Volume visualisation

COM3404

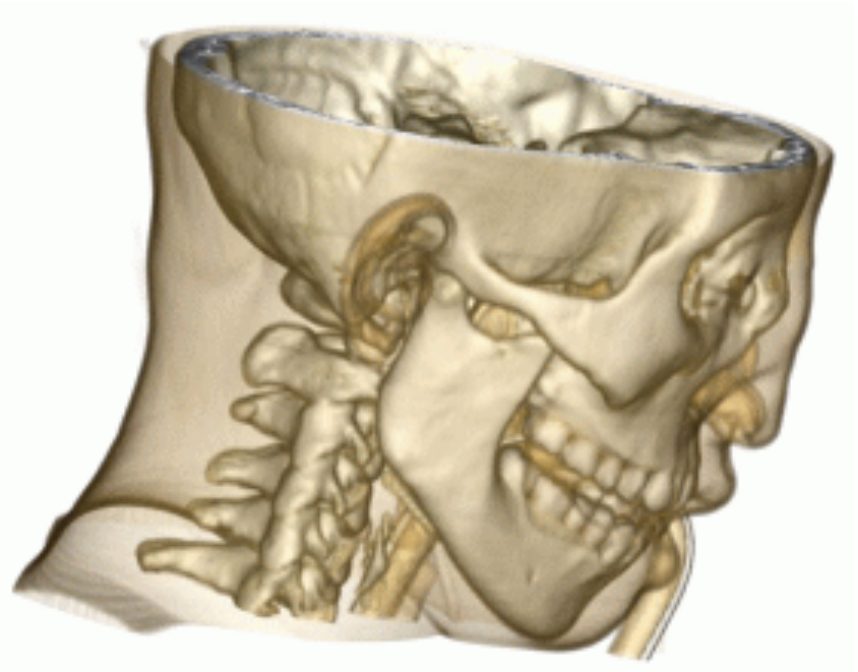
Richard Everson

School of Engineering, Computer Science and Mathematics
University of Exeter

R.M.Everson@exeter.ac.uk
http://www.secamlocal.ex.ac.uk/studyres/COM304

Outline

1. Voxel data
2. Contouring
   - Marching squares
3. Isosurfaces: Marching cubes
4. Volume rendering
   - Transfer function
   - Transformation
   - Compositing
5. Splatting



References

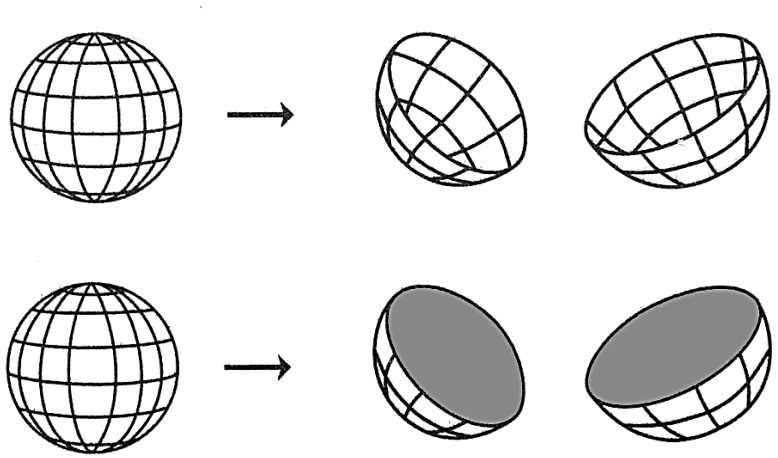
- Foley, van Dam, Feiner & Hughes. Computer Graphics.
- Watt. 3D Computer Graphics.
- Strothotte & Schlechtweg. Non-photorealistic Computer Graphics: Modeling, Rendering and Animation.
- Lorensen & Cline (1987) *Marching cubes: a high resolution 3D surface construction algorithm.* Computer Graphics, **21** (4), 163-169.

Solid modelling

Solid modelling

Objects to be modelled are solid rather than surfaces.

- Often only objects are defined; eg. engineering parts

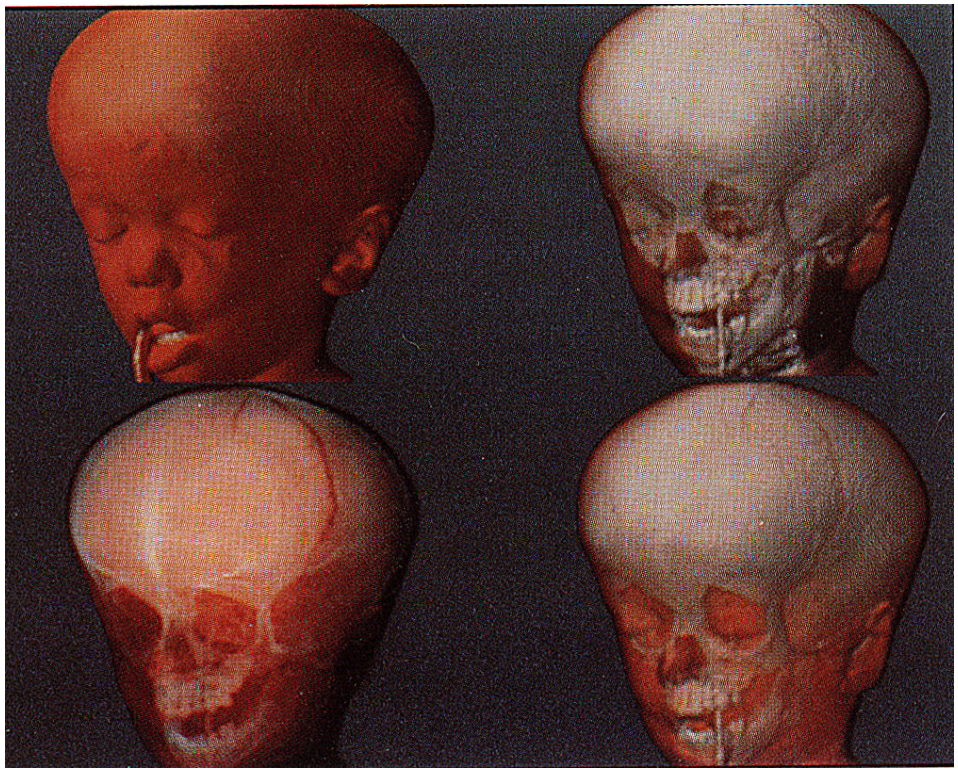


Gridded data

Data available for every point on a 3D grid

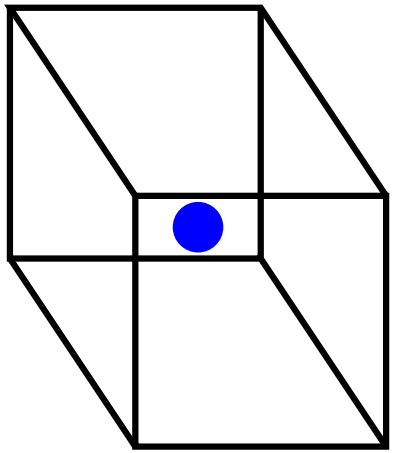
- Usually a regular, rectilinear grid
- Data often arises from physical measurements, eg CT scanning, MRI scanning.

Applications: voxel data



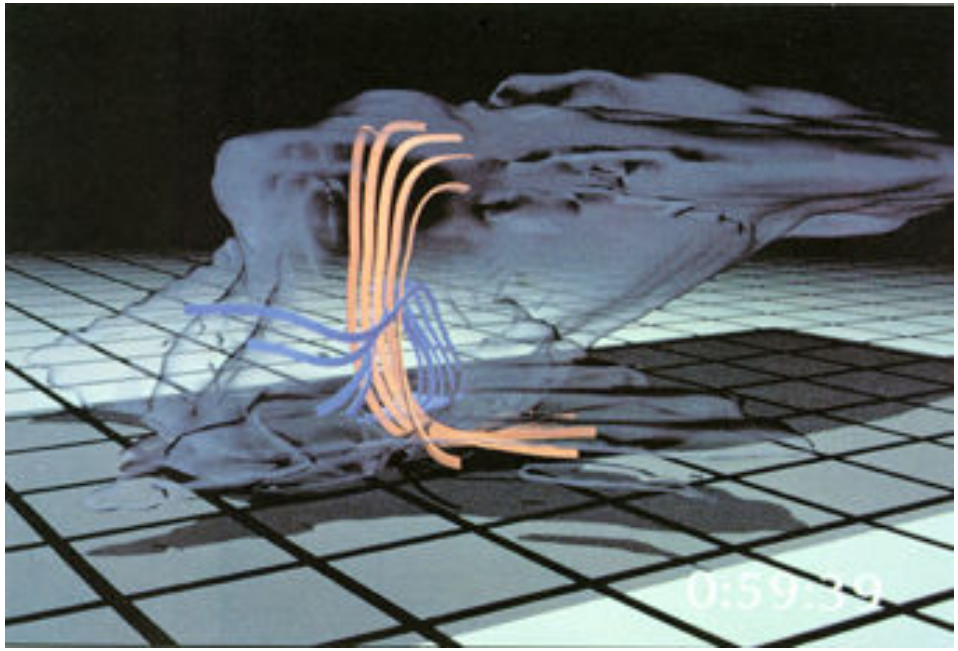
Scanning produces a value for each voxel dependent on the material characteristics. Preprocessing permits different structures to be studied.

Voxel = Volume element

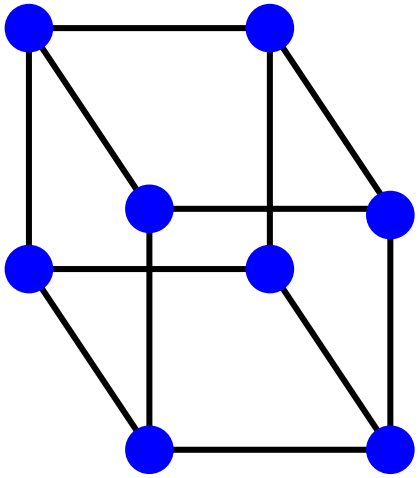


Value for each voxel represents an average value for the volume occupied by the voxel.

Applications: Cell data



Computational modelling produces a point value at the vertices of a regular grid.



- Modelling methods are often voxel-based
- Differences between voxel and cell data often ignored

Visualising 3D data

Surface rendering

Volume rendering

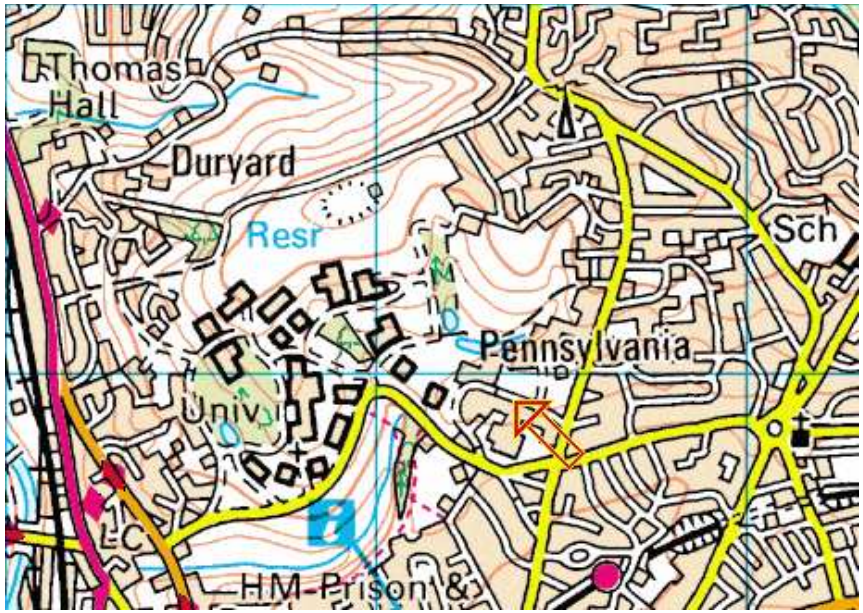
- Iso-surfaces: surfaces on which data value is constant
- Indirect: yields graphical primitives (polygonal mesh) which is then rendered

- Voxels endowed with colour and opacity
- Render the translucent data cuboid by ray casting
- Direct visualisation

Contouring

Two dimensions

- Construct lines on which a scalar is constant
  - Contours on OS maps: iso-height
  - Constant temperature lines: isotherms
  - Constant pressure: isobars
- Contours are the boundaries between different scalar values



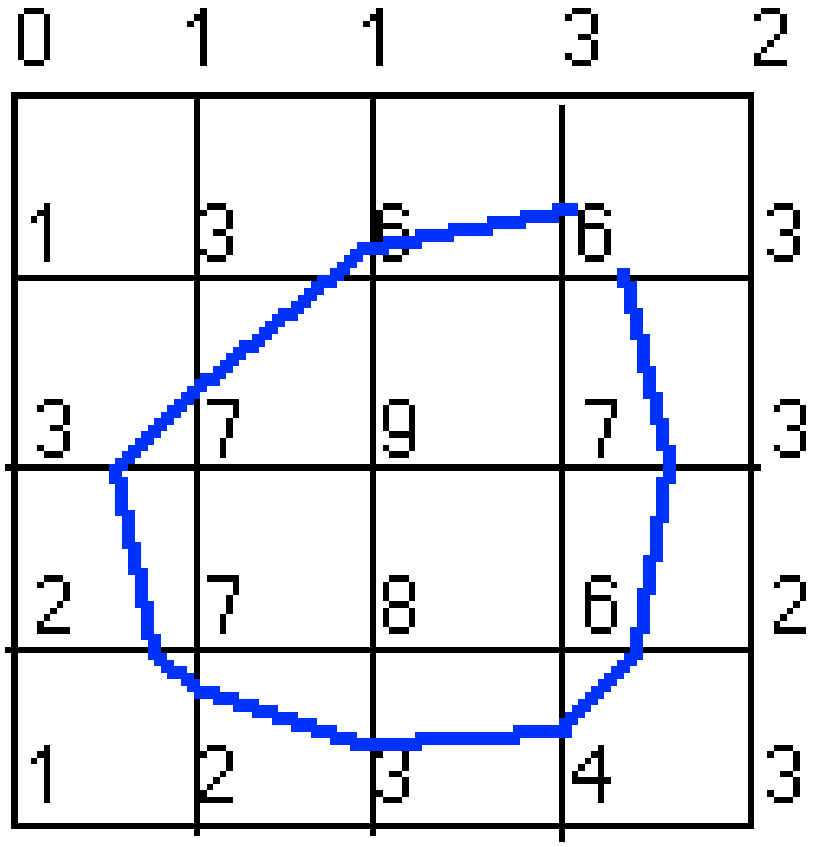
Three dimensions

- Construct iso-surfaces on which a scalar is constant
- Surfaces divide regions of differing scalar values

Contouring

- $f(x, y)$ is the value measured at location (x, y).
- Goal is to find the contour $f(x, y) = c$

1. Determine the location of intersection of contour with cell edges by *linear interpolation* between vertices.
   - Linear interpolation is simple, fast and usually sufficient.
2. Connect the intersections.
   - Connections with straight lines: simple and fast; splines etc may lead to crossing contours.



Contour tracking

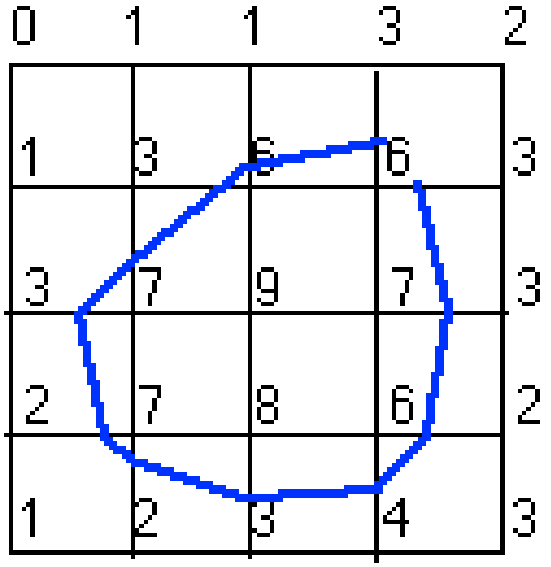
Follow a contour from start to finish

A contour that enters a cell on one edge must exit on another edge

1. Scan cells and edges to detect an edge intersection
2. while not finished
3. find the exit-edge in current cell
4. mark cell as processed
5. entry-edge := exit-edge

Terminate if

- Contour closes on itself
- Contour reaches a boundary edge



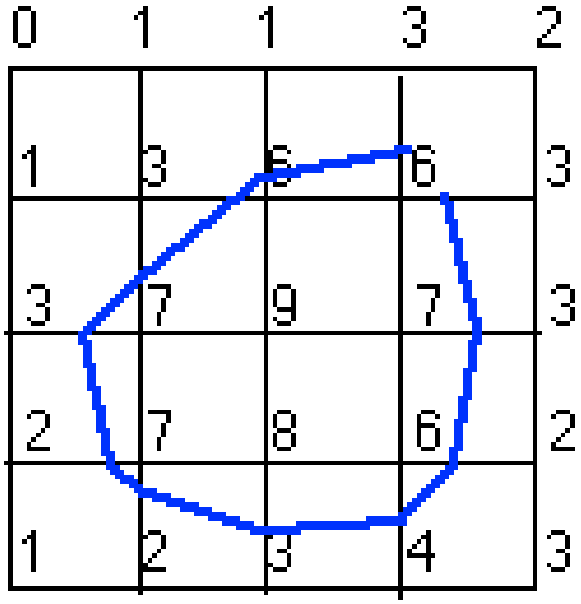
Yields contour as a single entity

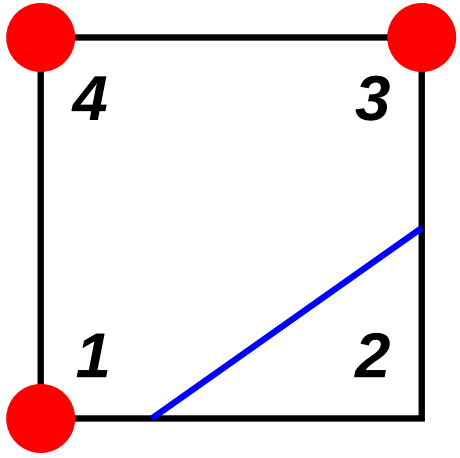
Helpful for labelling, measurement

Marching squares

- Contour tracking very difficult to extend to 3D
- Marching squares draws the contour in each cell in scanwise order
- Efficiency derived from rapid identification of the way in which contour passes through cell

- *Topological state* of cell depends on whether each of the 4 vertices is greater than or less than the contour.

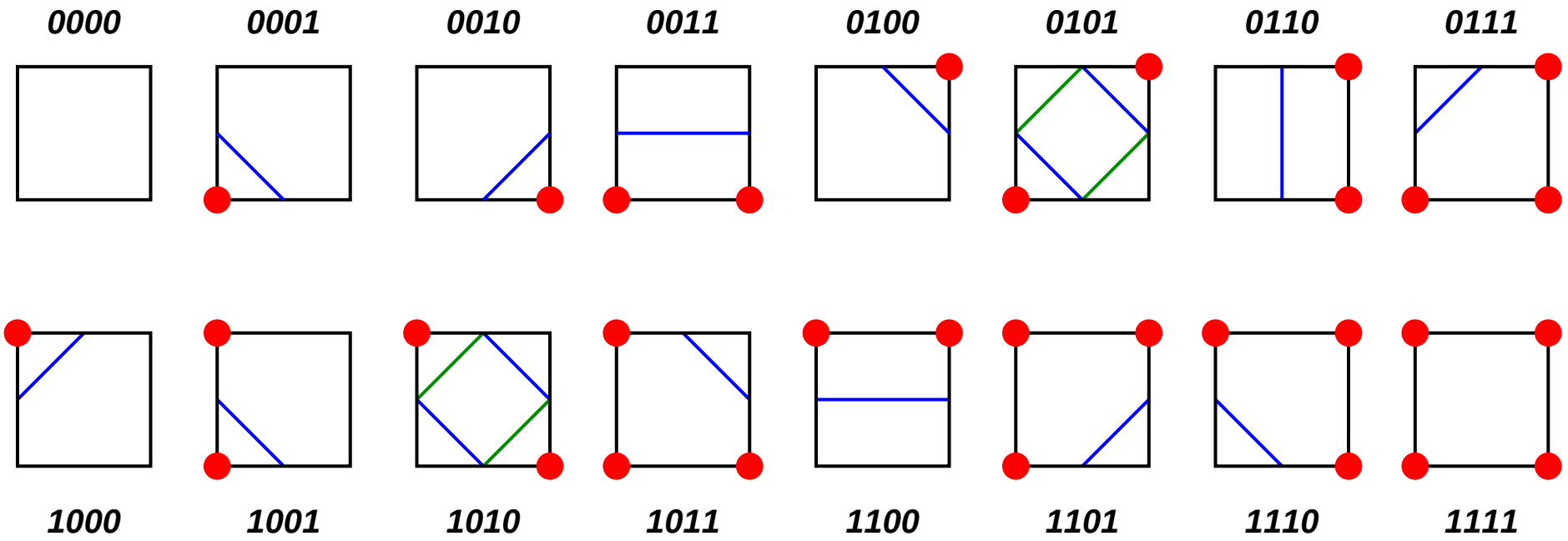




1101

Marching squares states

$2^4 = 16$ possible states



- denotes vertices greater than contour height.

Marching squares

Algorithm

1. for each cell:
2. Determine whether vertices are above or below c
3. Generate topological state index from bits
4. Locate intersection edges from state
5. Calculate intersection by linear interpolation
6. Draw contour segment

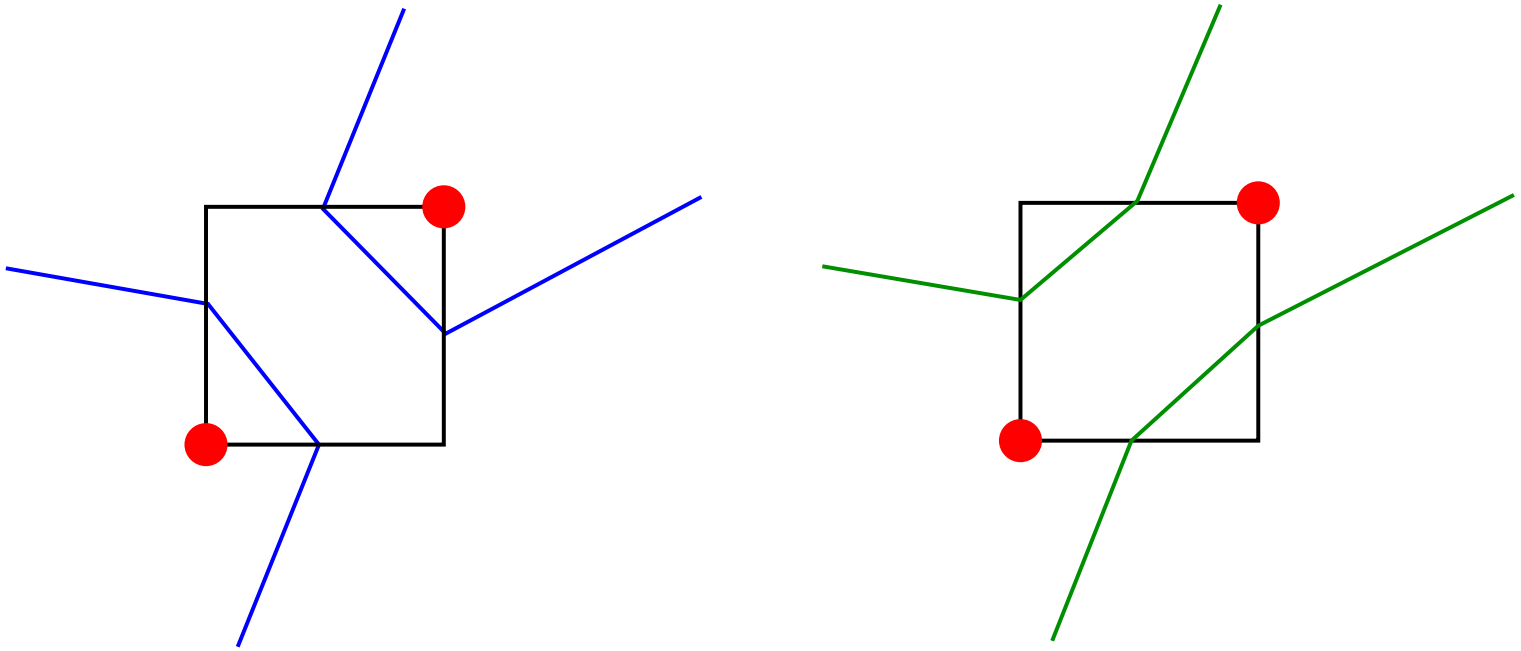
Notes

Contour produced in segments; can be time-consuming to re-assemble.

Interpolate in the same direction (eg low to high) to avoid round-off errors and ensure consistency of interpolated intersection.

Marching squares ambiguities

- States 0101 and 1010 are ambiguous: there are two possible contours at the saddle point.
- In two dimensions choose either interpretation.

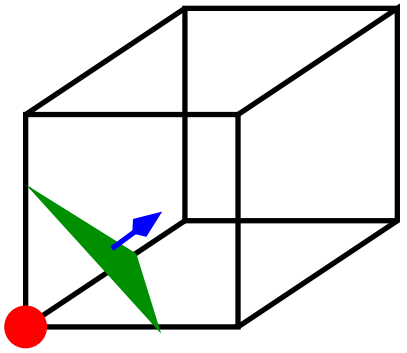


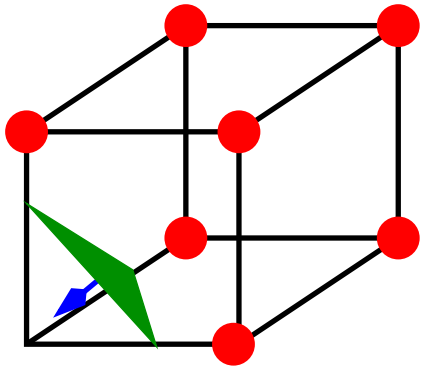
- Either interpretation is consistent with dataset.
- Rate of change of height can be used to infer the correct interpretation.

Marching cubes

Generalisation of marching squares to 3D to draw an isosurface $f(z, y, z) = c$

- 2^8 arrangements of vertices
- Inside/outside choice is arbitrary
- Other symmetries reduce the number of unique configurations to 15





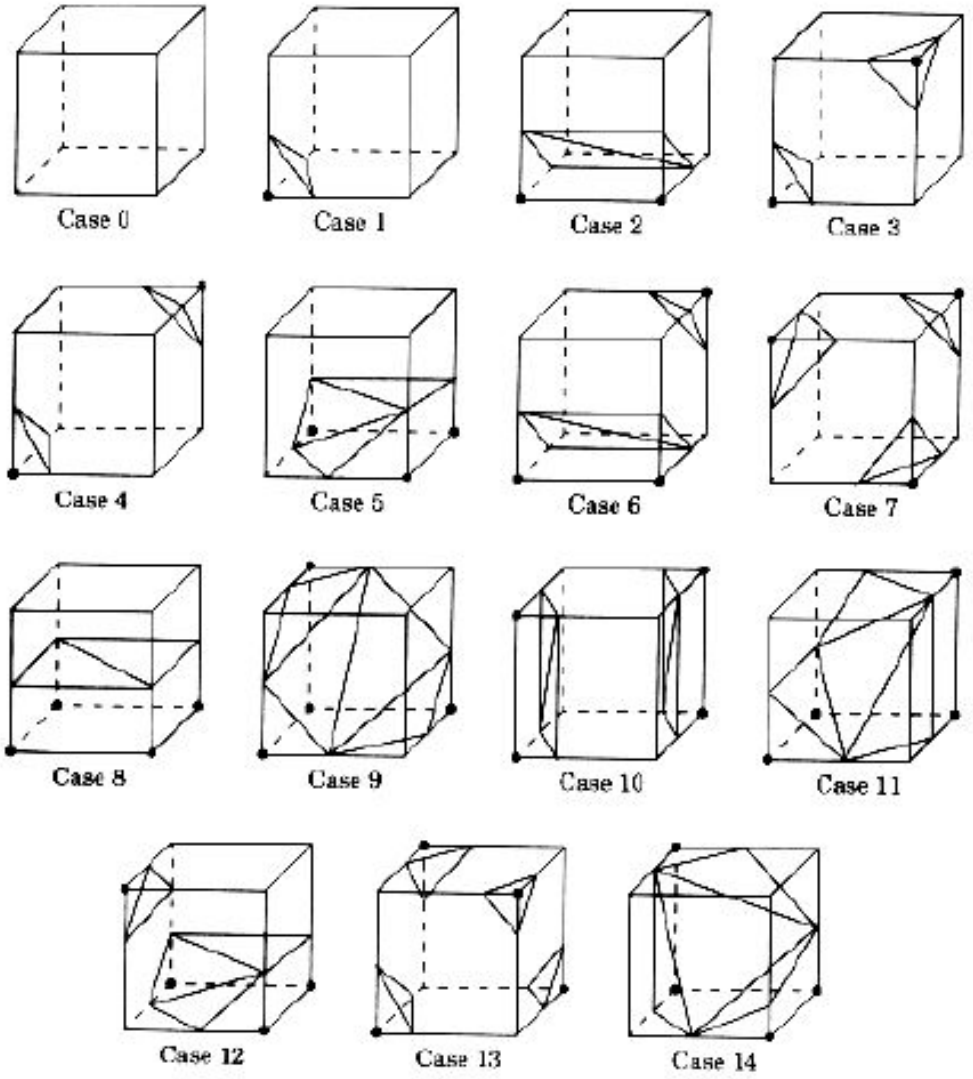
Marching cubes

For each cell:

1. determine state of cell from data values at vertices
2. find intersection of edges by inverse linear interpolation
3. draw triangular patches

Ambiguities

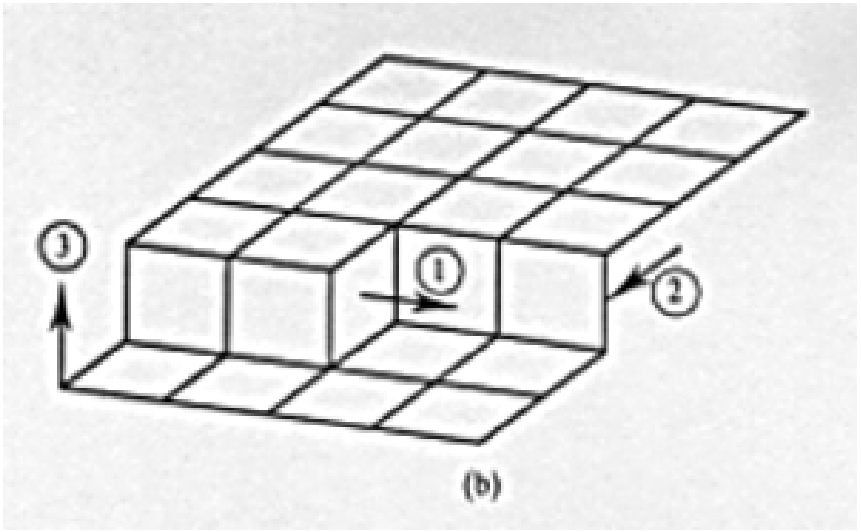
Resolved by reference to neighbouring cells to prevent holes in isosurface

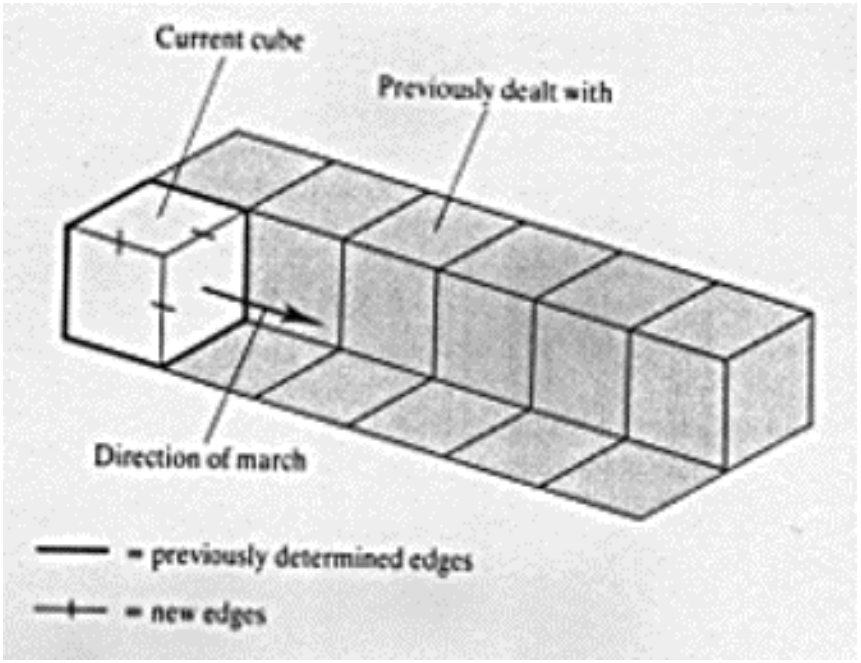


Efficiency

- Use *coherence of cells* to avoid expensive intersection calculations by
  - Storing edge intersections for reuse
  - Ordering the march to minimize necessary storage

  Buffers to retain intersections required until next slice is complete.

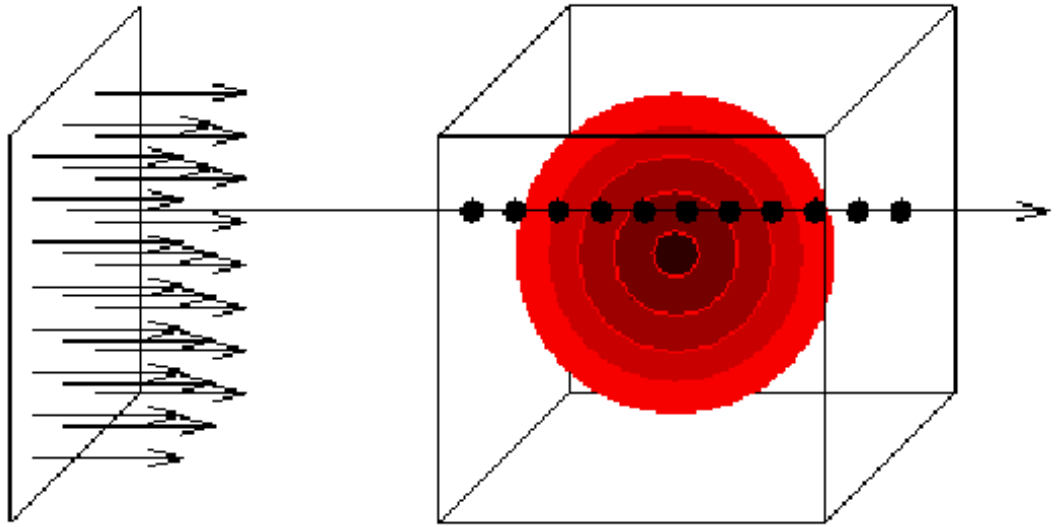




Limitations

- Visualisation uses intermediate geometrical primitives
- Poor resolution for small objects
- Unlike 2D data, outer iso-surfaces obscure inner ones: no information about the inside.

Volume rendering



- Endow each voxel with an *opacity* and *colour*
- View data cube by casting parallel rays from the image plane

References

- Examples: `http://www.fovia.com/gallery.php`
- Levoy's website: `http://graphics.stanford.edu/projects/volume`

Volume rendering

Classify each voxel into a particular material and thereby assign colour and opacity.
Assume that each voxel is composed of a single material

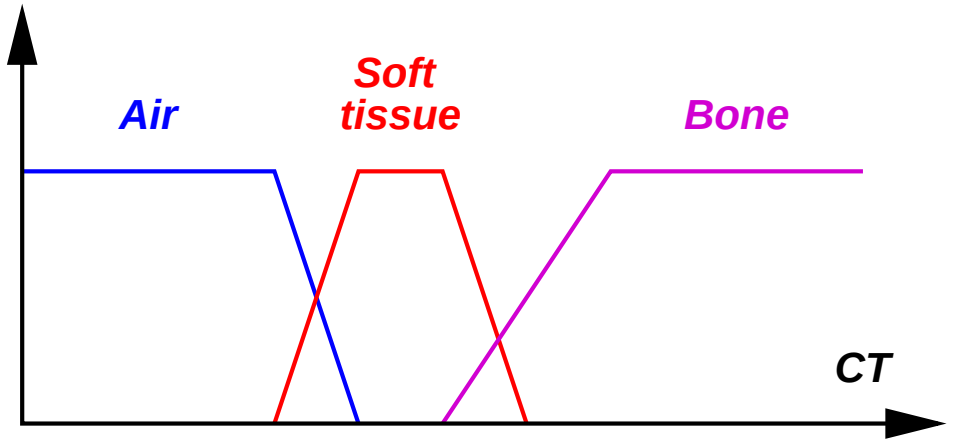
Transform classified volume data into the viewing direction

Cast rays from each pixel in the image plane find the overall pixel colour and intensity

Voxel classification: transfer function

Transfer function maps the data value (e.g., X-ray absorption coefficient) to RGBA.

- Frequently determined by mapping data value to a material and then material to RGBA.



- Choice of material to RGBA mapping defined by the user.

Transfer function

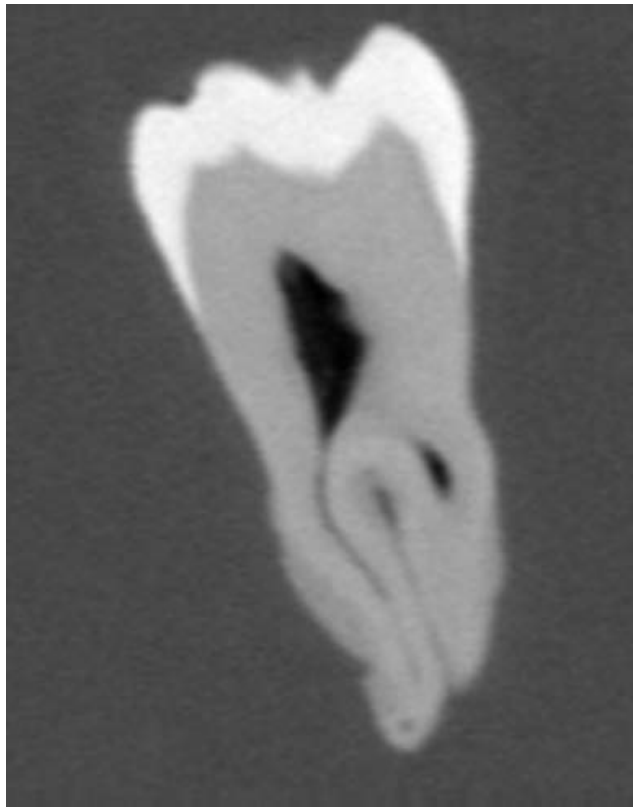
- Transfer functions often chosen interactively by specifying a colour for a region/organ of interest, but difficult unintuitive and slow.

Low data $\mapsto$ high α

High data $\mapsto$ high α

- Semi-automatic methods to incorporate spatial information, such as edges or the curvature of isosurfaces.
  `http://www.cs.utah.edu/~gk/papers/vis03/`

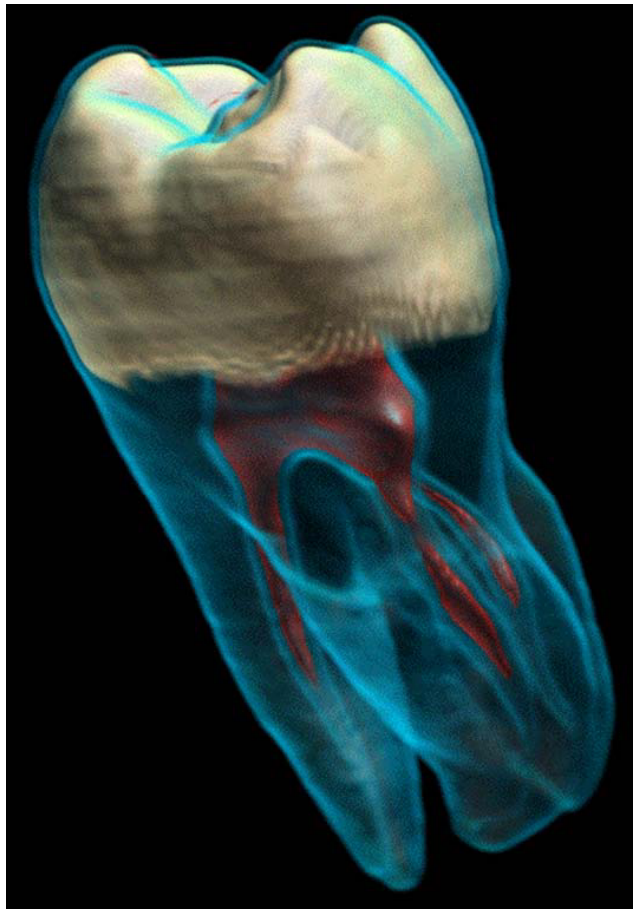
Transfer function



CT adsorption, f

$RGB(f)$

$\alpha(f)$



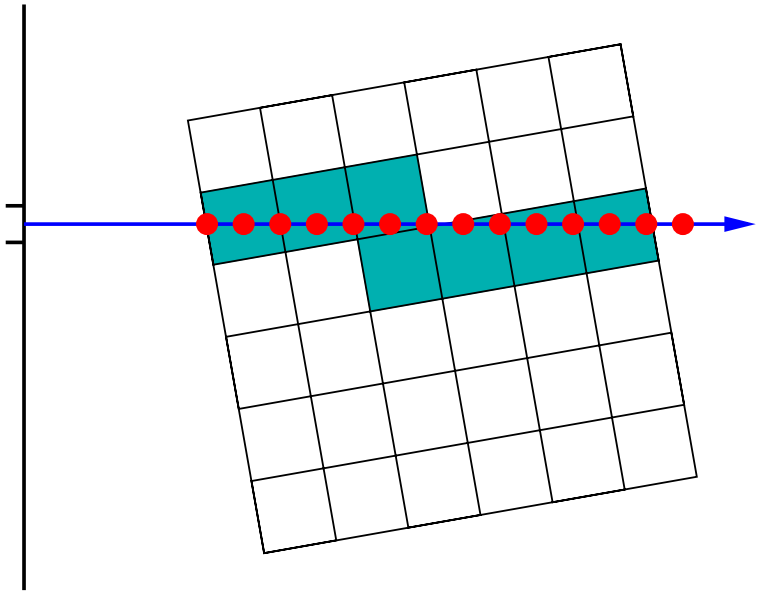
Composited tooth

Transformation into viewing direction

Either

Sample the data at equal intervals along the untransformed data

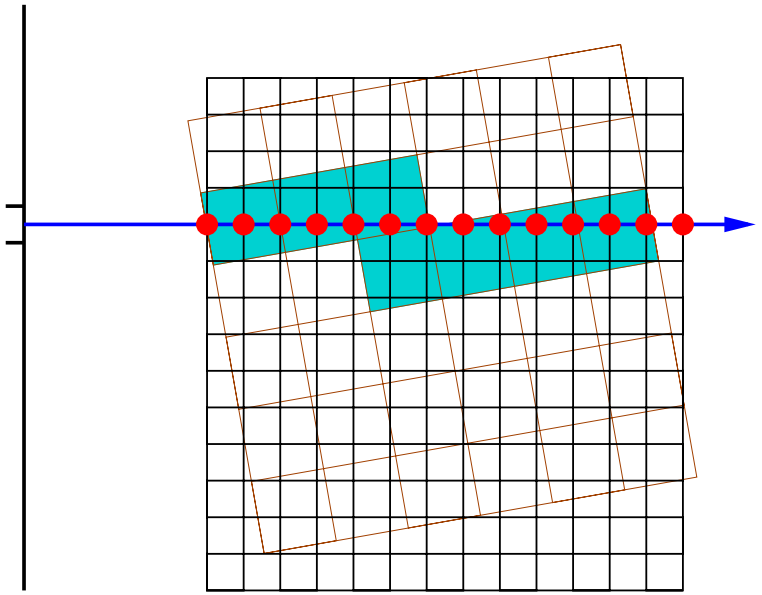
Linear interpolation to find the value at each sample point.



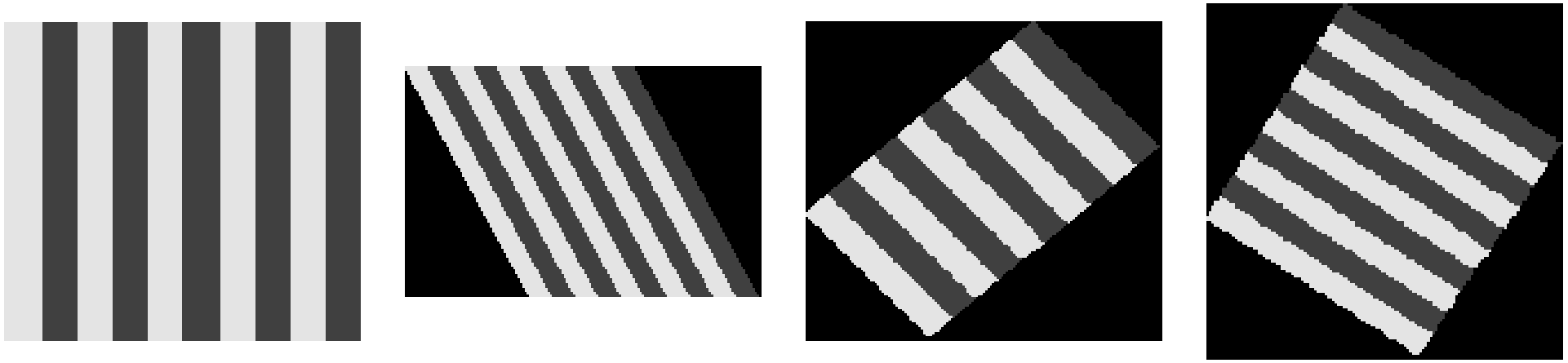
Or

Pre-transform the data so that ray is parallel with transformed pixels.

Rotation about each axis efficiently accomplished by 3 shears in hardware.



Three shear rotation



$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix}$$

- Each transformation is just a shift along rows or columns.
- Filtering to reduce aliasing can be applied at each stage.

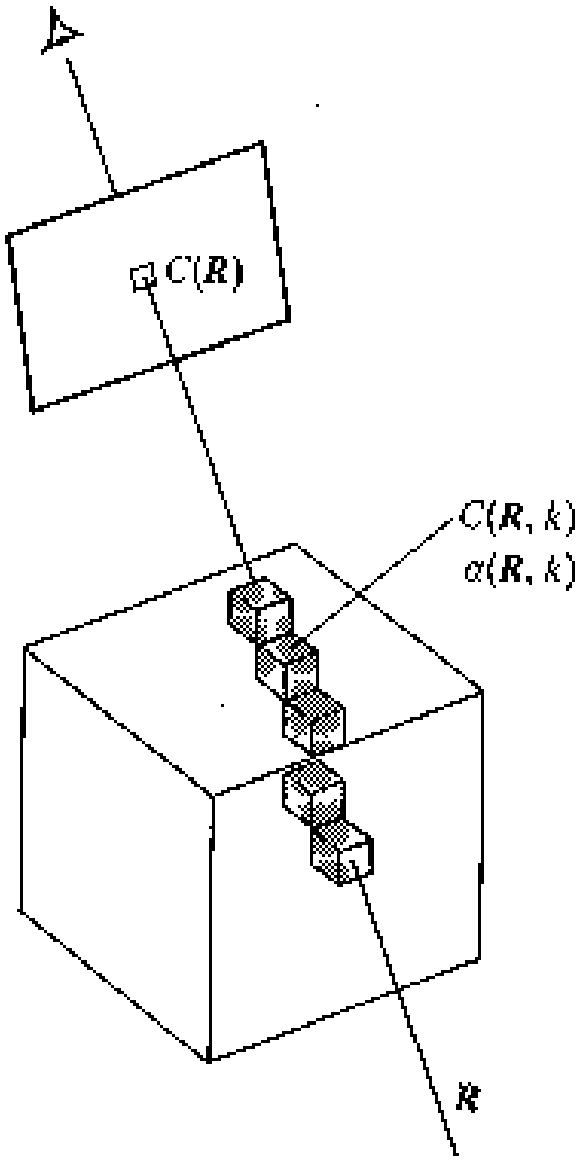
Compositing

- Ray intensity is modified by each voxel/sample it passes through on the way to the eye:

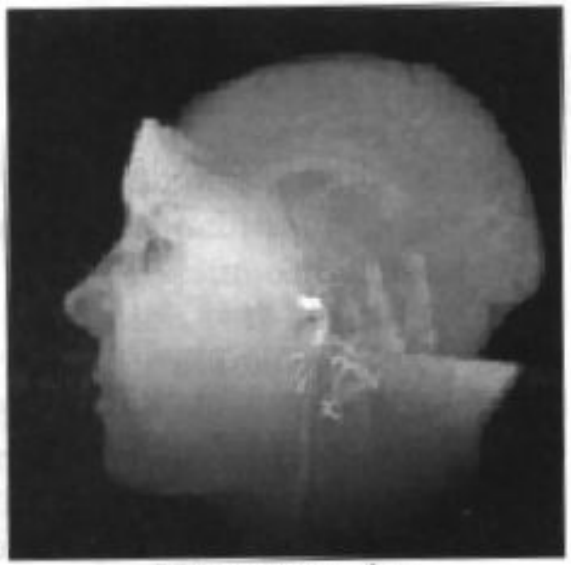
$$C_{out} = C_{in}(1 - \alpha) + \alpha C$$

C_{in}	R/G/B for incoming ray
C_{out}	R/G/B for outgoing ray
C	R/G/B for this sample/voxel
α	opacity for this sample/voxel

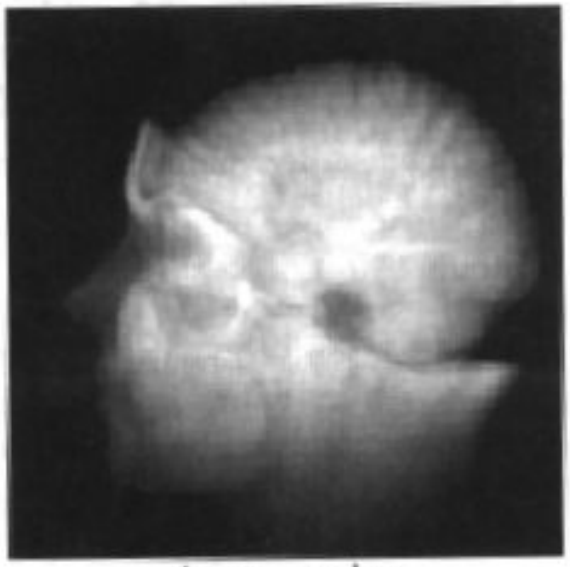
- High α voxels are visible: they obscure voxels behind them and low-α voxels in front are relatively transparent.



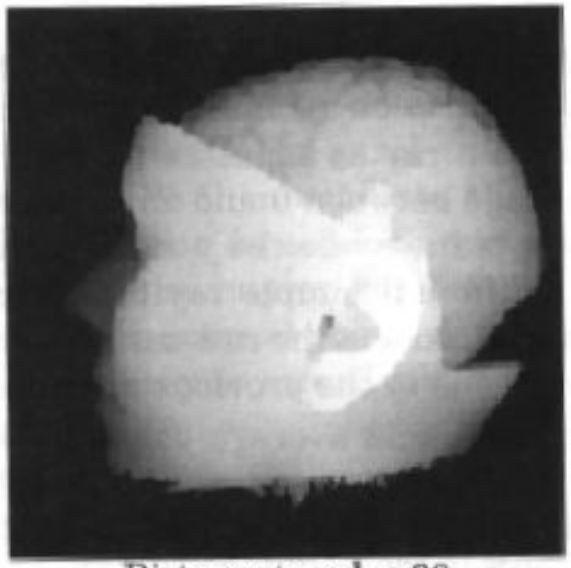
Alternative ray functions



Maximum value



Average value

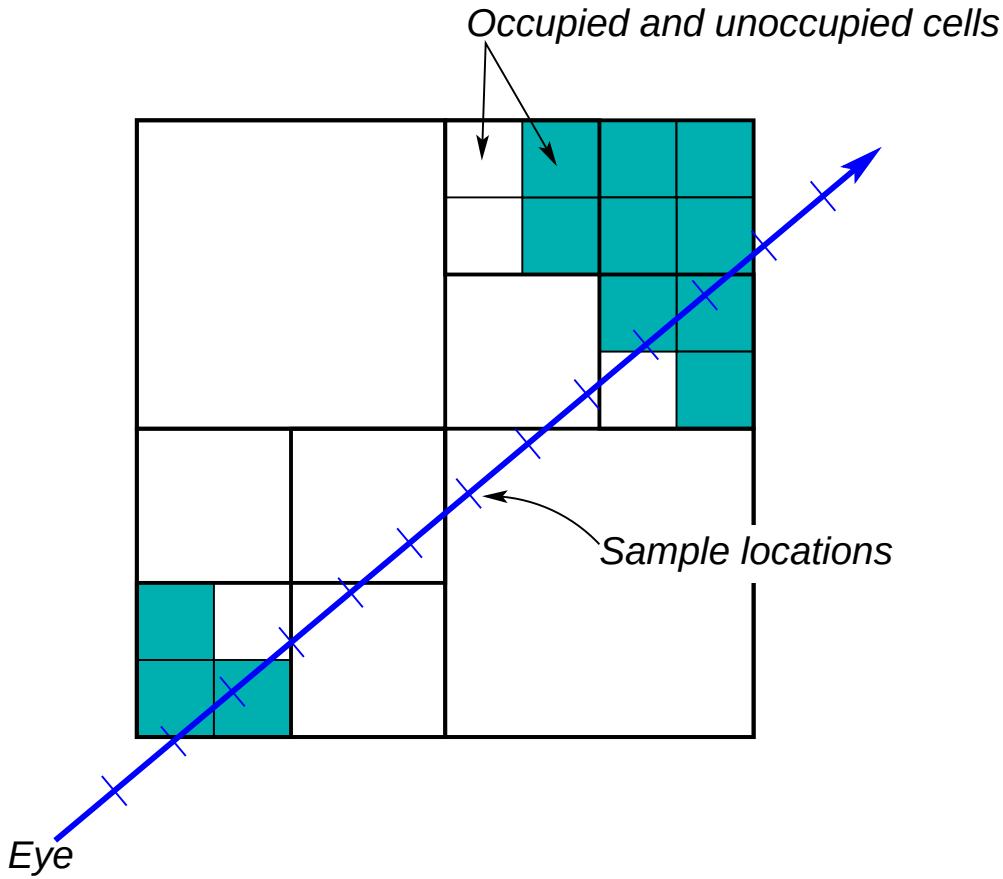


Distance to value 30

Composite

Efficiency

- Build pyramidal data structure (eg. octree) with 1s at high levels indicating non-zero α in lower levels. Only visit voxels with non-zero α.
- Early ray termination: when compositing from front to back, stop if the ray is too dark to be visible.
- Hardware: utilisation of GPU hardware permits rapid volume visualisation.

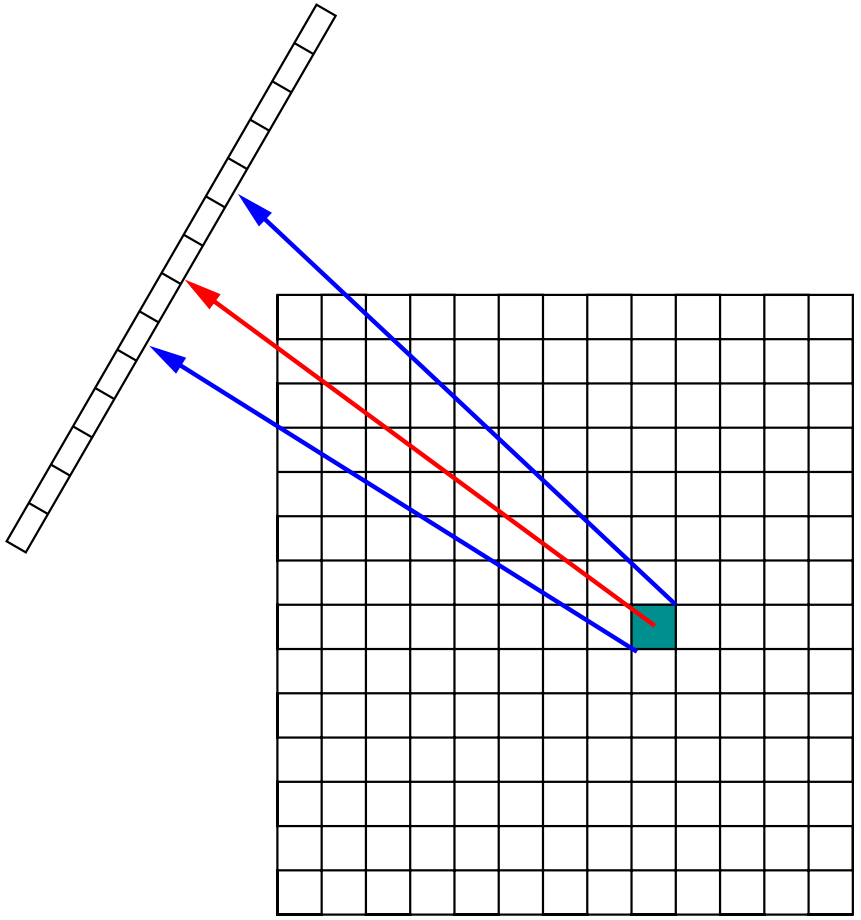


After: 'Efficient Ray Tracing of Volume Data', Marc Levoy, ACM Transactions on Graphics, 1990.

Splatting

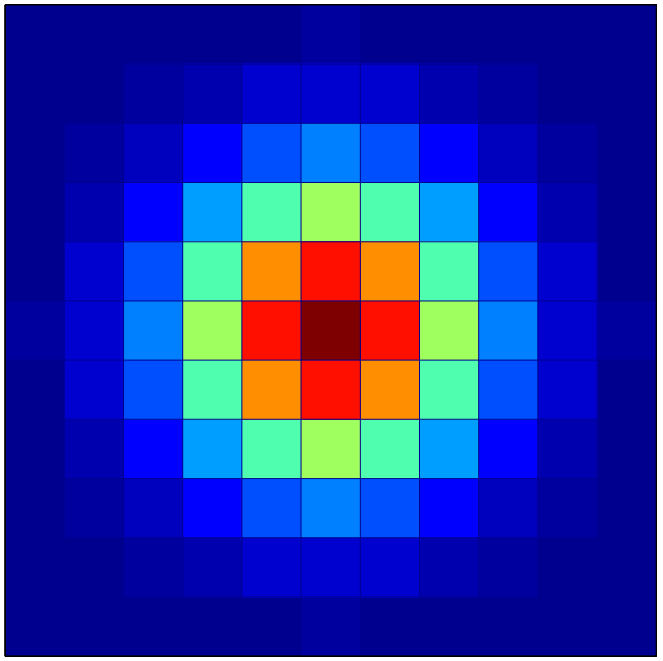
- Project voxels forward from data cube to the image plane.
- Accumulate total pixel intensities by working from front to back of data cube.

- Each voxel projects to more than one pixel.
- Kernel or spreading function determines the contribution of each voxel to the central pixel and neighbours.



Splatting

- Kernel function often a discretised Gaussian function



- Splatting operations very fast
- Advantageous to rotate (shear-warp) before splatting.
- Splatting can be performed in parallel and can take advantage of GPUs; memory access is the key to a fast algorithm.
- Care to avoid aliasing required when more than one voxel maps into a pixel
- *L. Westover 'Footprint evaluation for volume rendering', Computer Graphics,* **24**, *1990*

Volume Graphics

Advantages

- Insensitive to scene complexity
  Objects are preconverted into voxel form.
- Insensitive to object complexity
- Complexity is viewpoint independent
- Interior information is available

Disadvantages

- Memory requirements: $512^2 \times 2$ bytes per voxel $=$ 256 Mbytes
- Processing power
- Discrete form: resolution is limited by voxel resolution
- Geometric form is lost on conversion to voxel representation