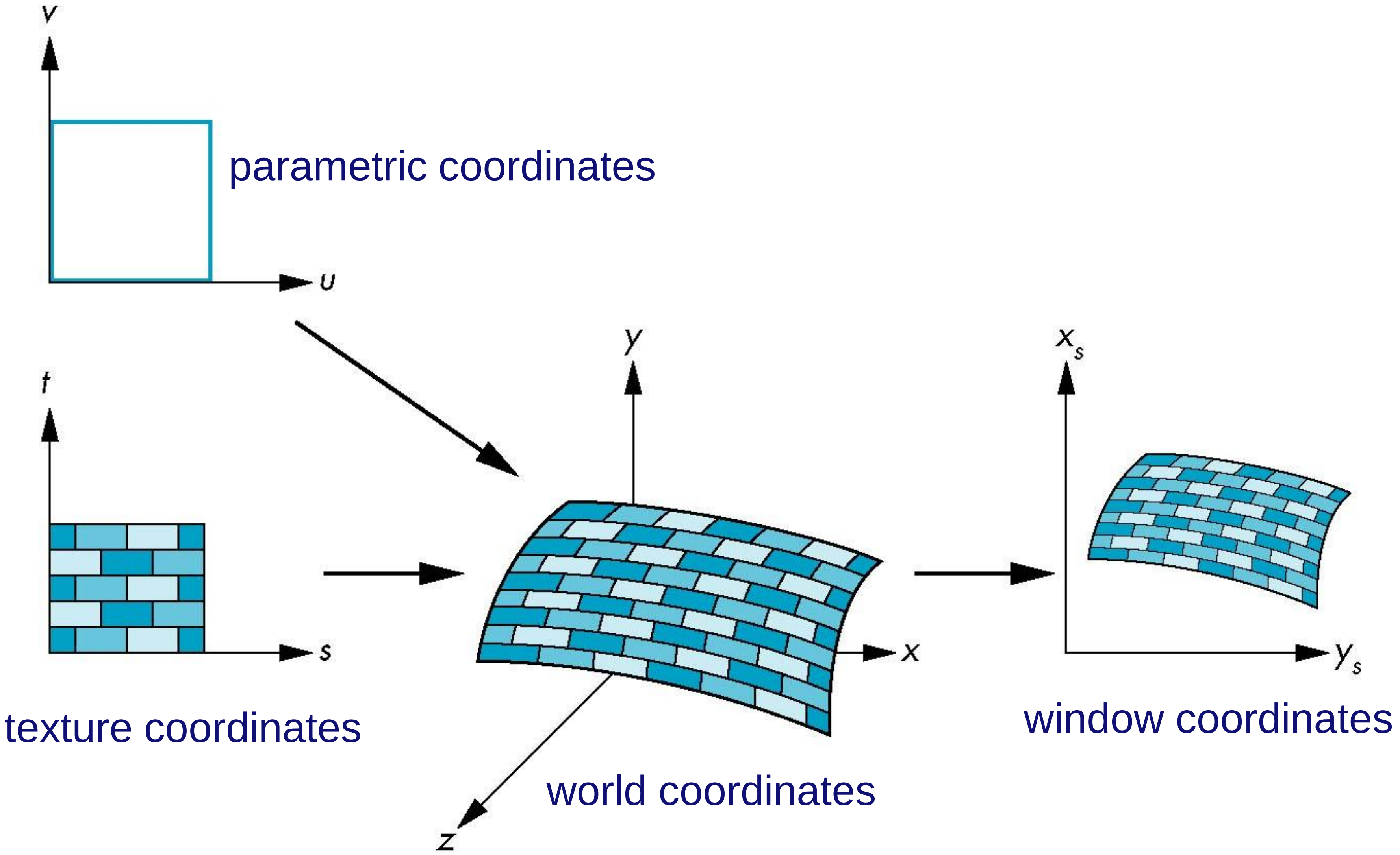
Texture mapping

- **Trick from computer graphics to make objects look more realistic**
  - **use simpler shape**
  - **paste image of more realistic looking object on it**



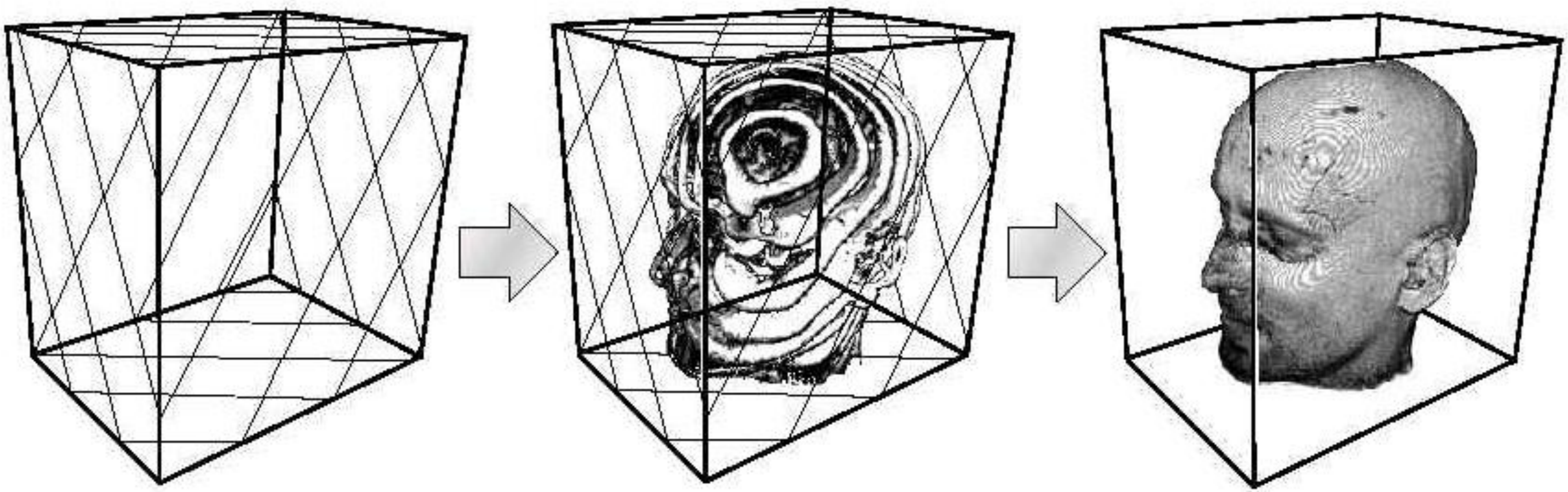


Texture mapping



Texture-based volume rendering

- **Idea: load volume data into texture memory, and use graphics hardware to resample and blend**



Texture-based rendering

- **Volume reconstruction undesirable**
  - **time consuming: reconstruction and transfer to GPU**
- **Solution**
  - **construct label volume and store on GPU in texture T_V**
    label corresponds to Max-tree node
  - **encode grey levels of nodes in another texture T**
  - **filtering requires only update of T**
  - **drawing requires an additional step to resolve labels**

Texture-based rendering algorithm

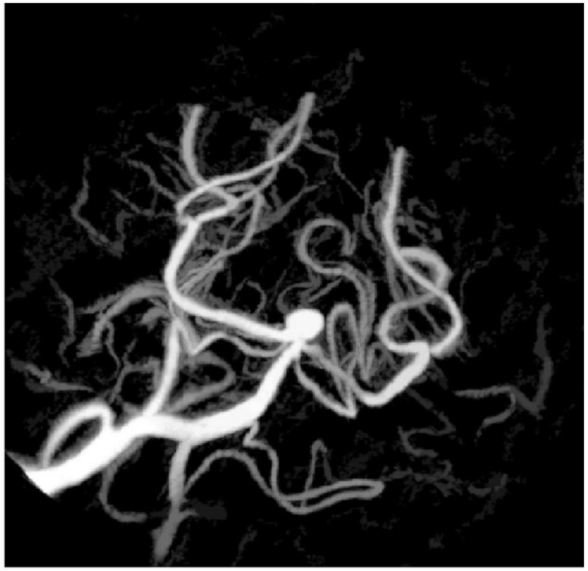
Update()

$\{g(p)$ denotes current grey value of node $p\}$
for $i \leftarrow 0$ to length(A) **do**
$T[i] \leftarrow g(A[i])$
end for
Transfer T to graphics hardware

Draw()

for all slice planes s **do**
for all fragments f in s **do**
$i \leftarrow$ sample T_v in point f {Fetch node index}
$g \leftarrow T[i]$ {Fetch current grey value}
f.color $\leftarrow$ TransferFunction(g)
end for
Blend s with frame buffer
end for

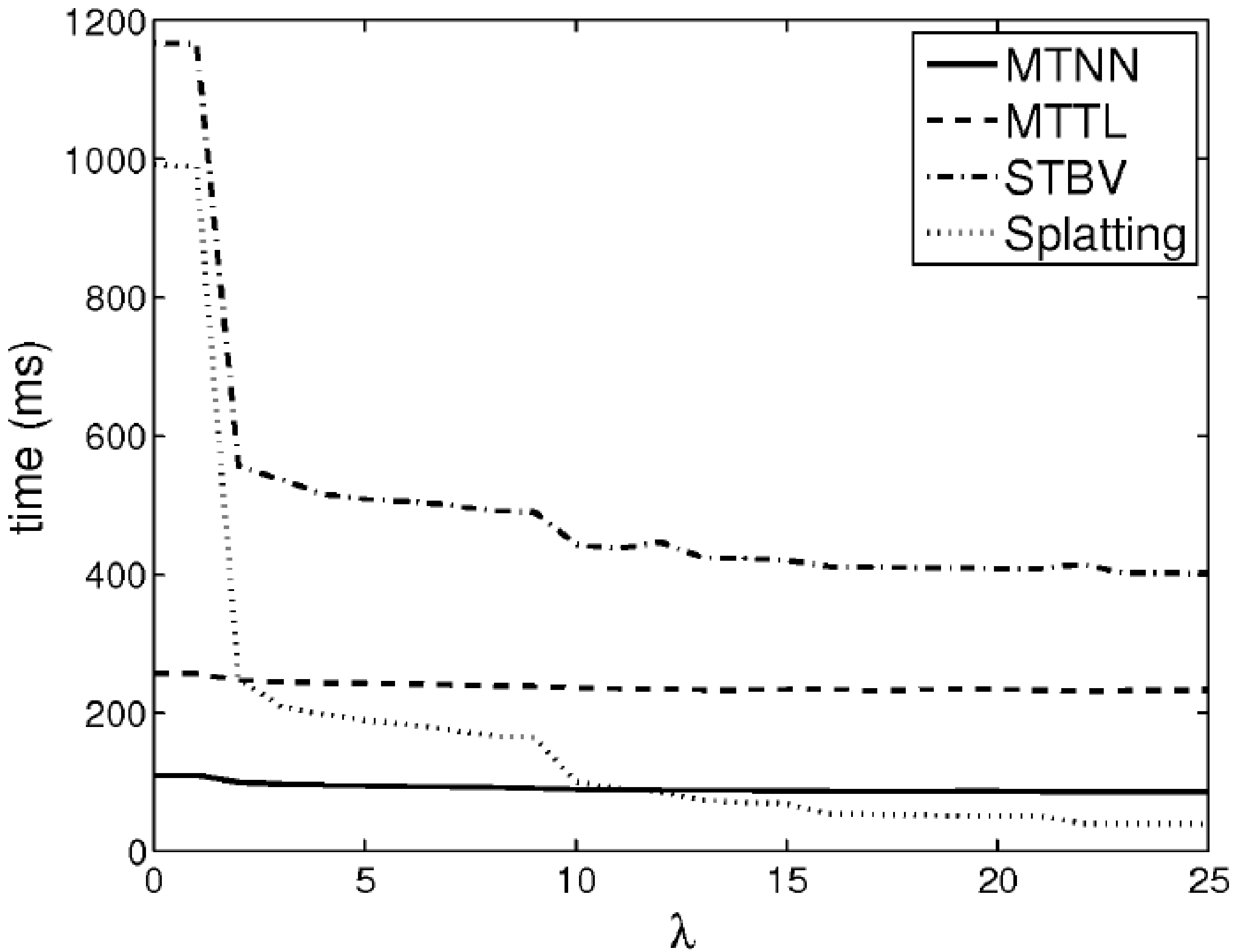
Data sets



Name	Data set size	Dynamic range	Max-Tree nodes	M-T build time (s)
Piggy bank	11,796,480	12-bits	206,529	14.7
Foot	16,777,216	8-bits	279,513	16.4
Aneurism	66,846,720	8-bits	505,952	100.0

Browsing filter threshold

Foot



Performance overview

Data set	Filter	Max-Tree			STBV	
			MTNN	MTTL		
		transfer	draw	draw	transfer	draw
Piggy bank	20	40	20	149	367	12
Foot	27	51	19	143	465	12
Aneurism	33	92	276	2575	1985	17

times in milliseconds

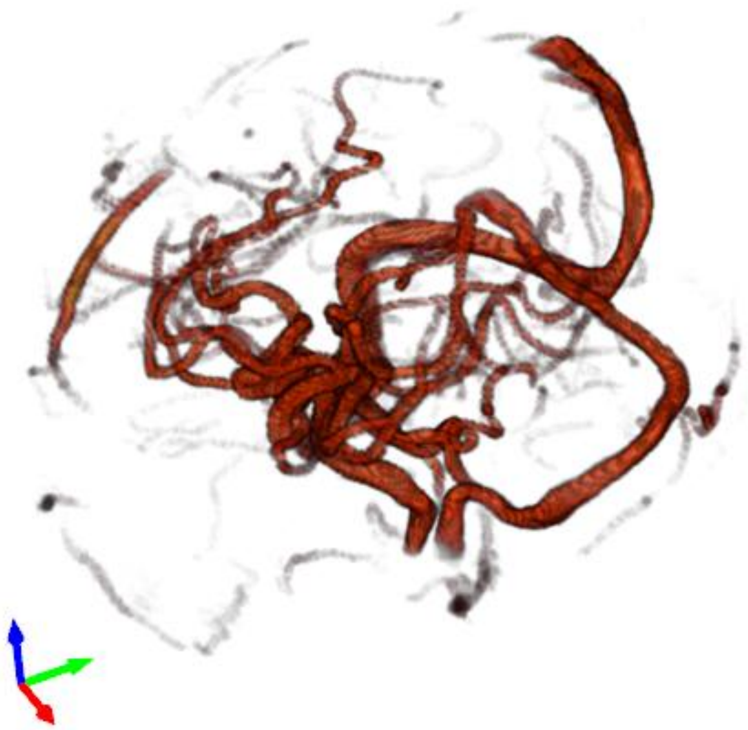
Max-tree-based transfer functions

Visualization – transfer function

- Map data value to color and opacity
  - standard TF
    - gray value to color from heated body color map
    - opacity from gradient magnitude
  - tree-based TF
    - region to color (each node random color at initialization)
      - along root path inspect attribute and flag node if change above some threshold
      - propagate colors of flagged nodes through tree
    - opacity from gradient magnitude

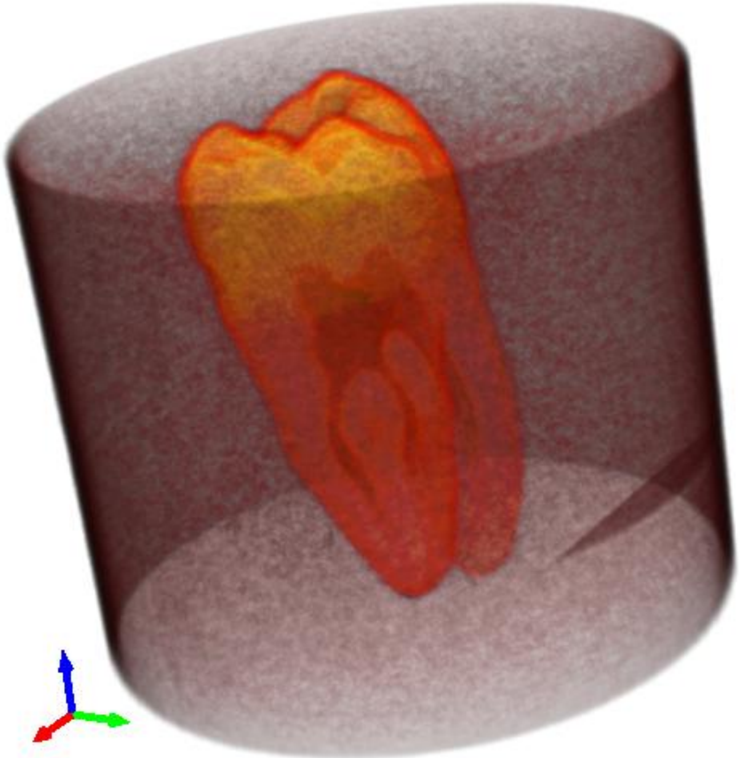
Examples

elongation criterion



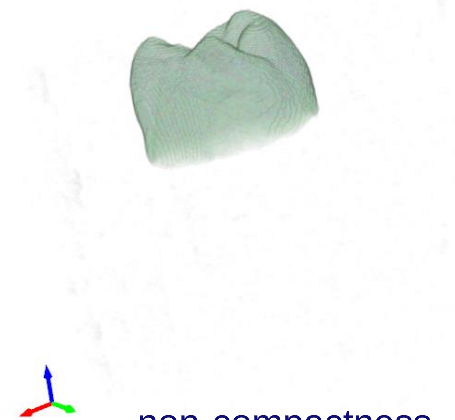
non-compactness criterion

Tooth

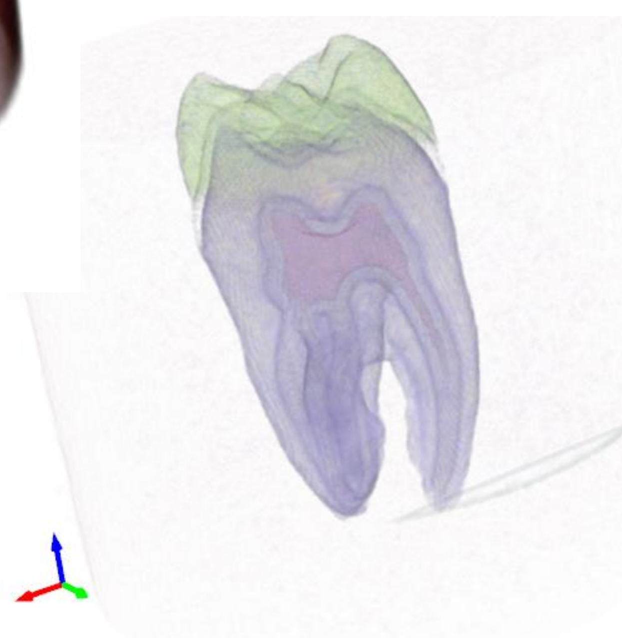


unfiltered

flatness

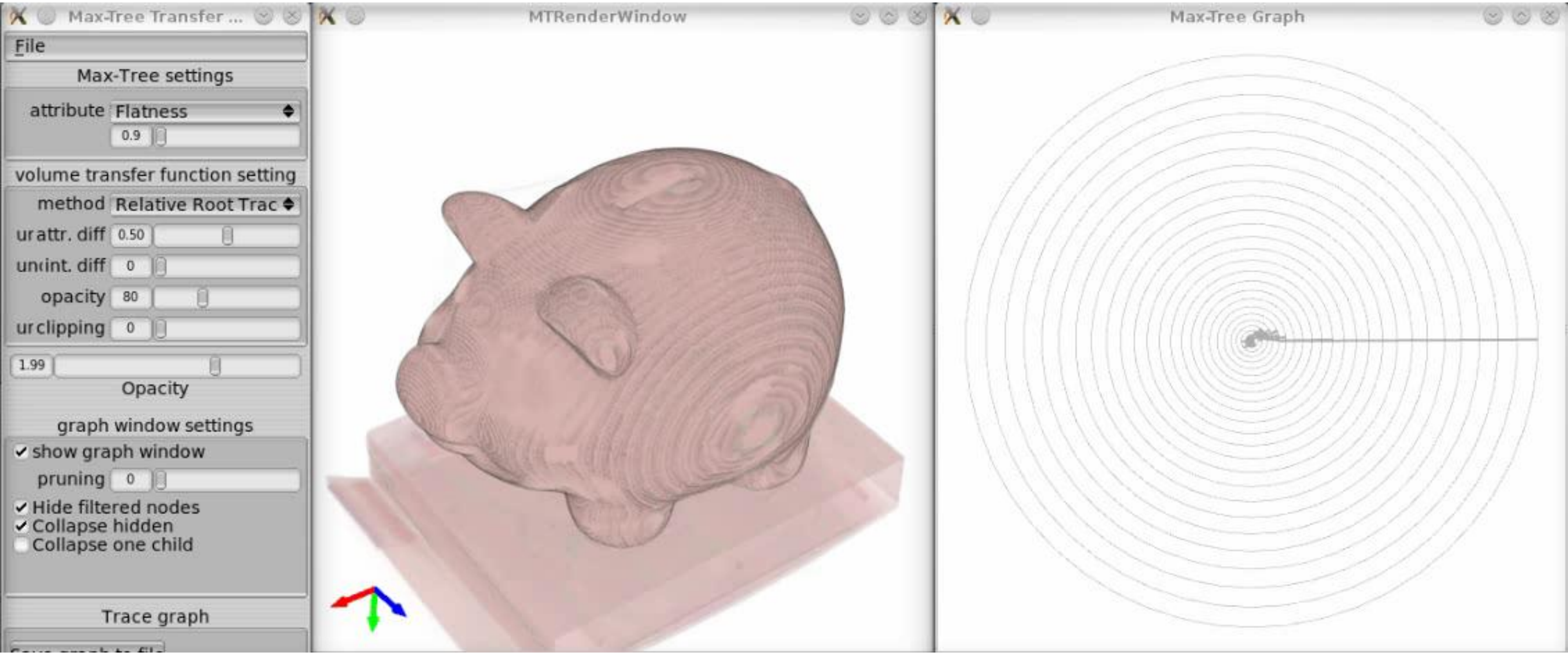


non-compactness



min-tree on gradient magnitude non-compactness and flatness

Piggy bank exploration



Conclusion

- Discussed several extensions to Max-tree to allow for efficient visualization
- Approach can be used for other connected filters