3D Scalar Field Visualization: Volume Rendering

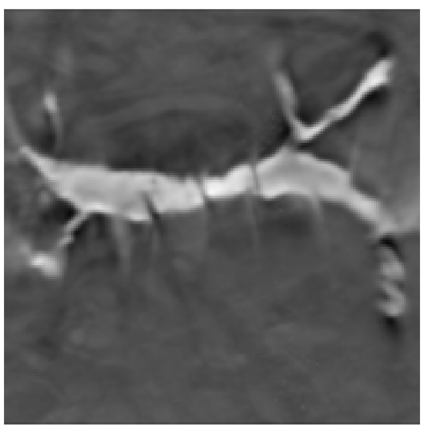
Goal: understand what is DVR and why it is useful; how to compute DVR (important steps); how to perform raycasting

Iso-surfacing Could Be limited

- Iso-surfacing is "binary"
  - point inside iso-surface?
  - voxel contributes to image?
- Is a hard, distinct boundary necessarily appropriate for all the visualization tasks?

Iso-surfacing Could Be limited

- Iso-surfacing is "binary"
  - point inside iso-surface?
  - voxel contributes to image?
- Is a hard, distinct boundary necessarily appropriate for all the visualization tasks?



Slice

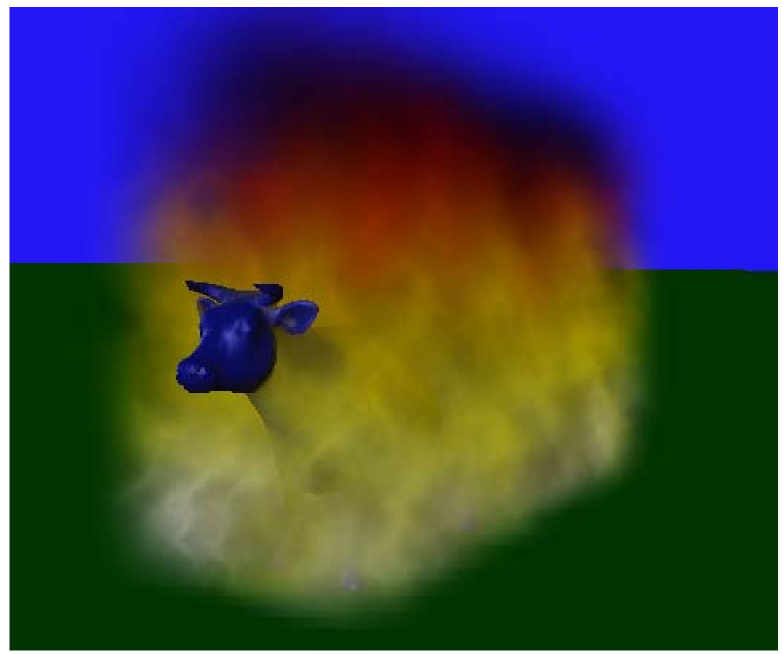
Isosurface

Volume Rendering

Iso-surfacing Could Be Limited

- Iso-surfacing poor for ...
  - measured, "real-world" (**noisy**) data
  - **Amorphous (fog-like)**, "soft" objects

virtual angiography



bovine combustion simulation

What is Direct Volume Rendering

- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry

- How do you make the data visible?

What is Direct Volume Rendering

- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry

- How do you make the data visible? : Via Color and Opacity

- How to achieve that?

What is Direct Volume Rendering

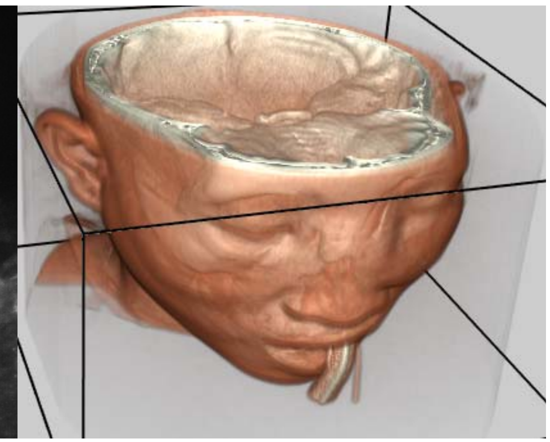
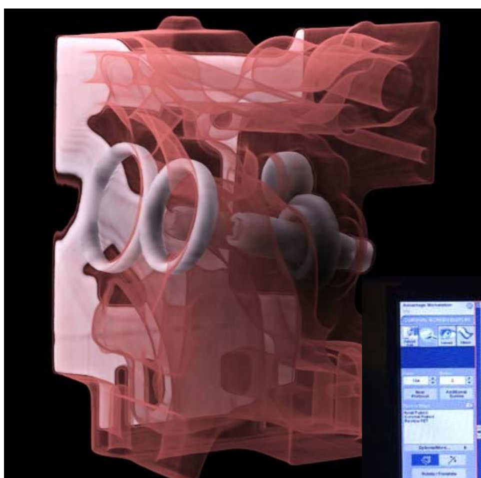
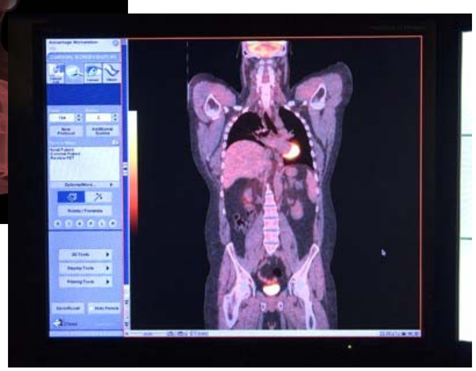
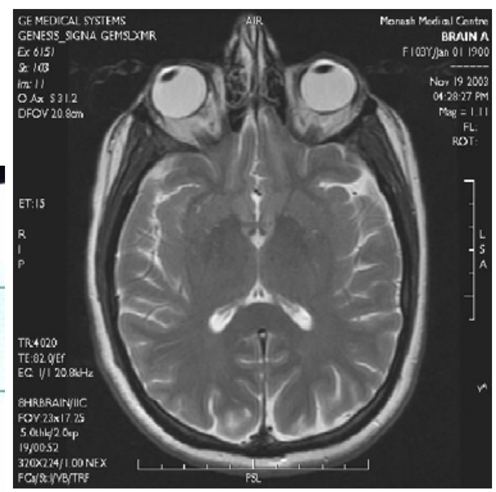
- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry

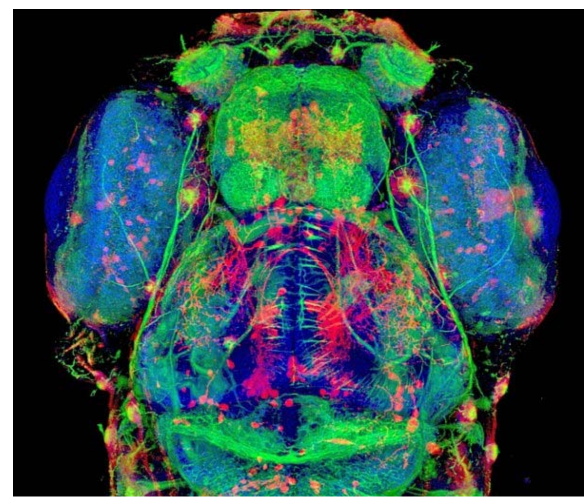
- How do you make the data visible? : Via Color and Opacity

- How to achieve that?
  - The data is considered to represent a **semi-transparent light-emitting medium**
  - Approaches are based on the **laws of physics** (emission, absorption, scattering)
  - The volume data is **used as a whole** (look inside, see interior structures). Think of color plots in 3D!

Volume Rendering is Useful

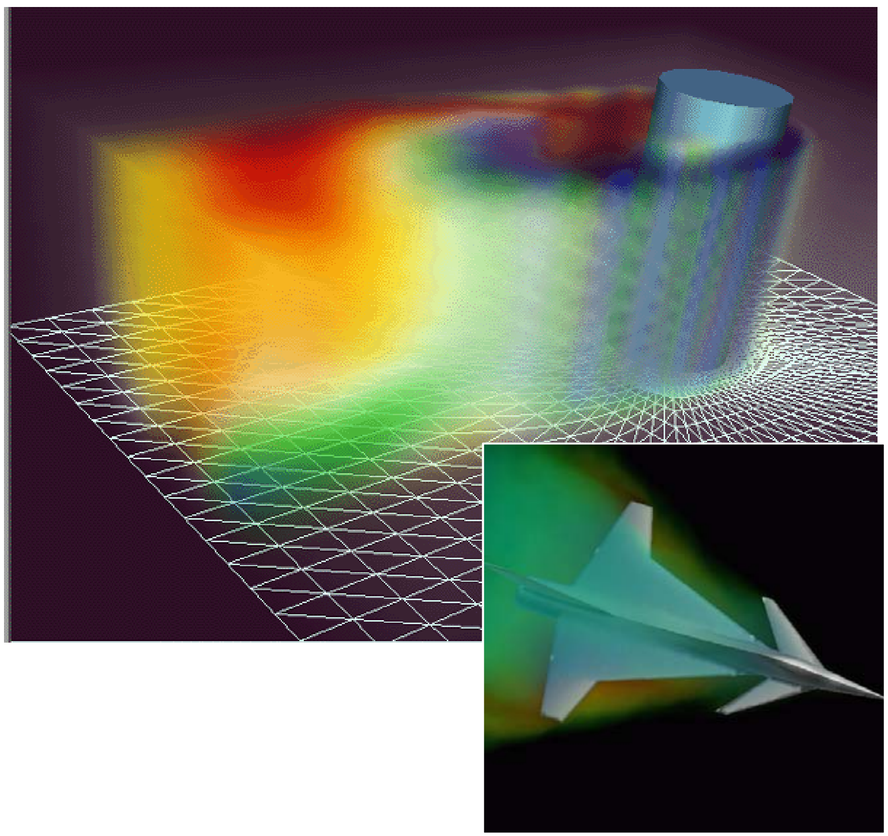
- **Measured** sources of volume data
  - CT (computed tomography)
  - PET (positron emission tomography)
  - MRI (magnetic resonance imaging)
  - Ultrasound
  - Confocal Microscopy

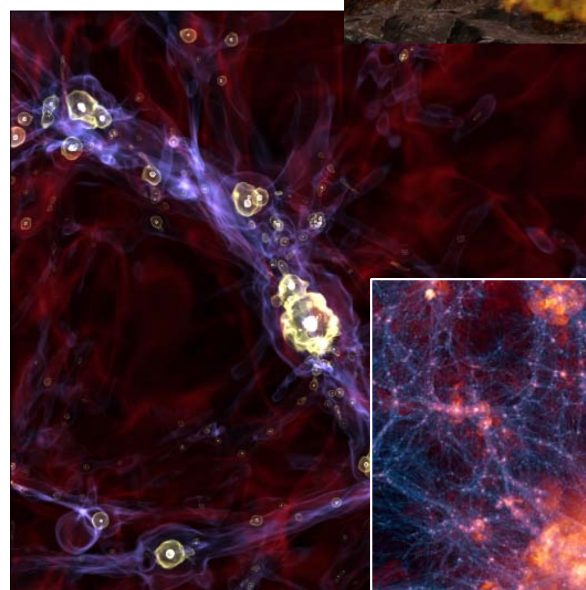






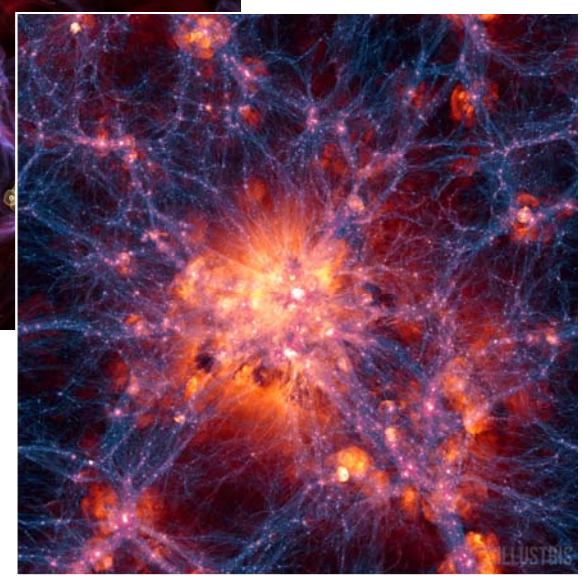


Volume Rendering is Useful

- **Synthetic** sources of volume data
  - CFD (computational fluid dynamics)
  - Voxelization of discrete geometry







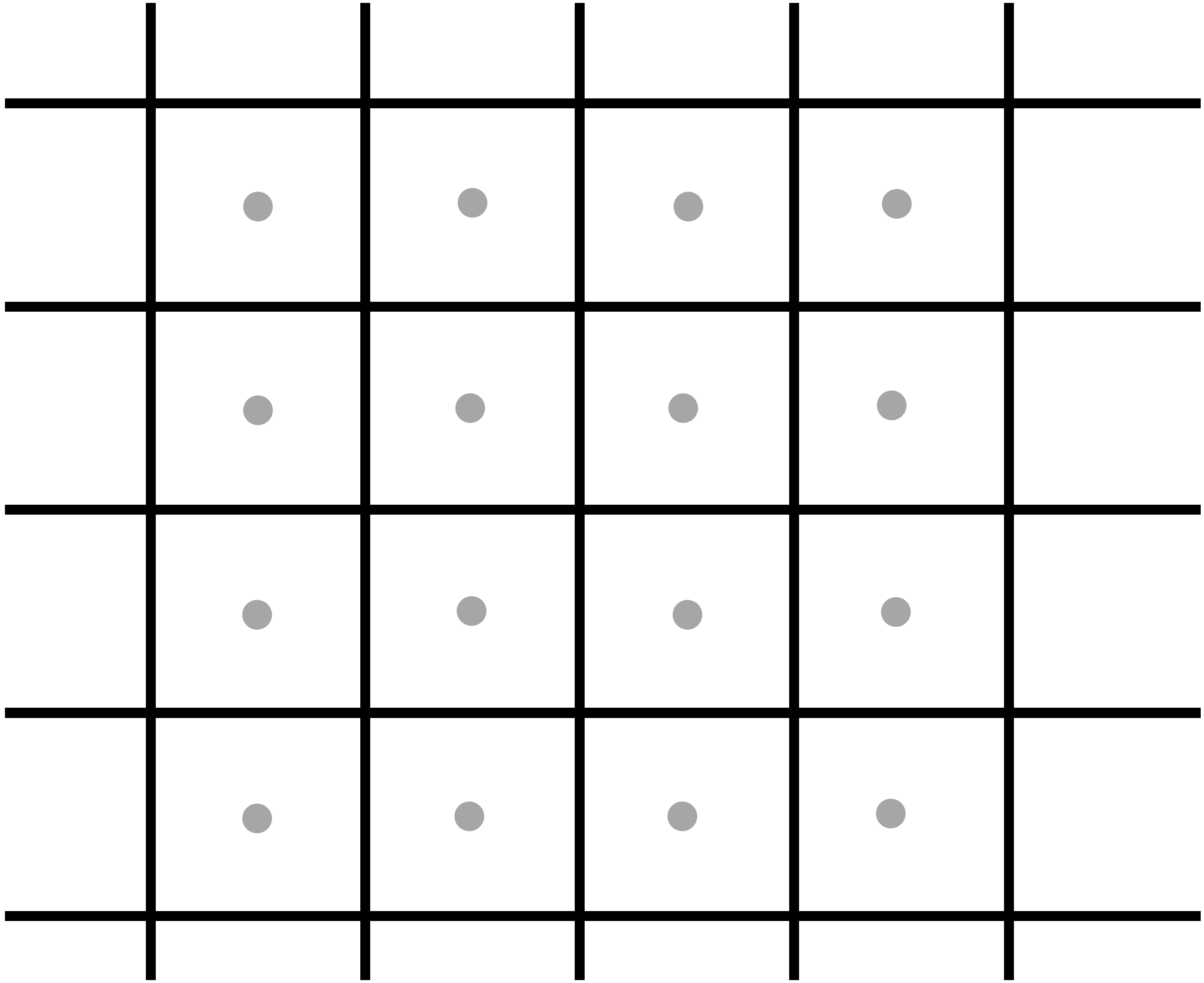
Data Representation

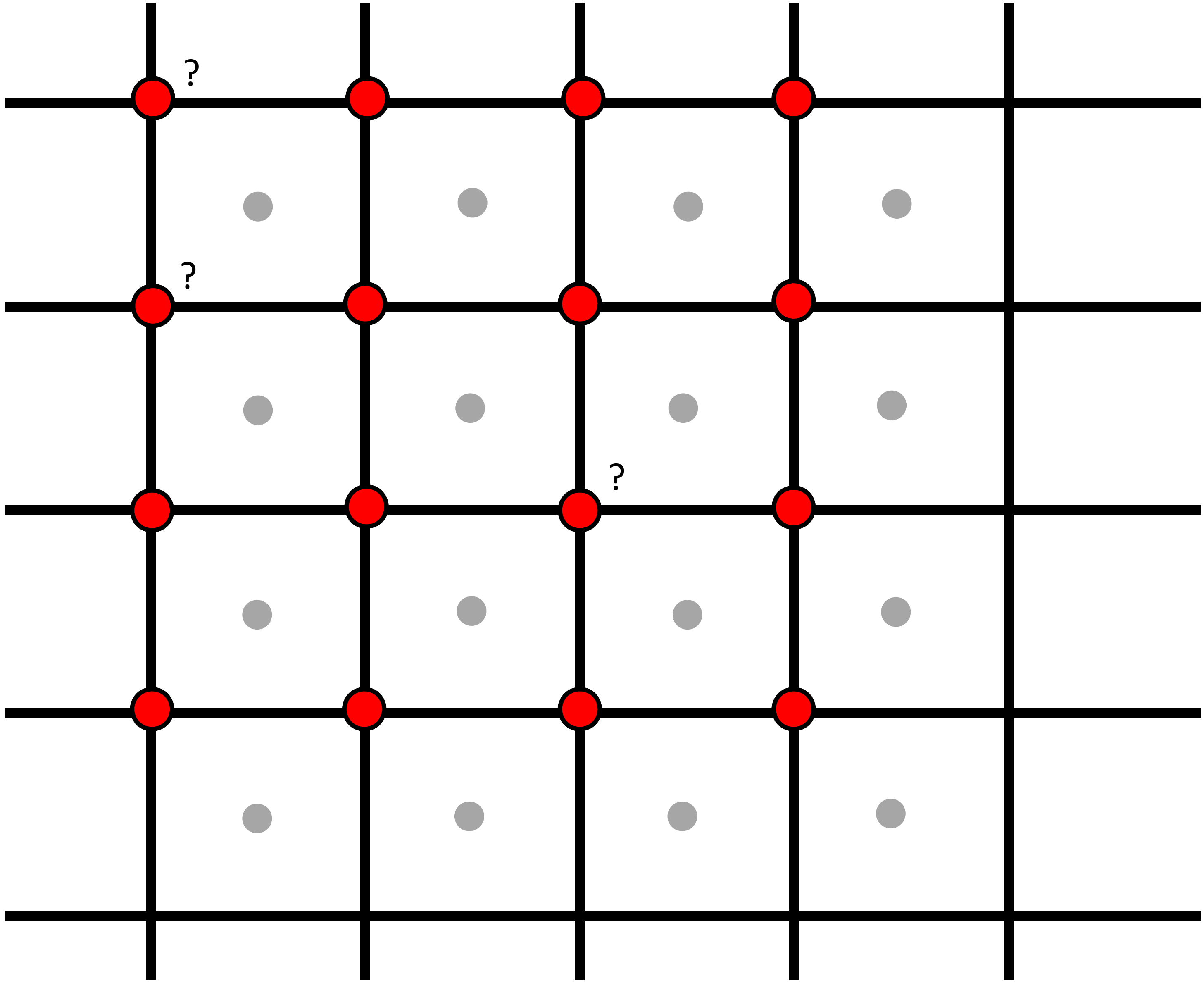
- Volume rendering techniques
  - depend strongly on the **grid type**
  - exist for both structured and unstructured grids
  - are predominantly applied to **uniform grids (3D images).**
  - for uniform grid, **voxels** are the basic unit

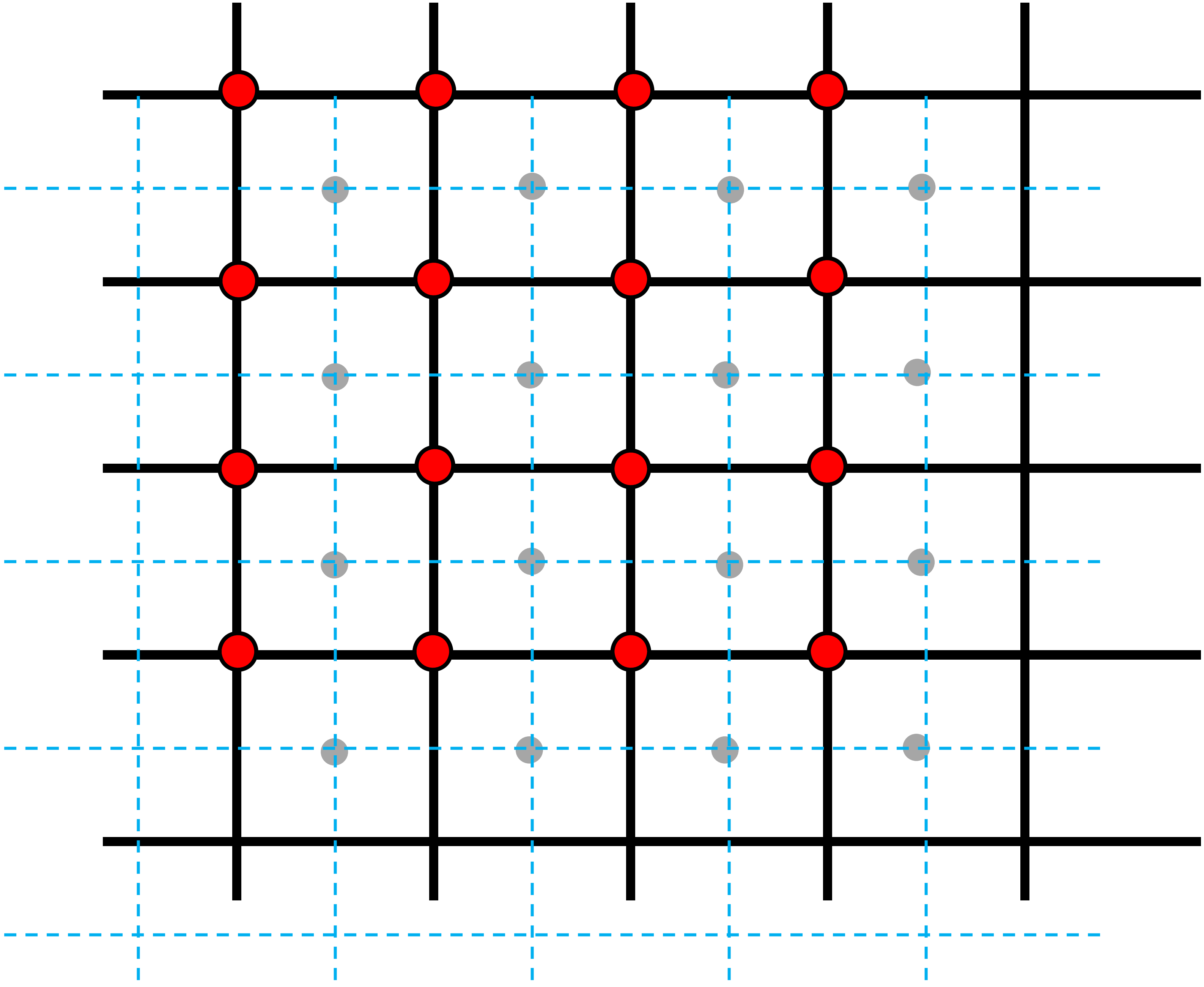
Data Representation

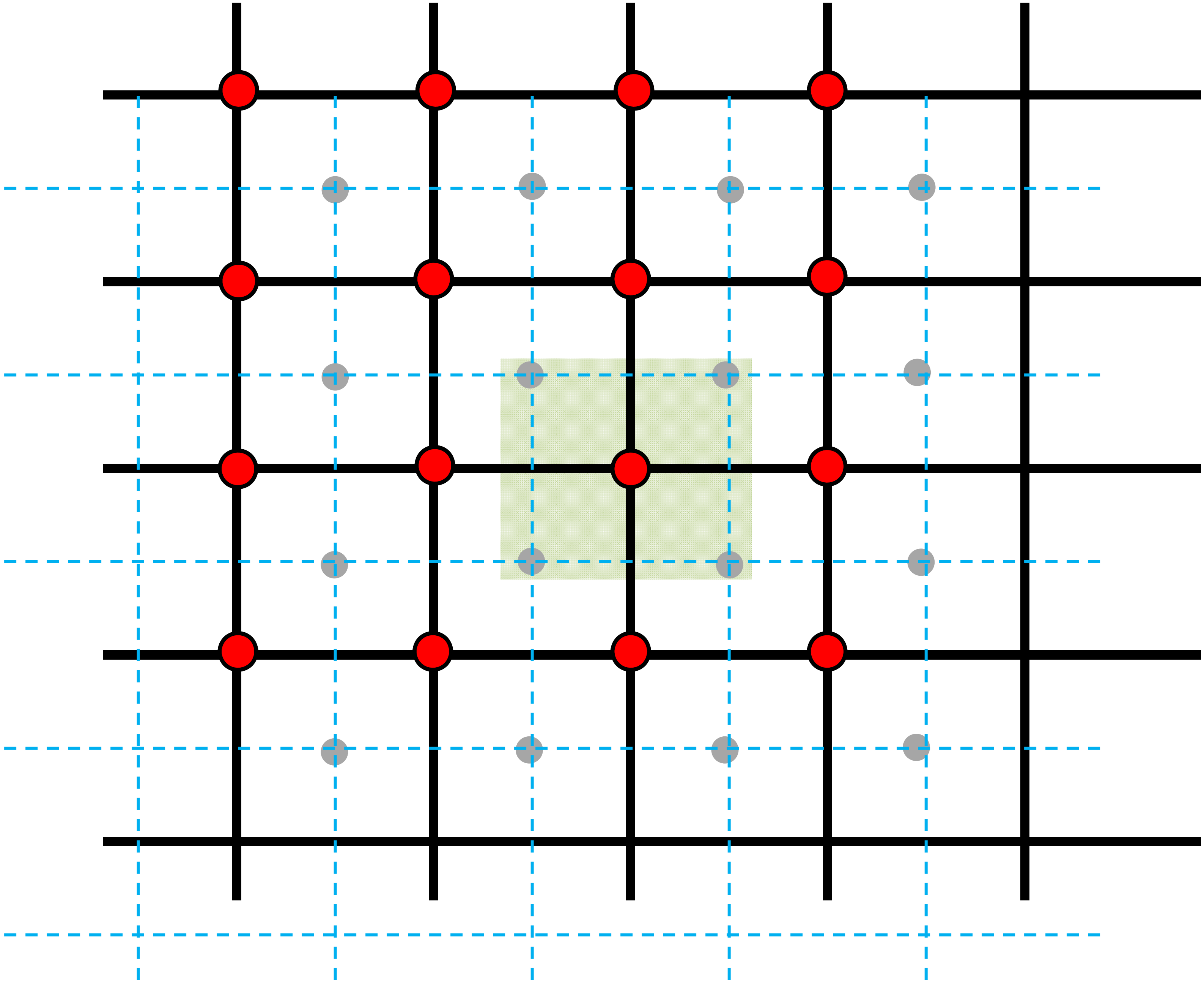
- Volume rendering techniques
  - depend strongly on the **grid type**
  - exist for both structured and unstructured grids
  - are predominantly applied to **uniform grids (3D images).**
  - for uniform grid, **voxels** are the basic unit

- **Cell-centered data for uniform grids**
  - are attributed to cells (pixels, voxels) **rather than nodes**
  - can also occur in (finite volume) CFD datasets
  - **are converted to node data (e.g., for iso-surfacing)**
    - by taking the dual grid (easy for uniform grids, n cells -> n-1 cells!)
    - or by interpolating.





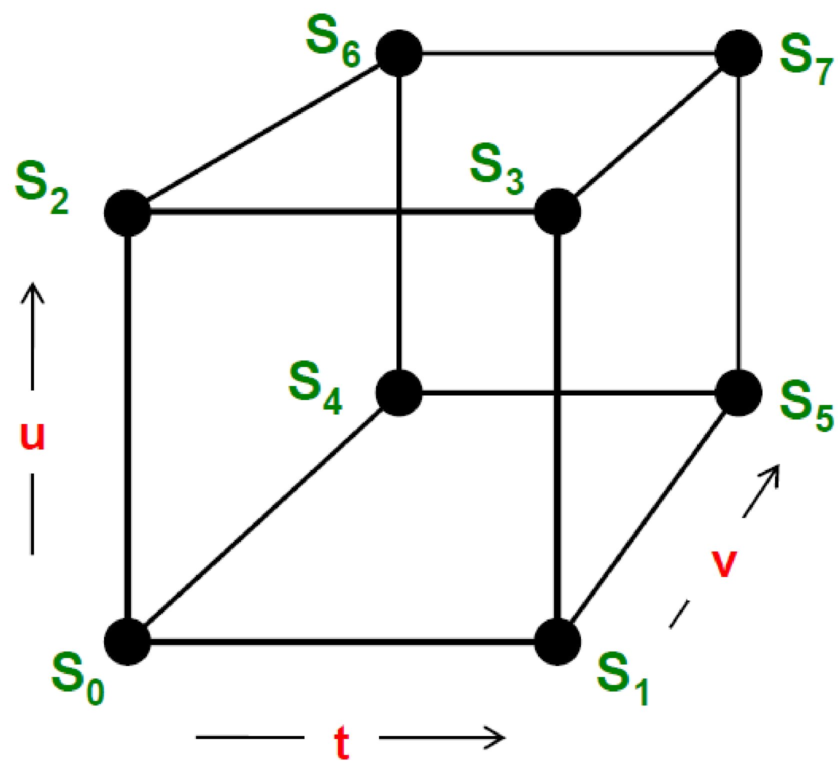




Important Concepts

- Interpolation
  - **trilinear** common, others possible
- Color and opacity transfer function
  - Turning scalar values to colors
- Gradient
  - direction of fastest change
- Compositing
  - "over operator"

Trilinear Interpolation



$$S(t,u,v) = (1-t)(1-u)(1-v)S_0 + t(1-u)(1-v)S_1 + (1-t)u(1-v)S_2 + tu(1-v)S_3 + (1-t)(1-u)vS_4 + t(1-u)vS_5 + (1-t)uvS_6 + tuvS_7$$

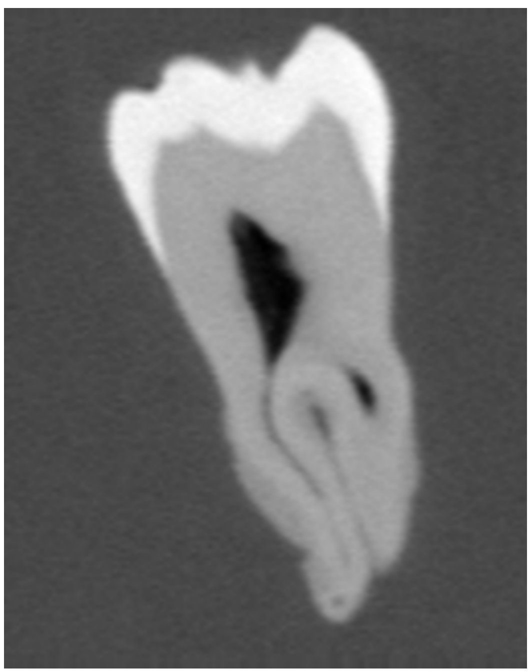
This is useful, for example, if we have passed an oblique cutting plane through a 3D mesh of points and are trying to interpolate scalar values from the 3D mesh to the 2D plane.

Color and <u>Opacity</u> Transfer Functions

- $C(f(p))$, $\boldsymbol{\alpha(f(p))}$ – p is a point in volume

- Functions of input data value $f(p)$
  - $C(f)$, $\alpha(f)$ – these are **1D functions**
  - Can include lighting effects
    - $C(f, N(p), \mathbf{L})$ where $N(p) = grad(f)$
  - Derivatives of f
    - $C(f, grad(f))$, $\alpha(f, grad(f))$

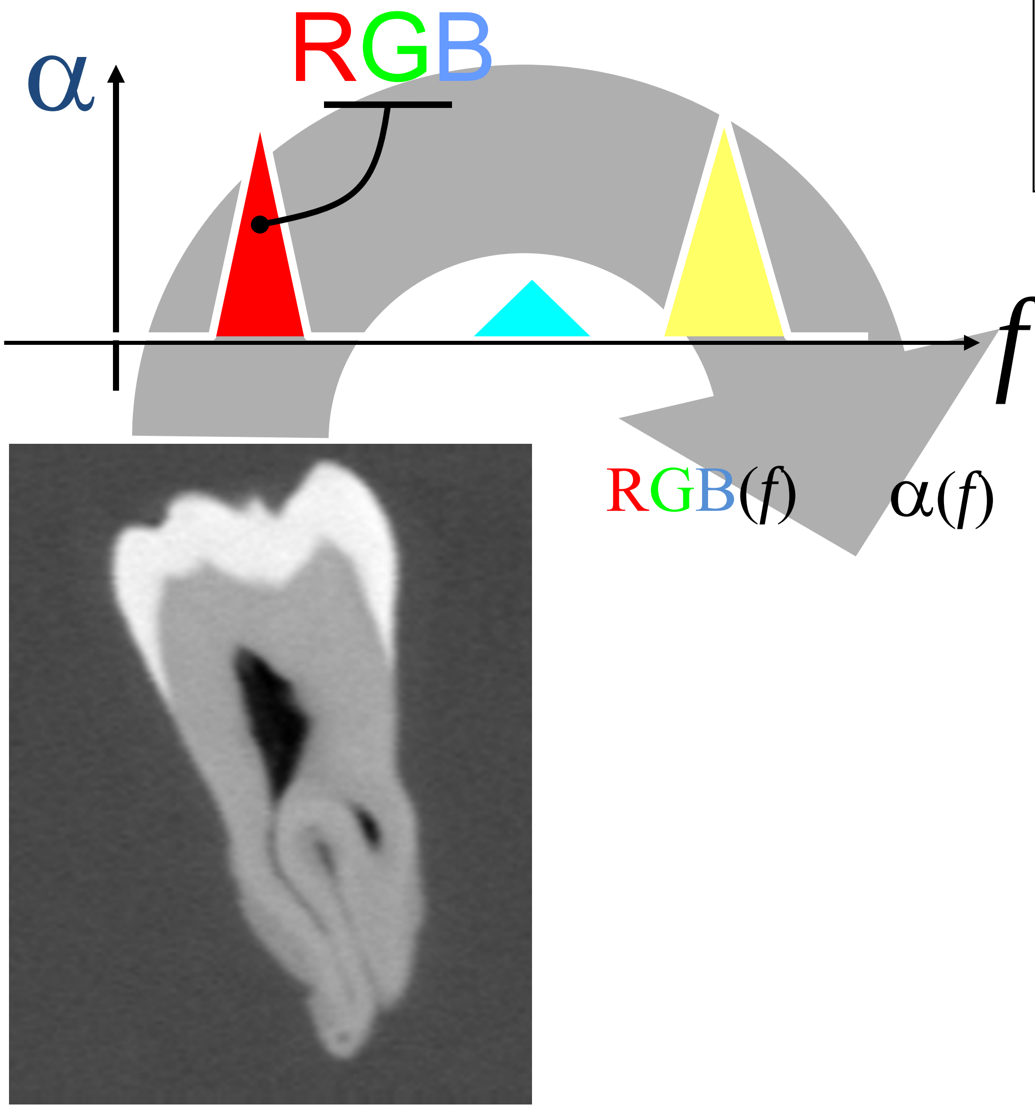
Transfer Functions (TFs)



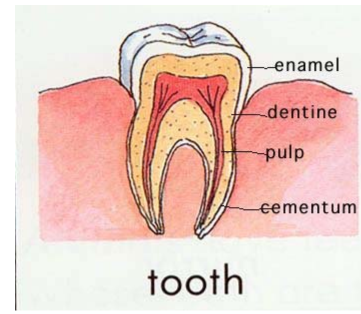
Human Tooth CT

Transfer Functions (TFs)

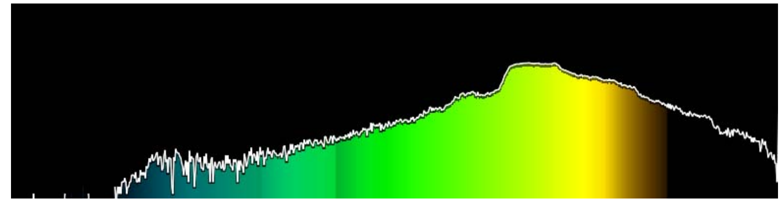
Map data value f to color and opacity



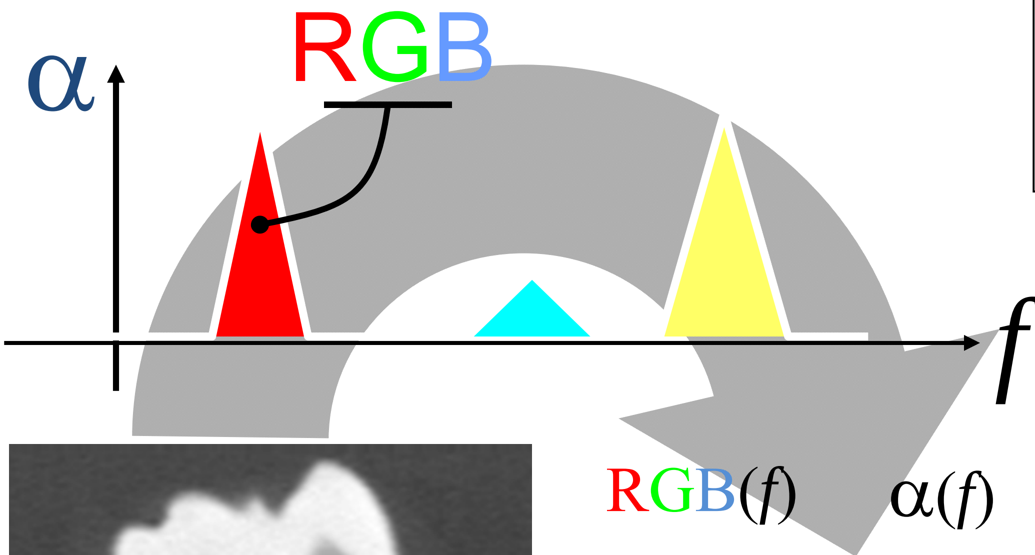
Human Tooth CT

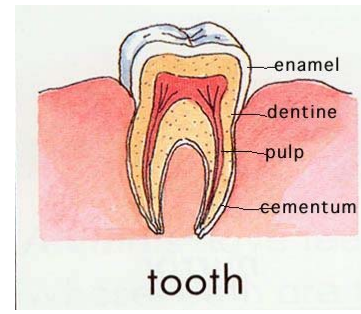


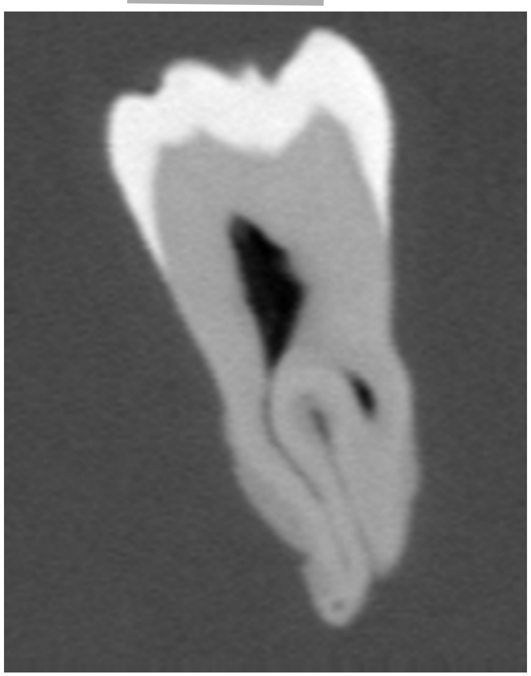
Transfer Functions (TFs)



Map data value f to color and opacity

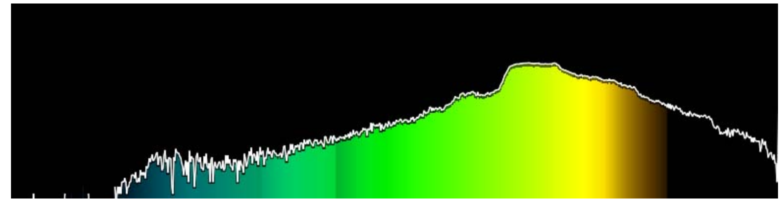




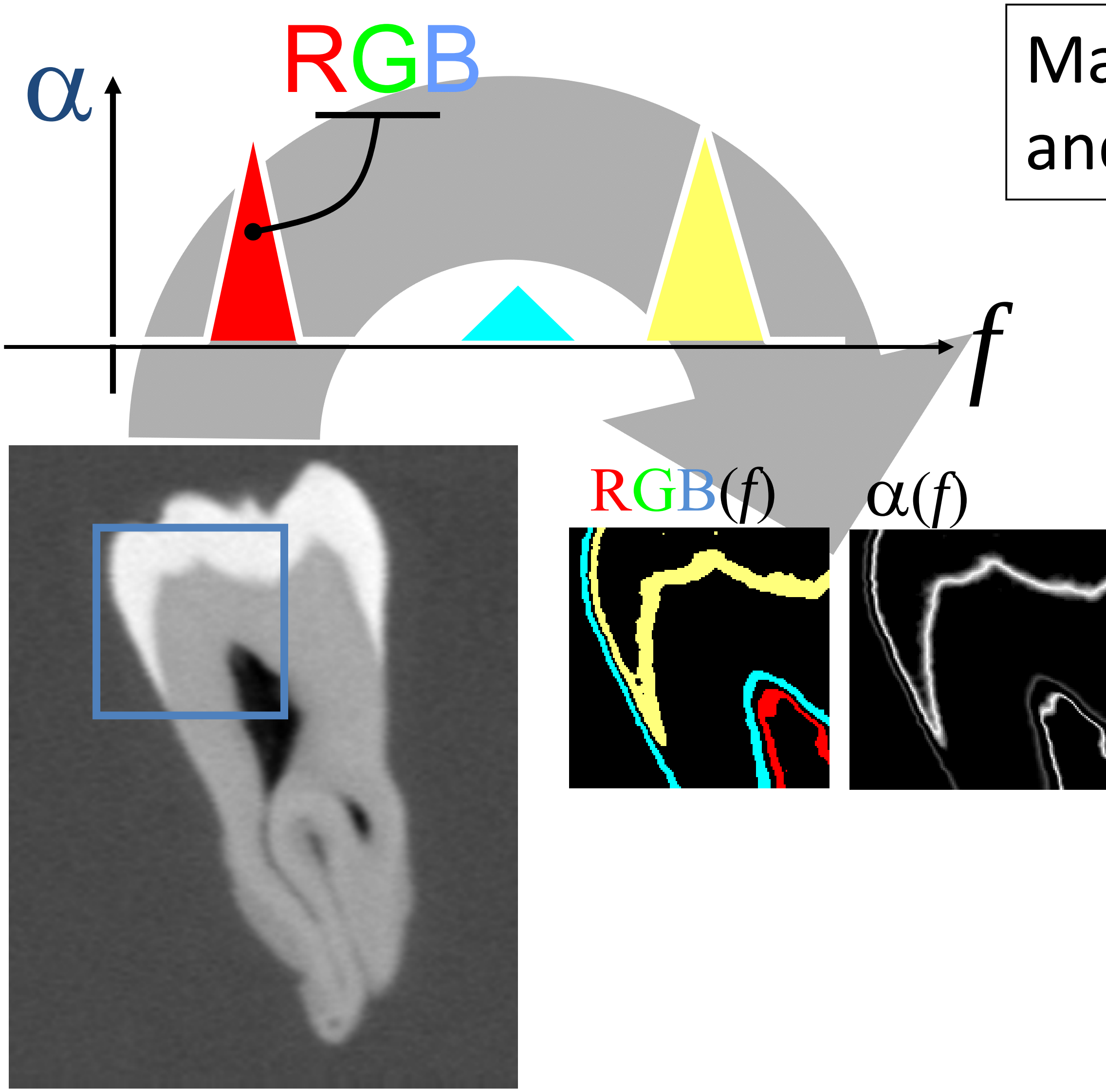


Human Tooth CT

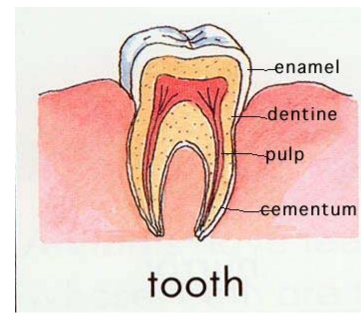
Transfer Functions (TFs)



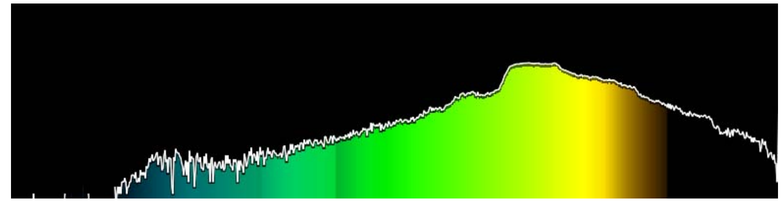
Map data value f to color and opacity



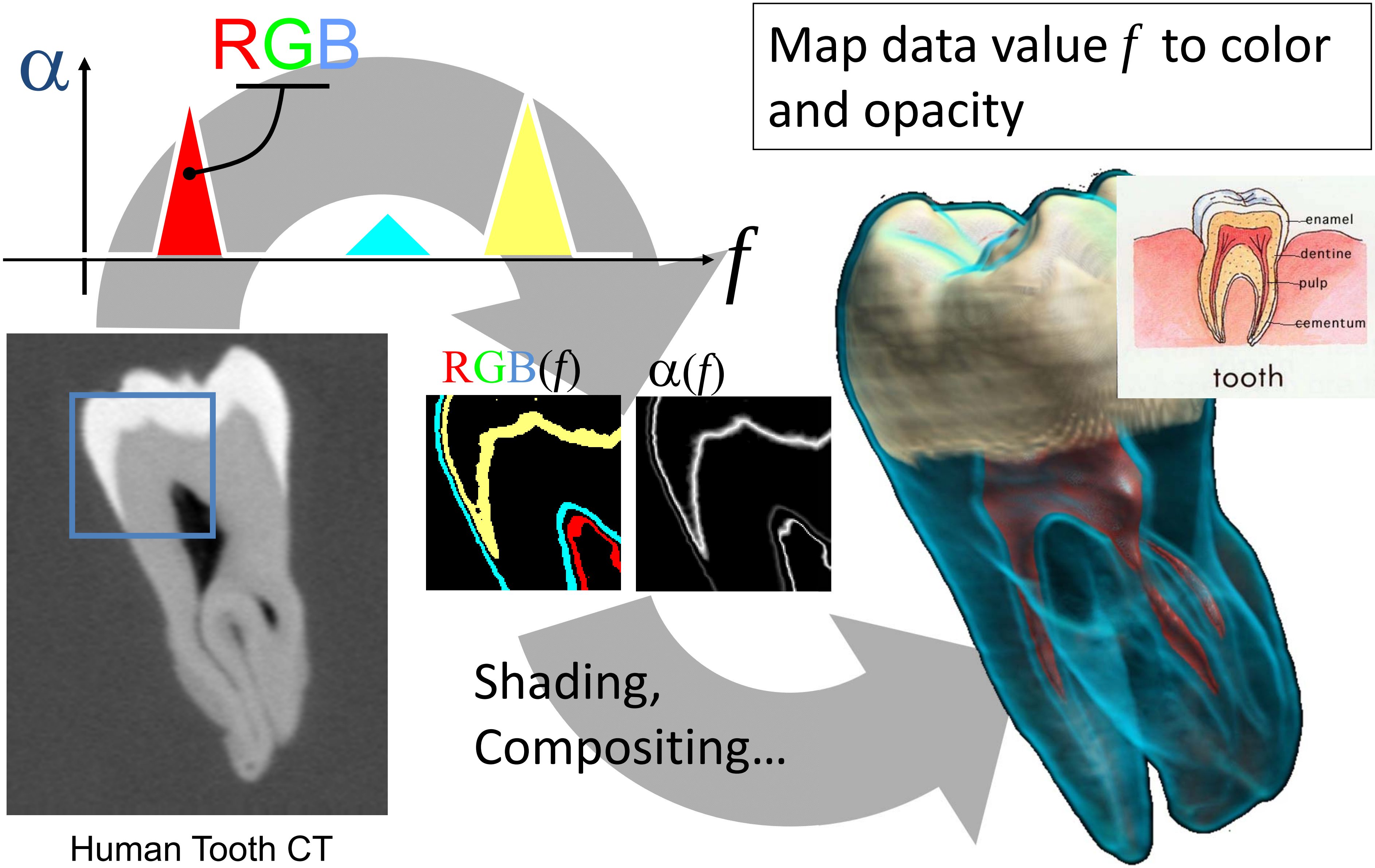
Human Tooth CT



Transfer Functions (TFs)



Map data value f to color and opacity



Human Tooth CT

Gradient

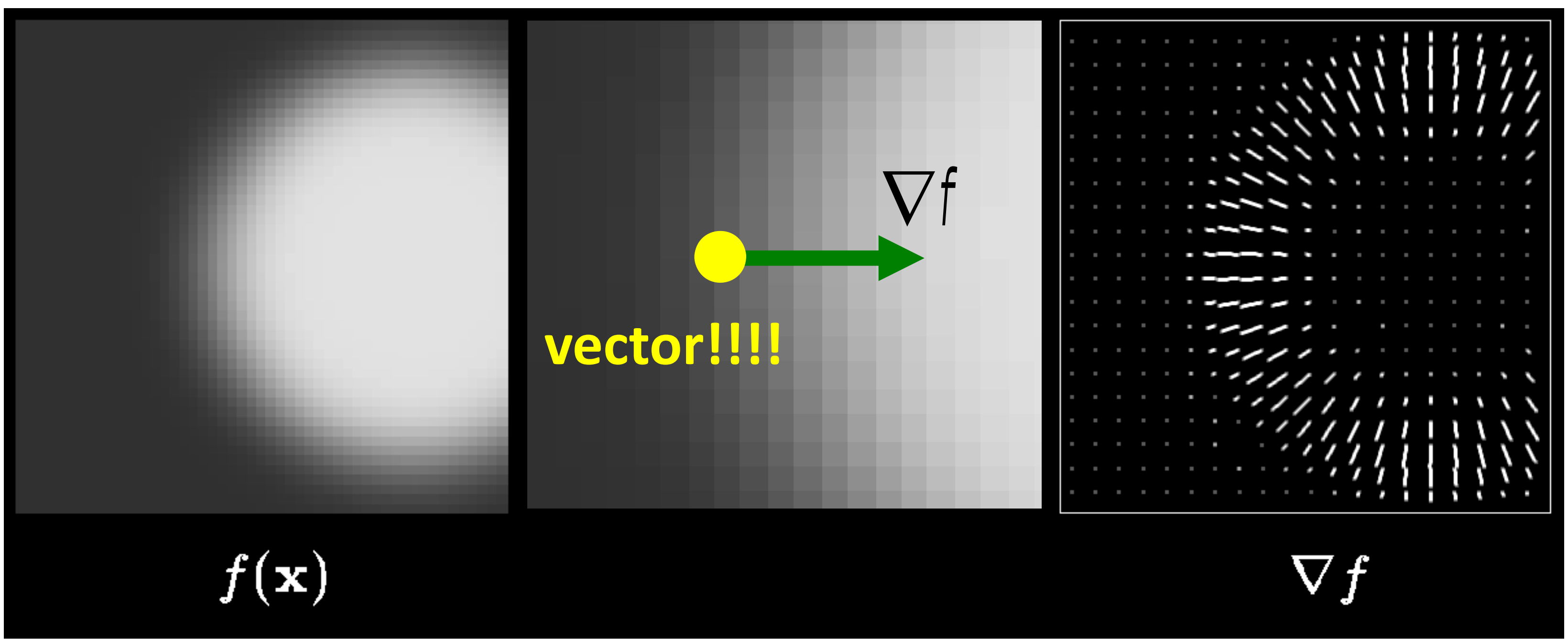
$\nabla f = (df/dx, df/dy, df/dz)$
$= ((f(1,0,0) - f(-1,0,0))/2,$
$(f(0,1,0) - f(0,-1,0))/2,$
$(f(0,0,1) - f(0,0,-1))/2)$

Central difference

$$\frac{df}{dx} = \frac{f(x+h) - f(x-h)}{2h}$$

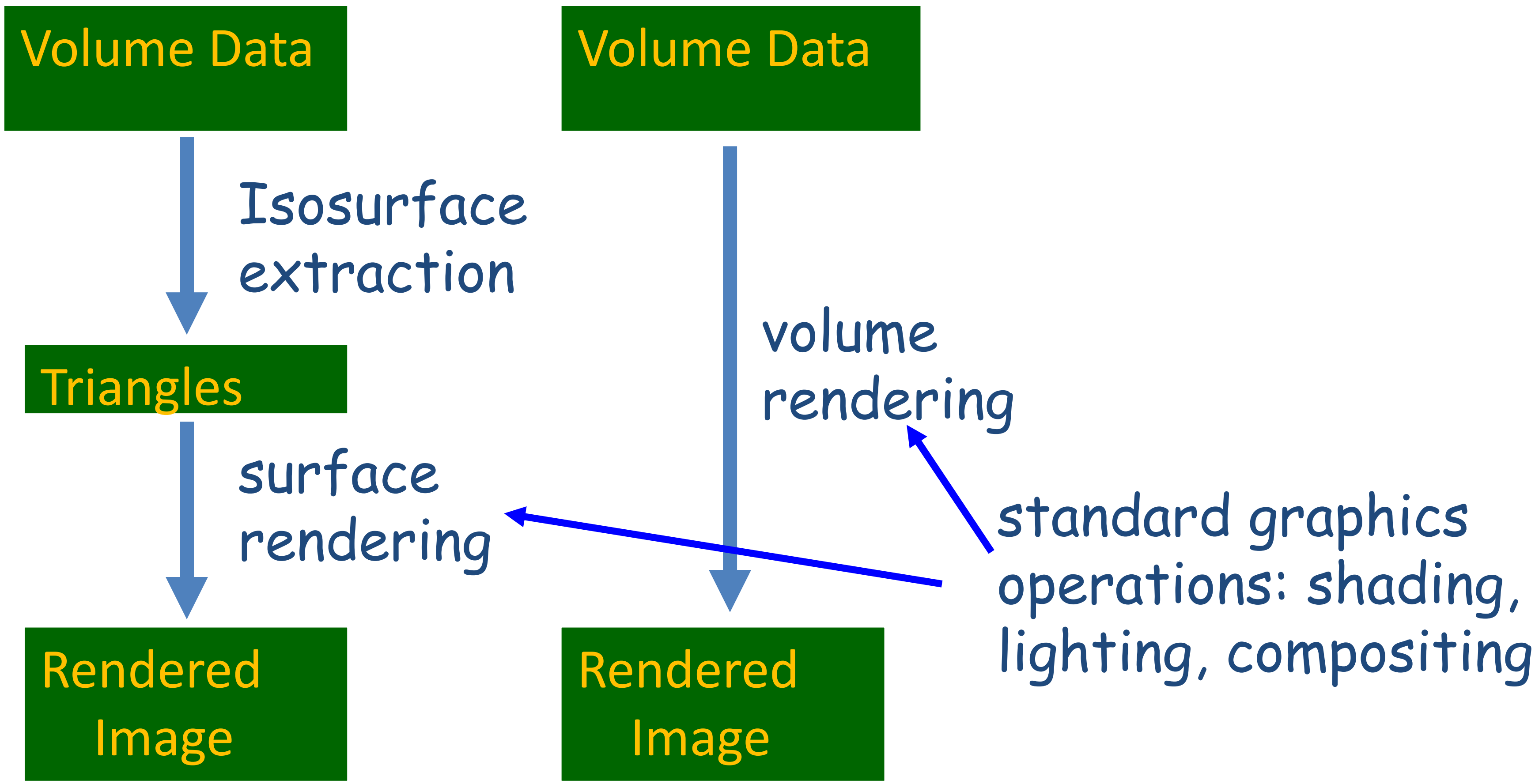
$$\frac{df}{dy} = \frac{f(y+h) - f(y-h)}{2h}$$

Approximates "surface normal“ (of iso-surface!)



Pipelines: Iso vs. Vol Ren

- The standard line - "no intermediate geometric structures"



Computational Strategies

- How can the basic ingredients be combined:
  - Image Order (in screen coordinate)
    - Ray casting (many options)
  - Object Order (in world coordinate)
    - splatting, texture-mapping
  - Combination (neither)
    - Shear-warp, Fourier

Computational Strategies

- How can the basic ingredients be combined:
  - Image Order (in screen coordinate)
    - Ray casting (many options)
  - Object Order (in world coordinate)
    - splatting, texture-mapping
  - Combination (neither)
    - Shear-warp, Fourier

Image Order

- Render image <u>one pixel at a time</u>

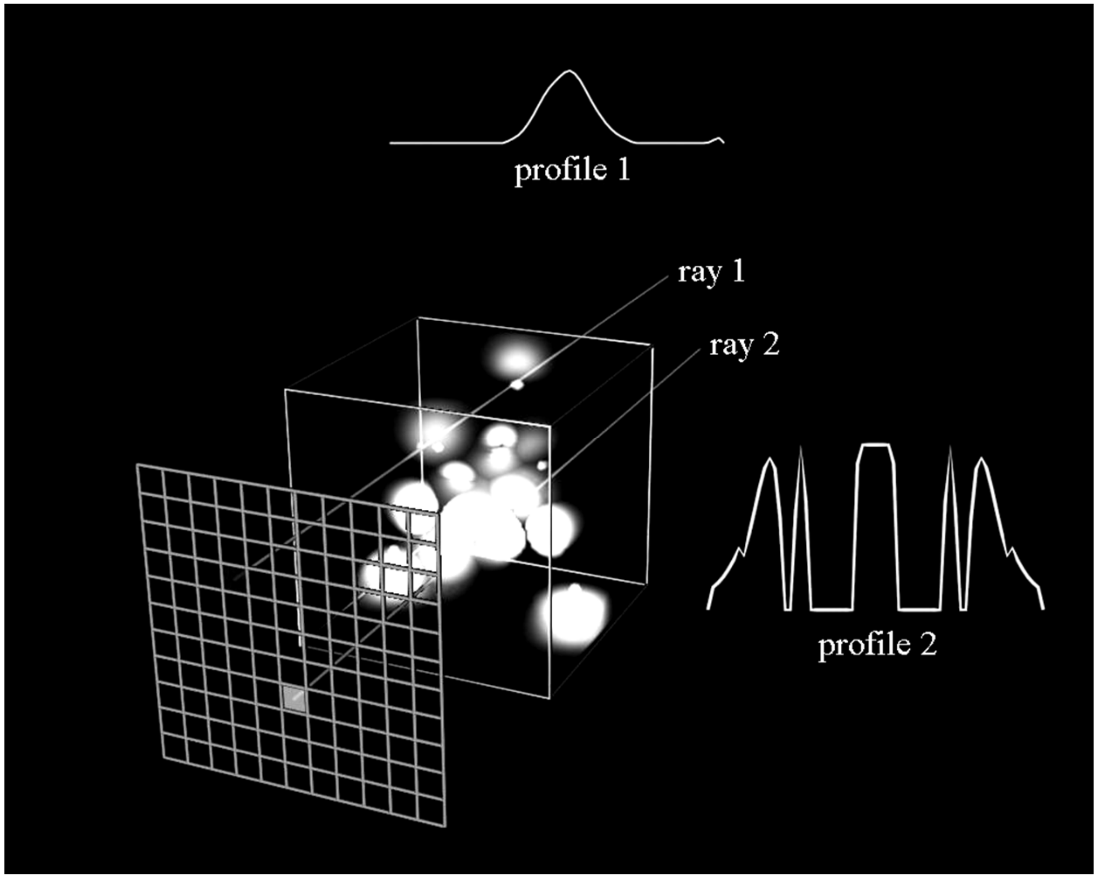
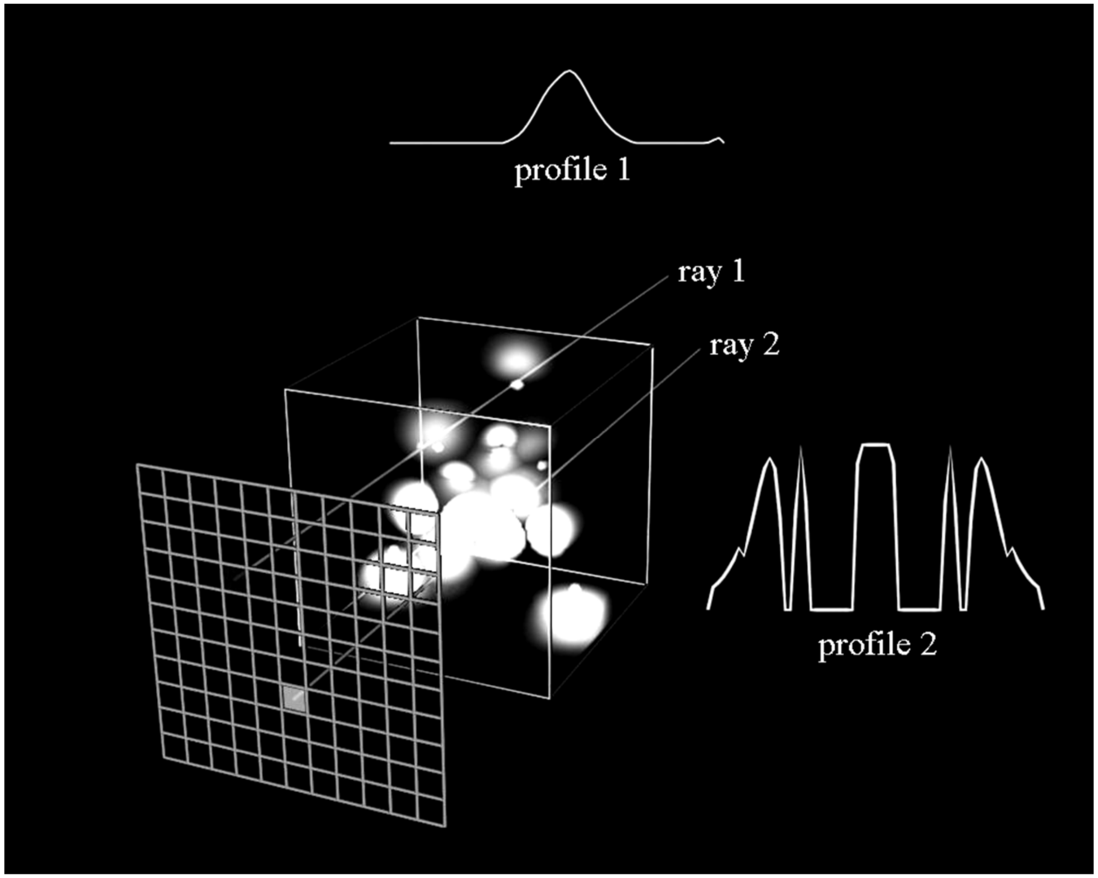


Image Order

- Render image <u>one pixel at a time</u>



For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Raycasting

- Raycasting is historically the **first volume rendering technique**.

- It shares some similarity with raytracing:
  - image-space method: main loop is over pixels of output image
  - a view ray per pixel (or per sub-pixel) is traced **backward**
  - samples are taken along the ray and composited to a single color

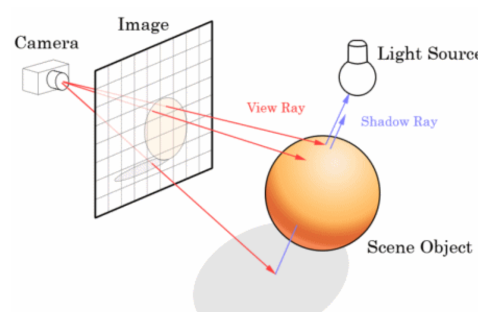


Image source: wikipedia

Raycasting

- Raycasting is historically the **first volume rendering technique**.
- It shares some similarity with raytracing:
  - image-space method: main loop is over pixels of output image
  - a view ray per pixel (or per sub-pixel) is traced **backward**
  - samples are taken along the ray and composited to a single color
- **Differences** are:
  - no secondary (reflected, shadow) rays
  - transmitted ray is not refracted
  - more elaborate compositing functions
  - samples are taken at intervals (not at object intersections) inside volume

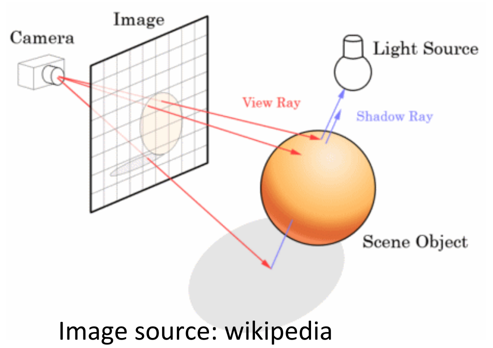
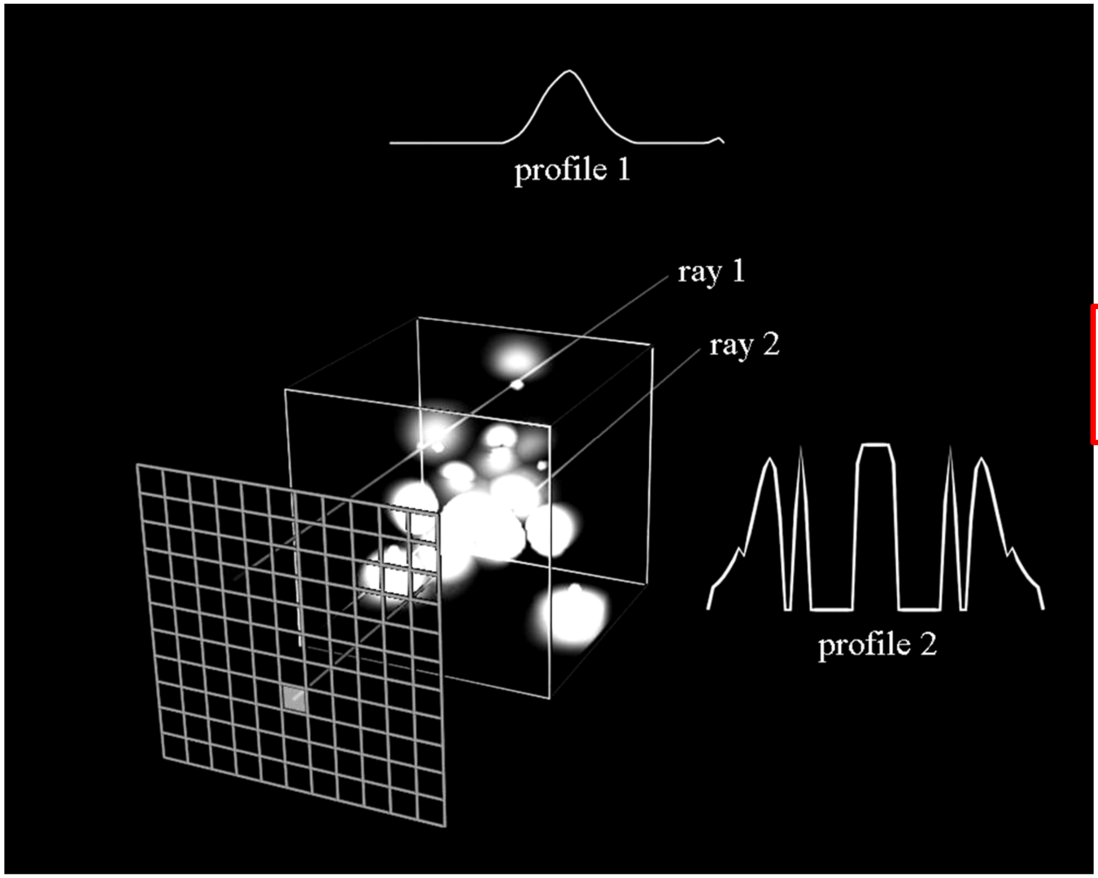


Image source: wikipedia

Image Order

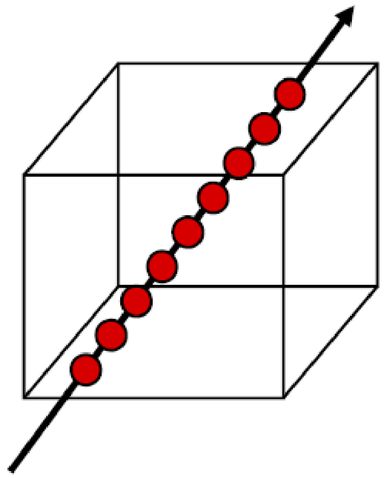
- Render image <u>one pixel at a time</u>



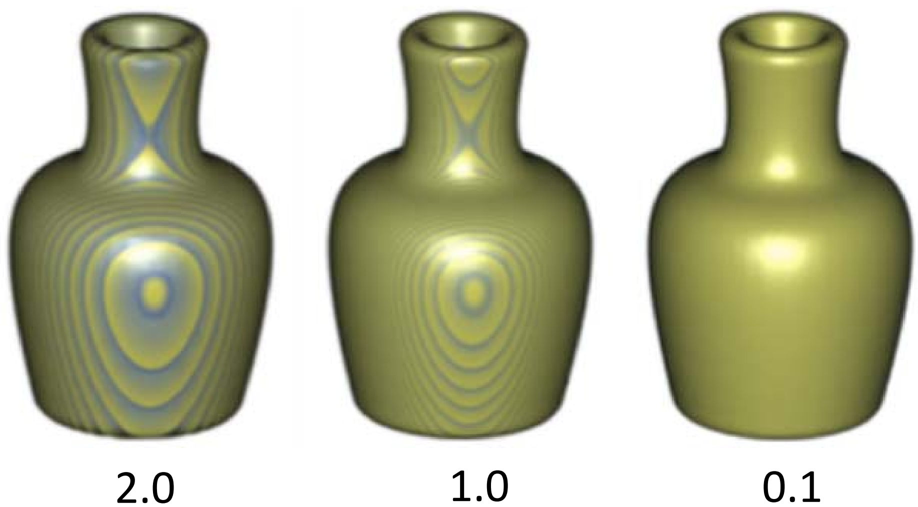
For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Raycasting

Sampling interval can be fixed or adjusted to voxels:



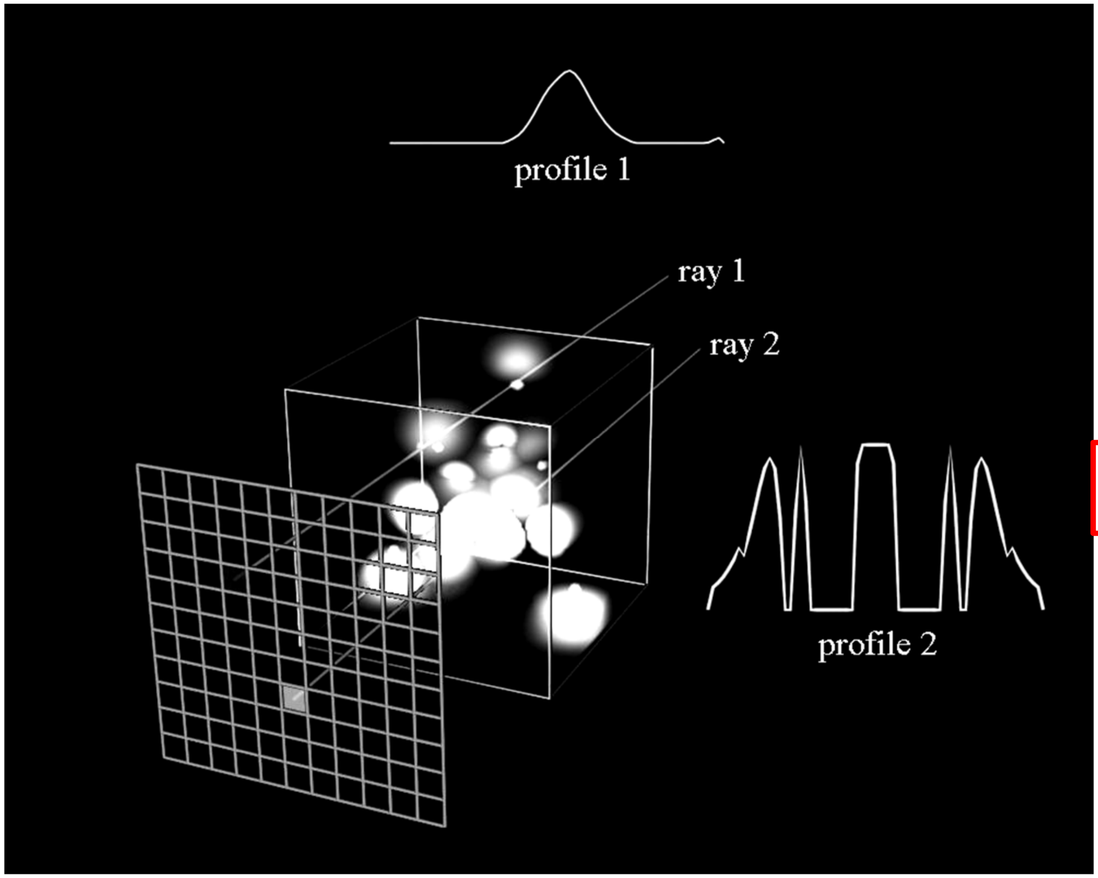
uniform sampling



Images generated using a ray casting method with three different step sizes (or sample rate). Vase data courtesy of SUNY Stony Brook.

Image Order

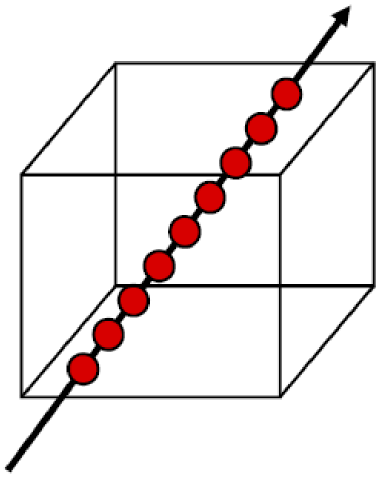
- Render image <u>one pixel at a time</u>



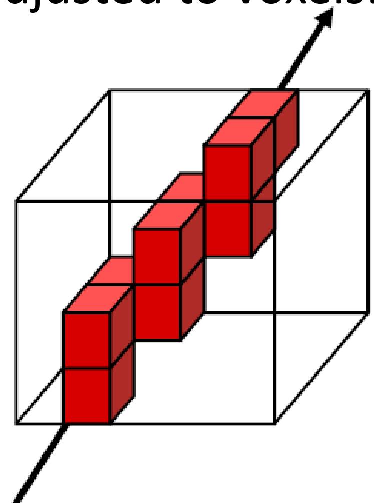
For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Raycasting

Sampling interval can be fixed or adjusted to voxels:



uniform sampling



voxel-by-voxel traversal (faster)

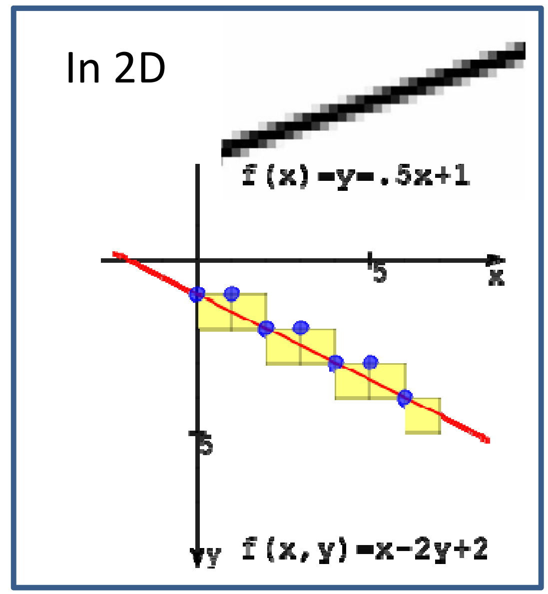
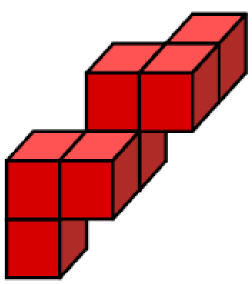
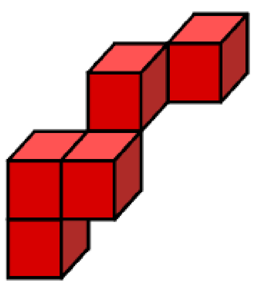


Image source: wikipedia

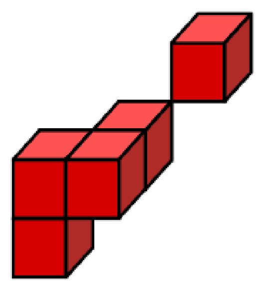
Connectedness of "voxelized" rays:



6-connected (strongest)



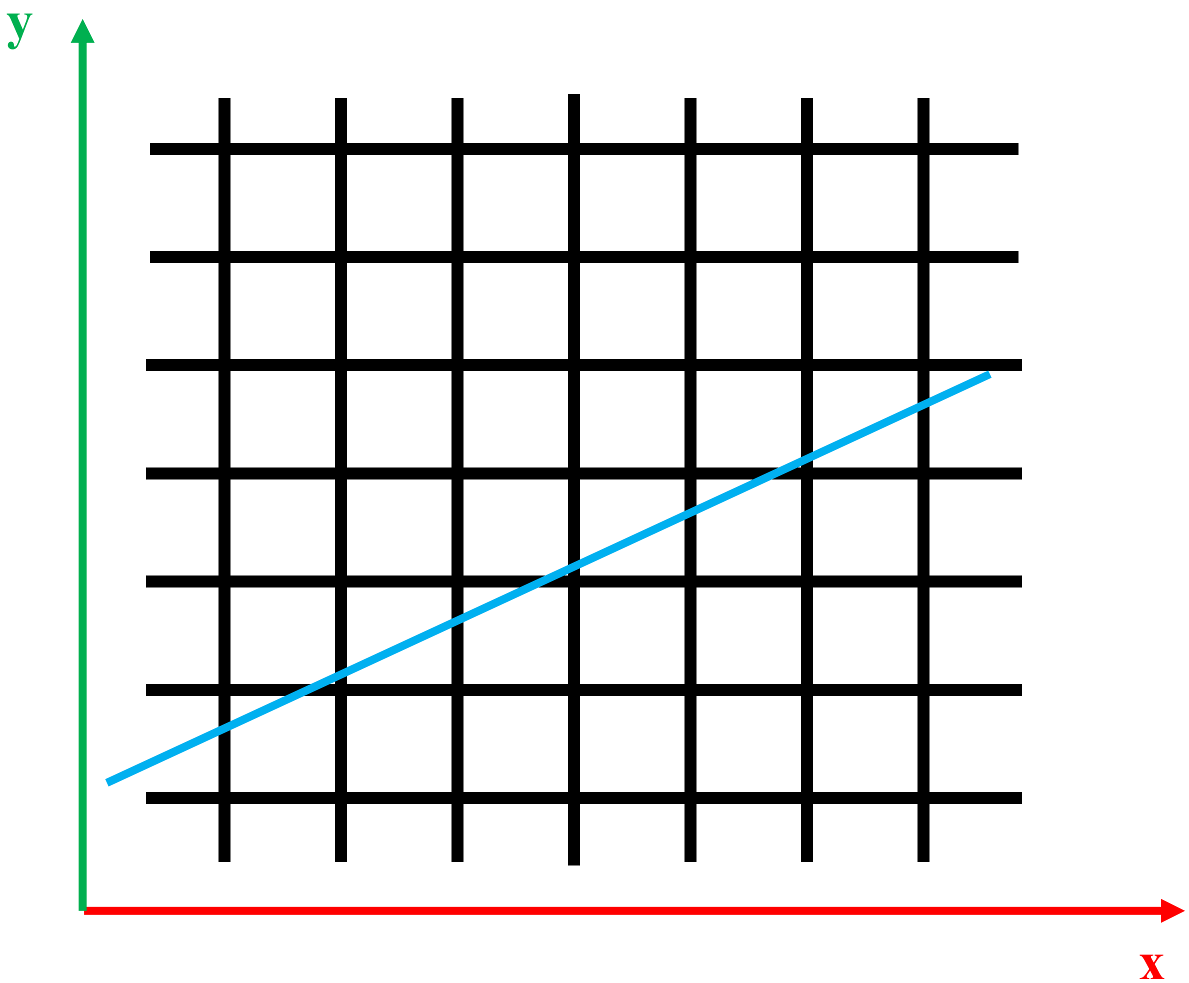
18-connected



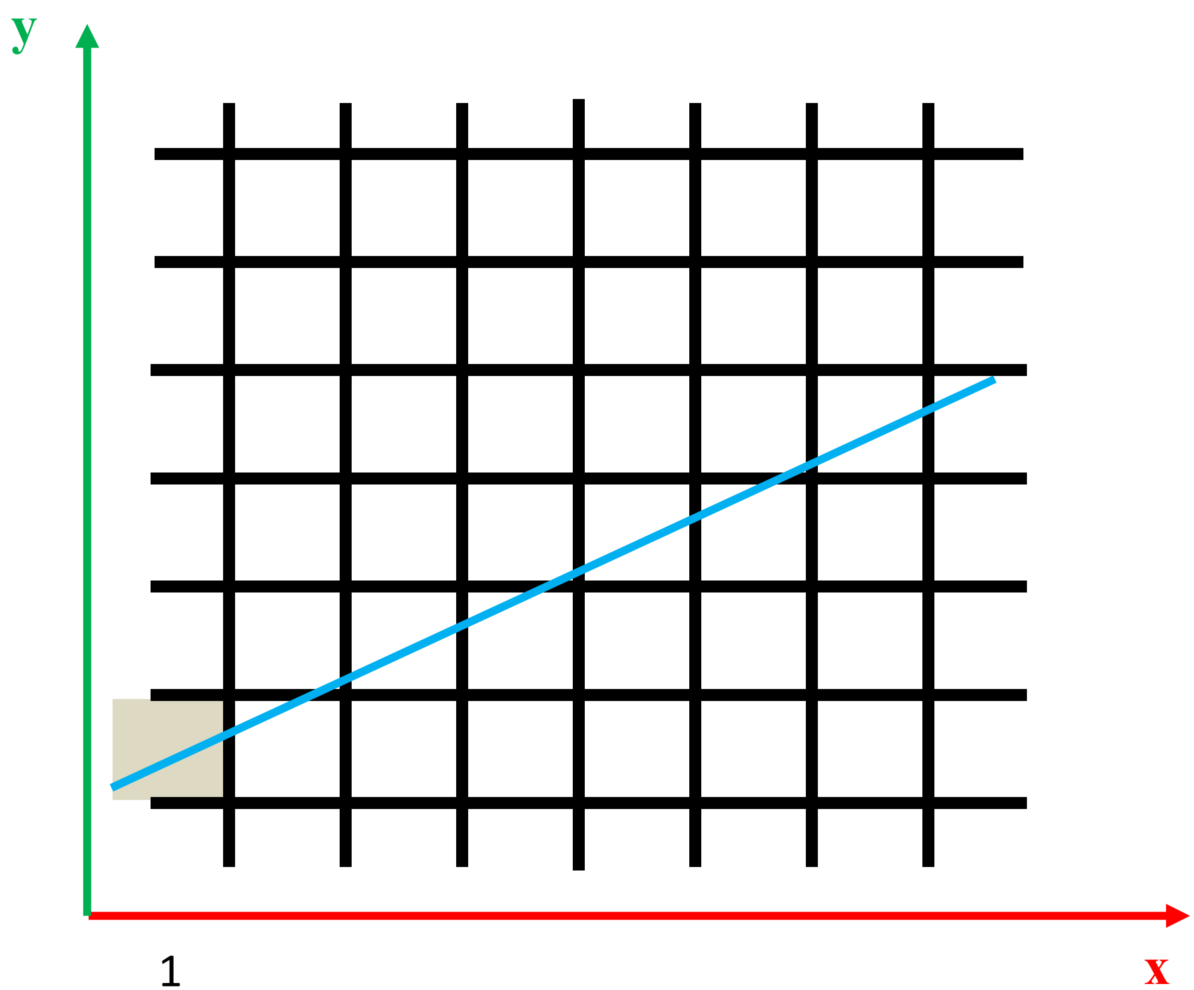
26-connected (weakest)

Accelerate the sampling and interpolation

Line rasterization process

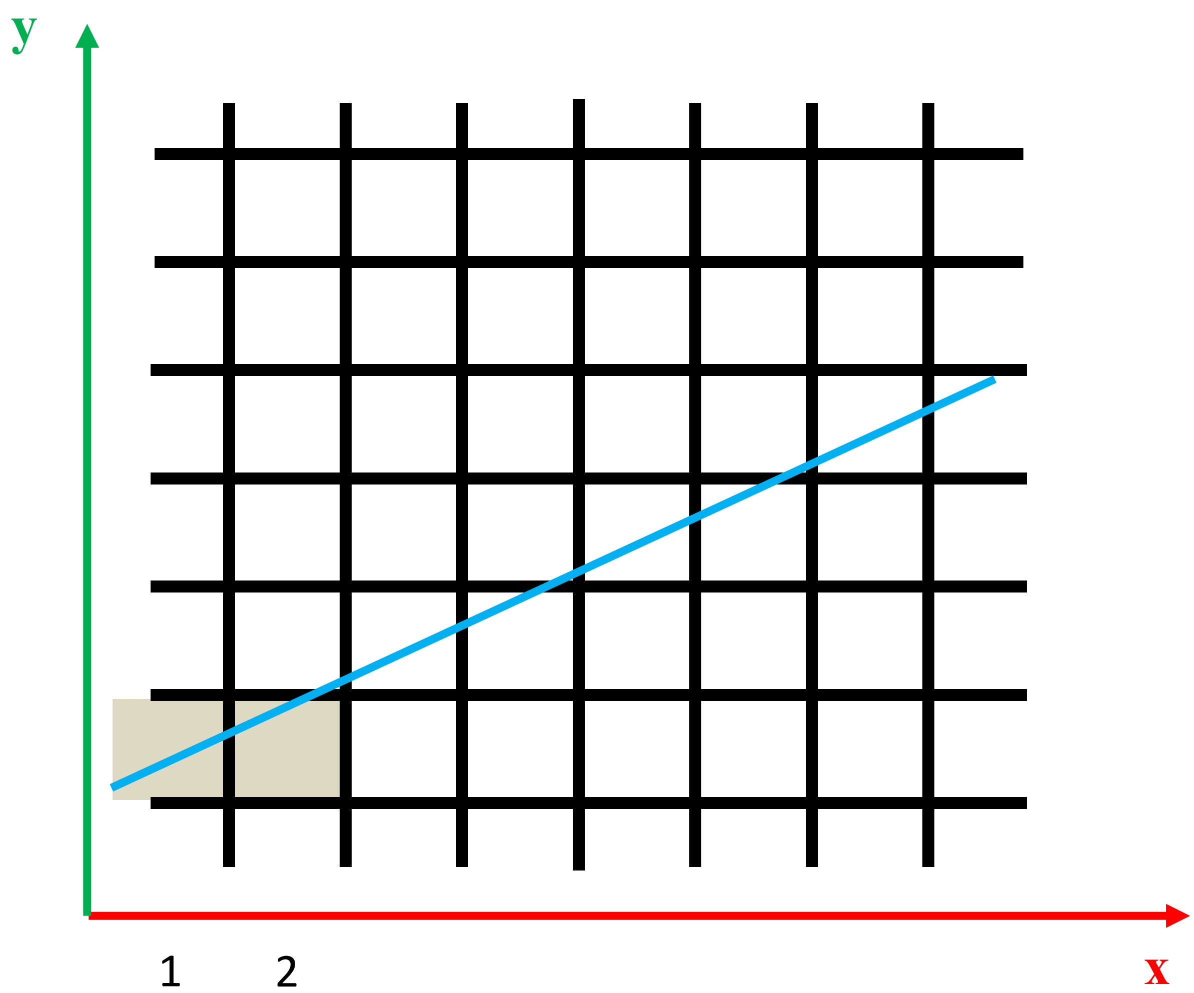


Line rasterization process



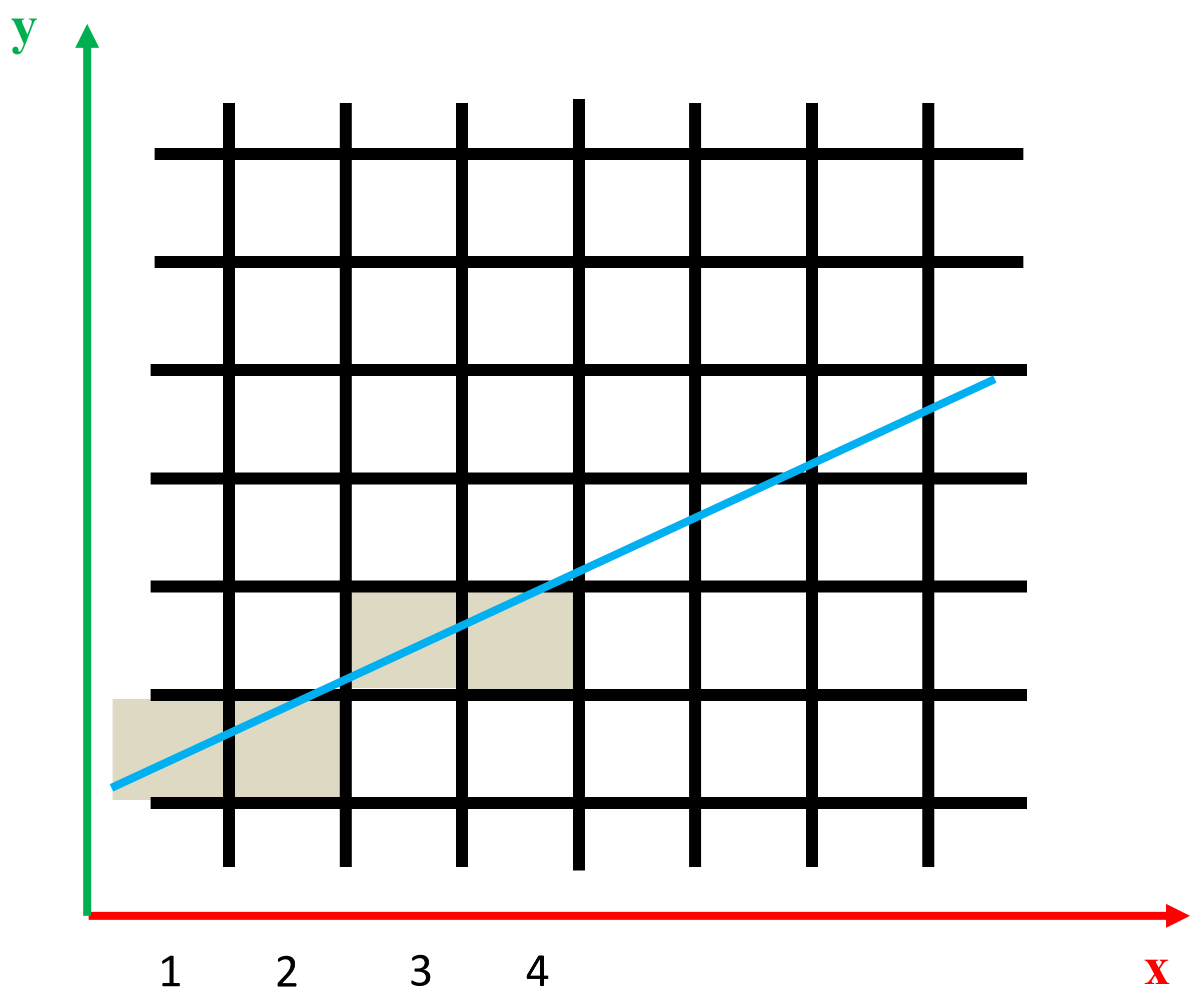
Let us increase along x

Line rasterization process



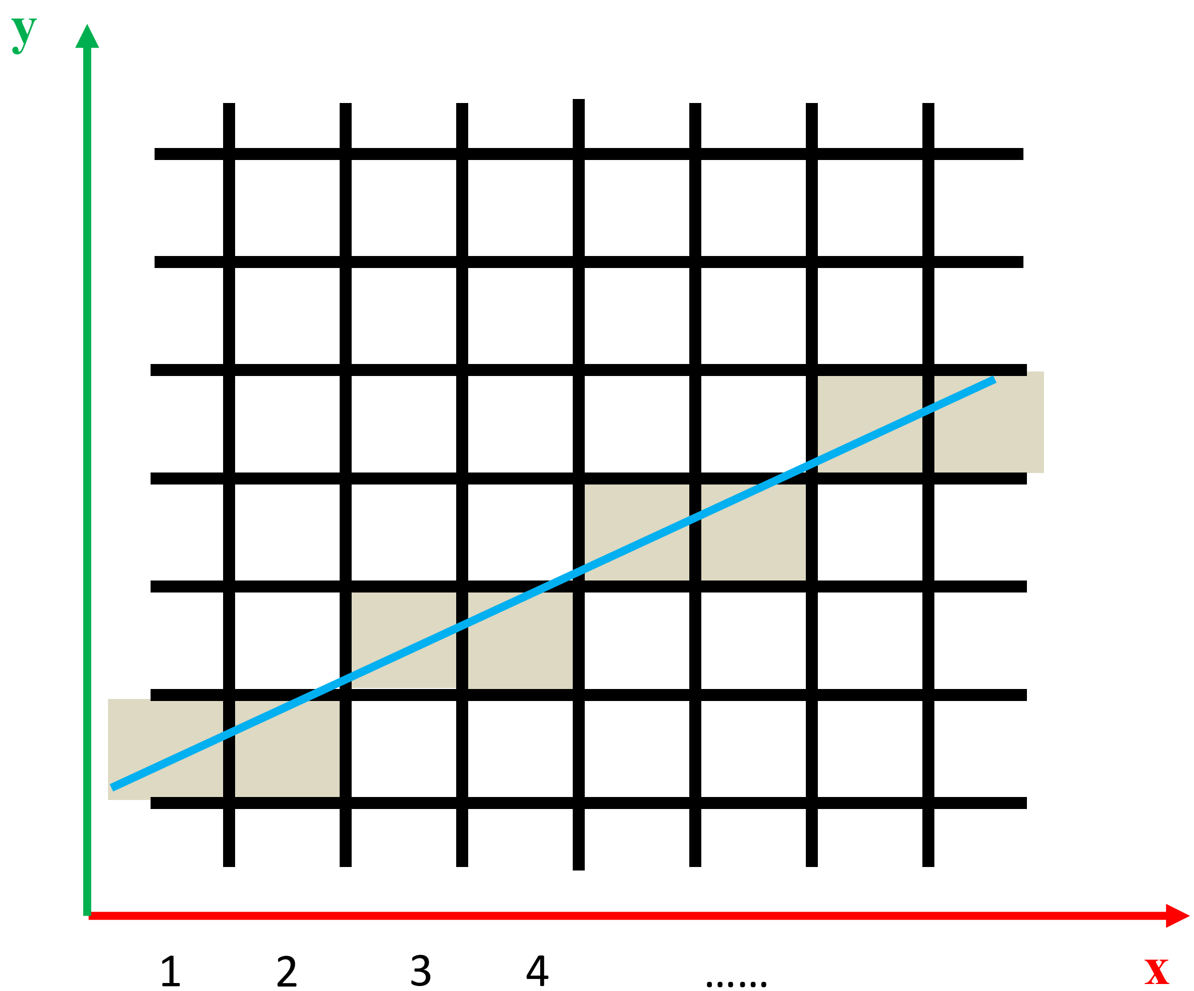
Let us increase along x

Line rasterization process



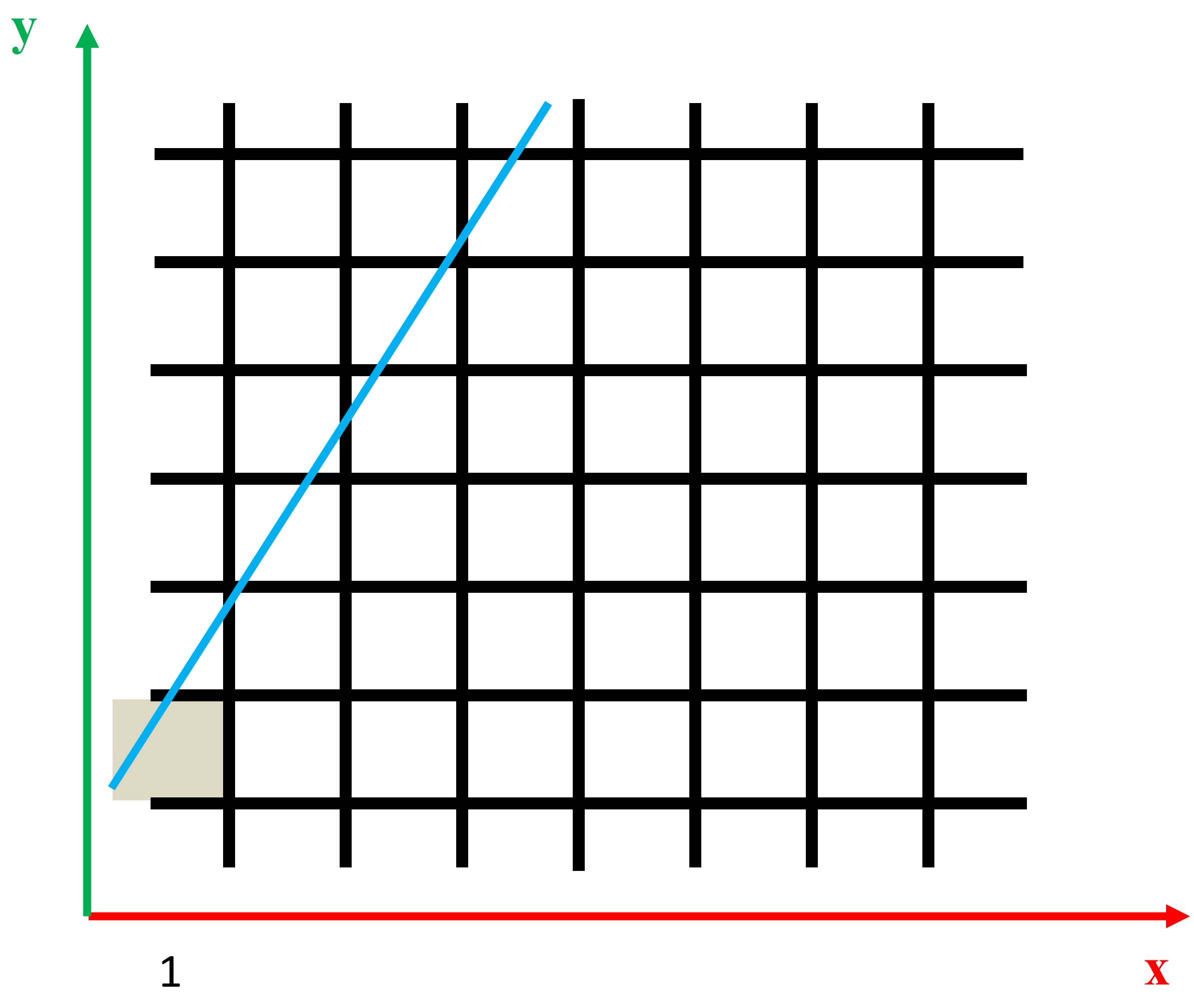
Let us increase along x

Line rasterization process



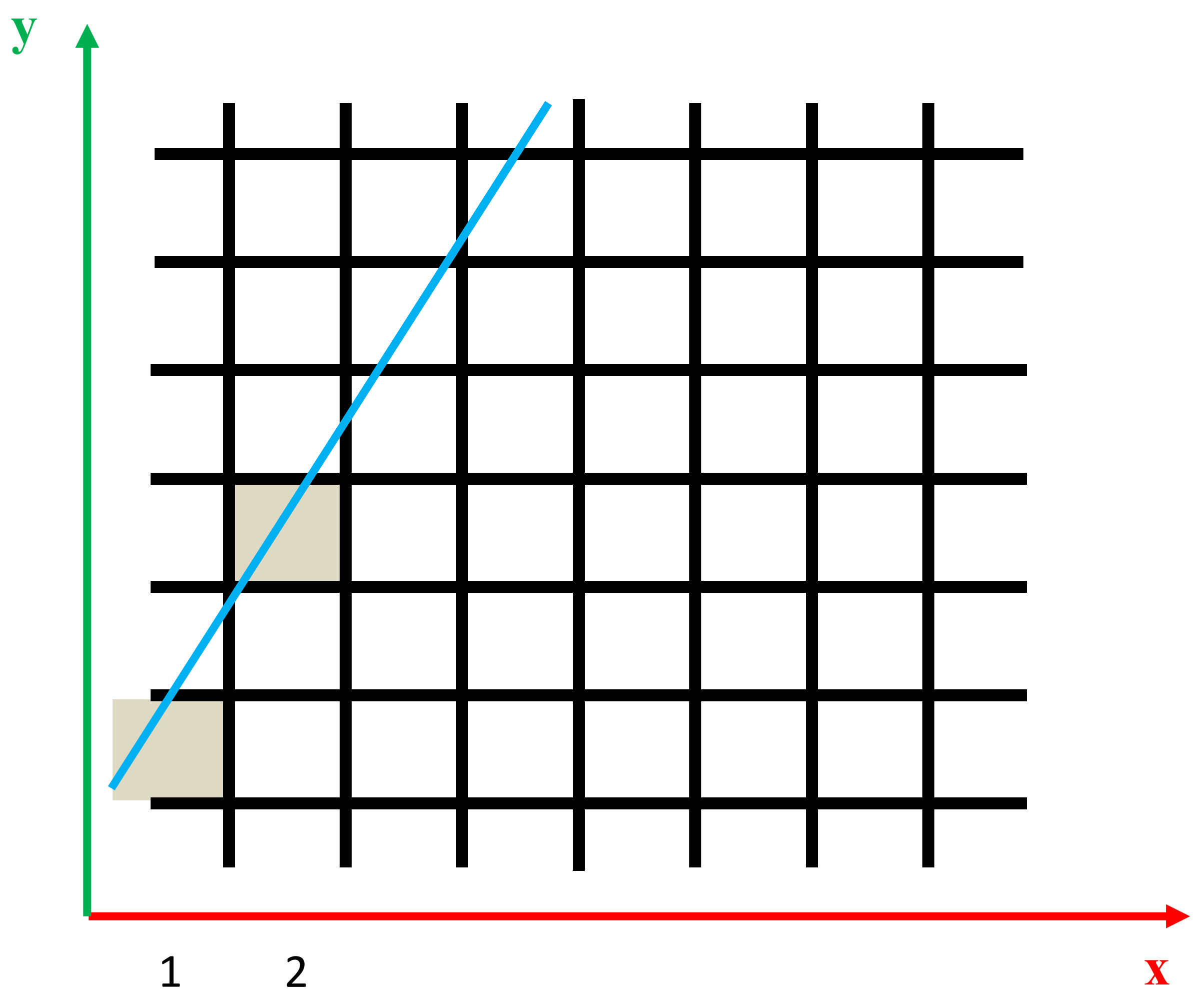
Let us increase along x

Line rasterization process



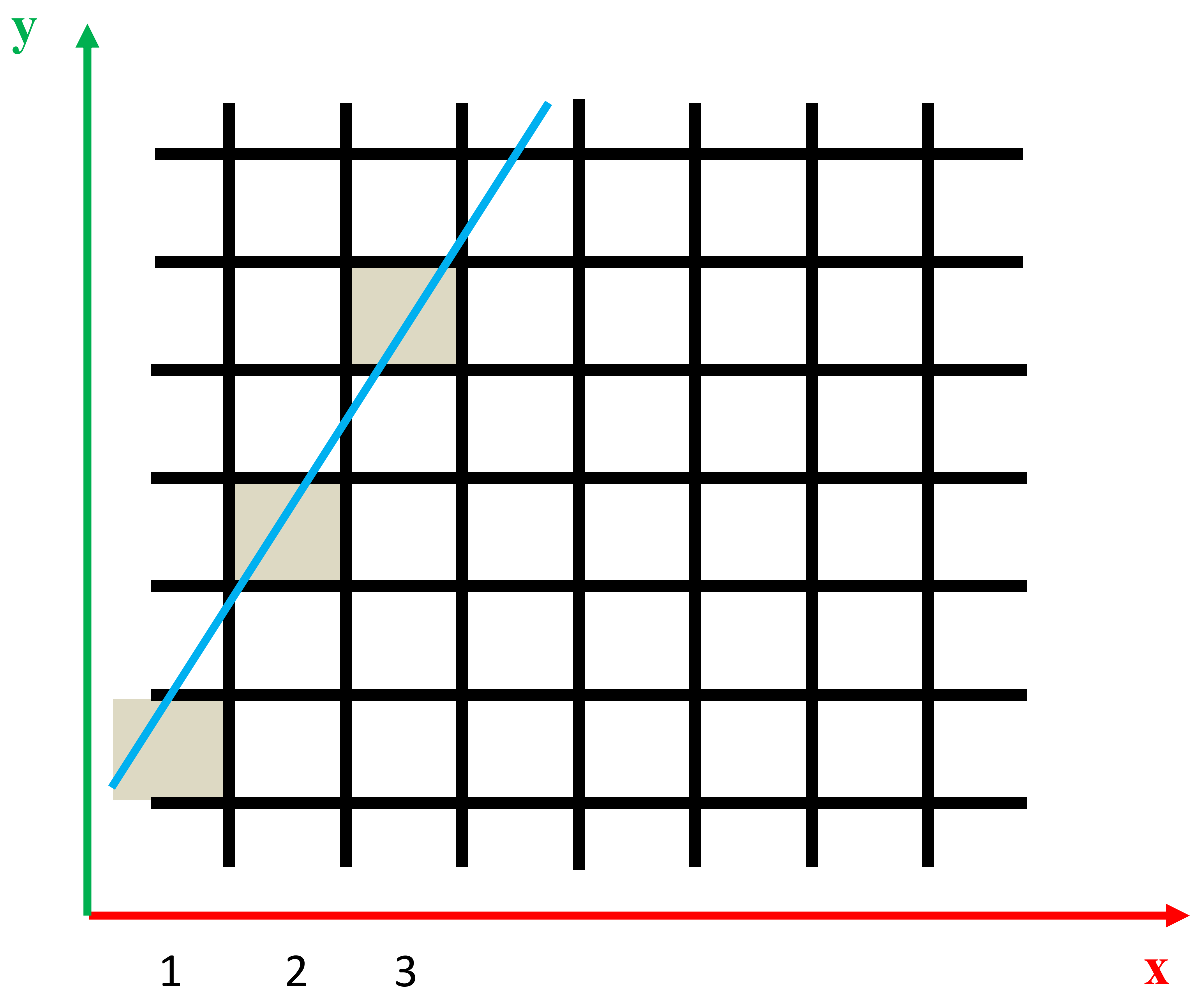
If we still increase along x, what will happen?

Line rasterization process



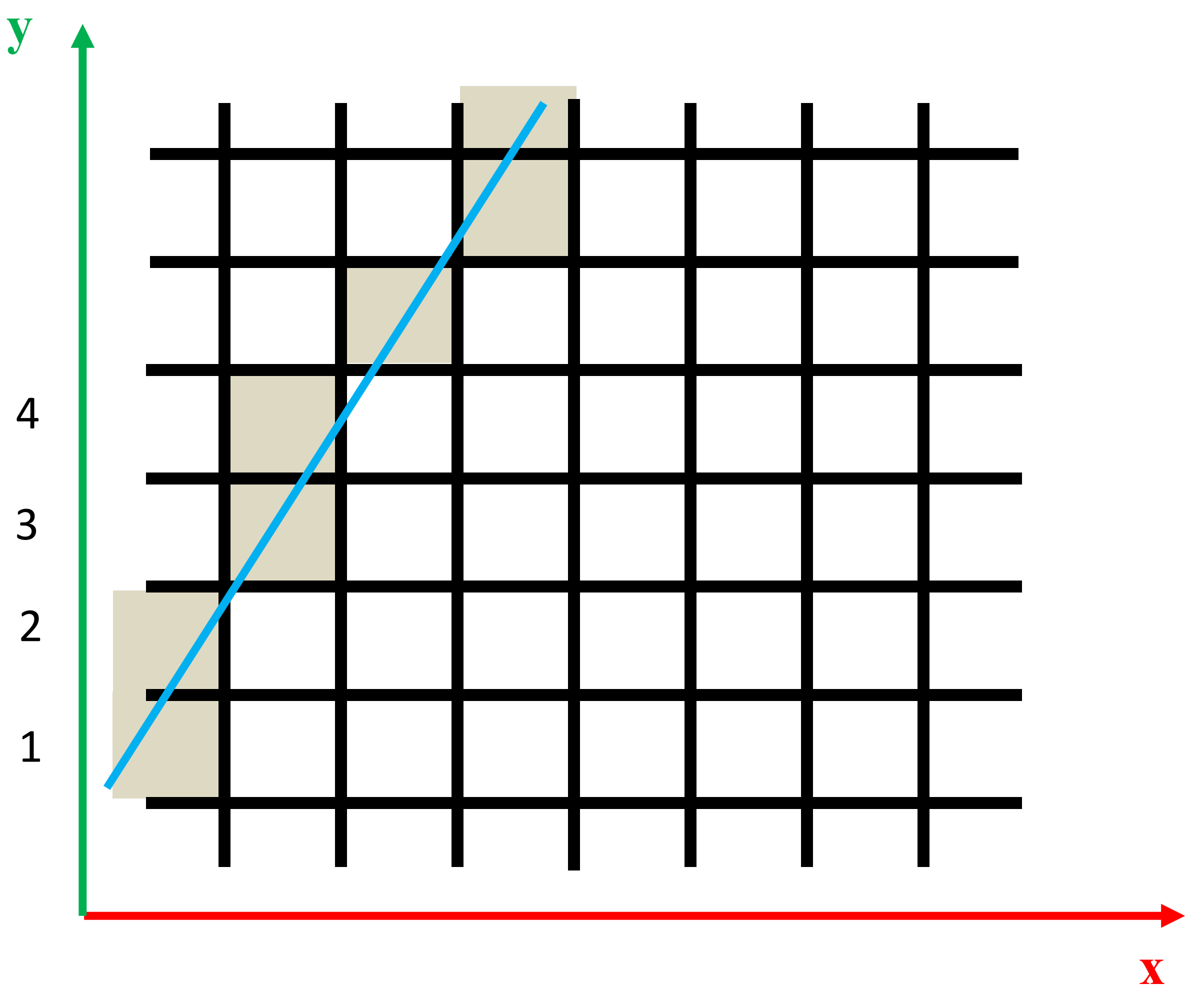
If we still increase along x, what will happen?

Line rasterization process



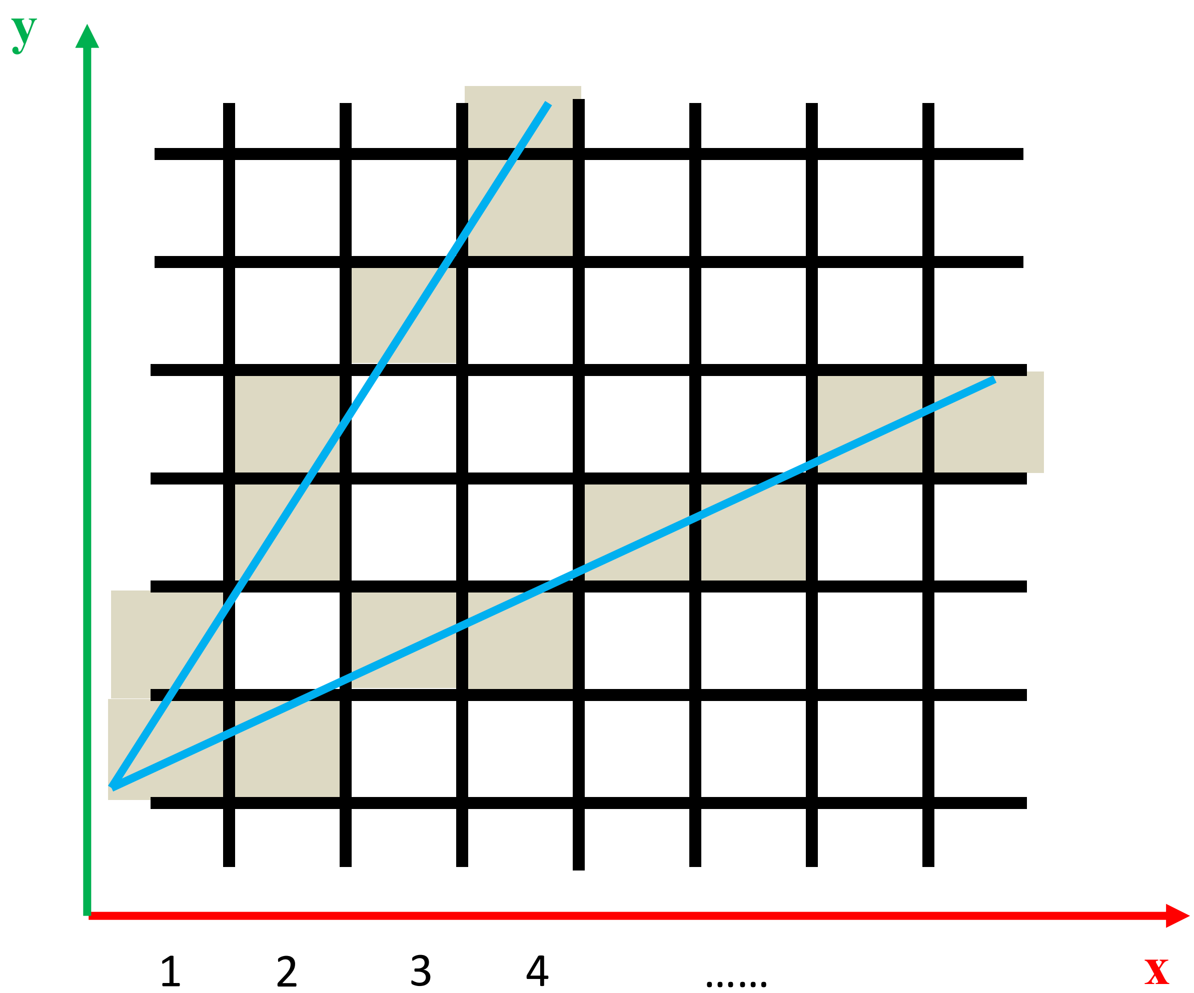
There will be many gaps between the pixels!!!

Line rasterization process



The correct way is to increase along y!!!

Line rasterization process

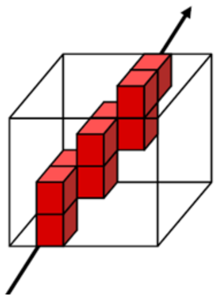


Lesson learned? We have to march along the axis that is most parallel to the line!

Ray Templates

A **ray template** (Yagel 1991) is a **voxelized ray** which by translating generates all view rays.

Ray templates speed up the sampling process, but are obviously **restricted** to **orthographic** views.



Algorithm:

• Rename volume axes such that z is the one "most orthogonal" to the image plane (without loss of generality).

• Create ray template with 3D version of **line pixelized** algorithm, giving 26-connected rays which are functional in z coordinate (have exactly one voxel per z-layer)

• Translate ray template in **base plane**, not in image plane

Accelerate the sampling and interpolation

Ray Templates

Incorrect: translated in image plane

Correct: translated in base plane