3D Scalar Field Visualization: Volume Rendering

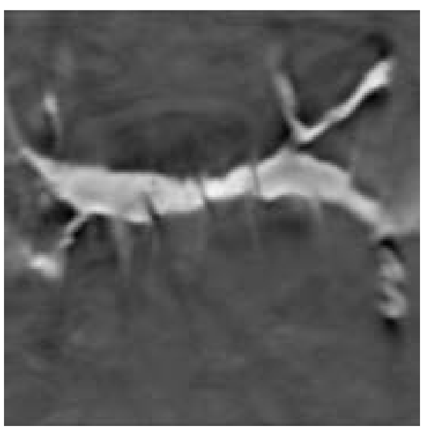
Goal: understand what is DVR and why it is useful; how to compute DVR (important steps); how to perform raycasting

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

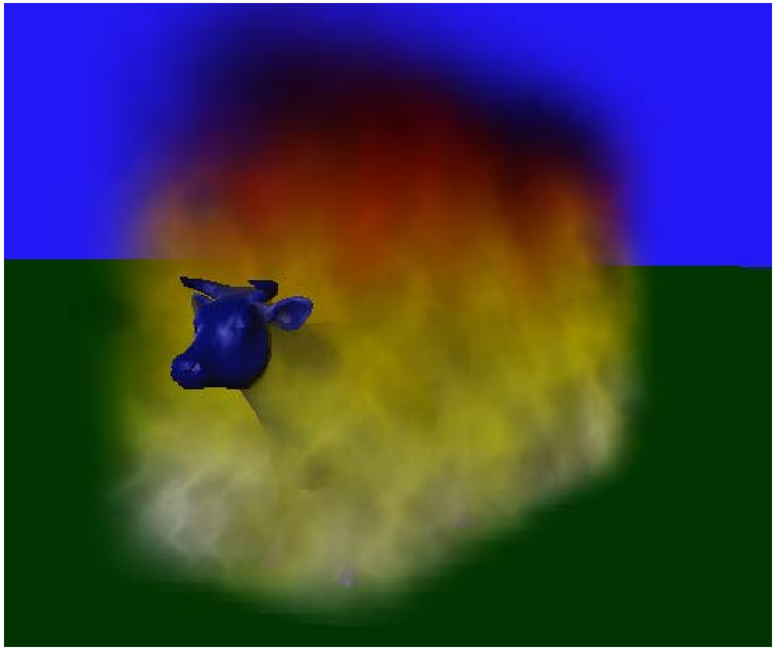
Isosurface

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

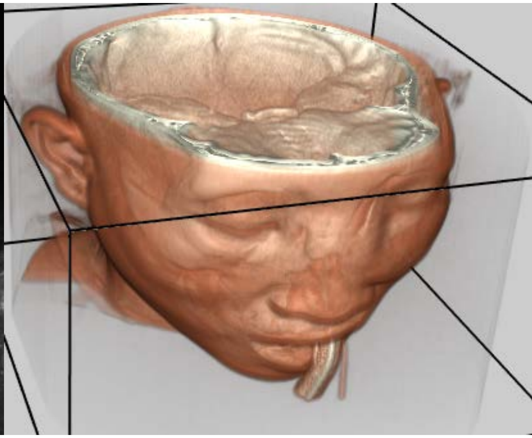
- Any rendering process which maps from volume data to an image without introducing binary distinctions / intermediate geometry

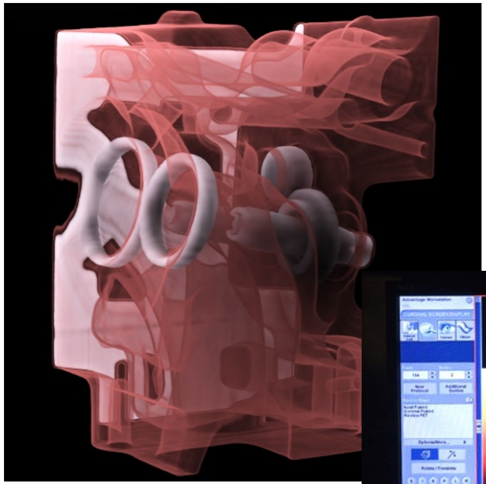
- How do you make the data visible? : Via <u>Color</u> and <u>Opacity</u>

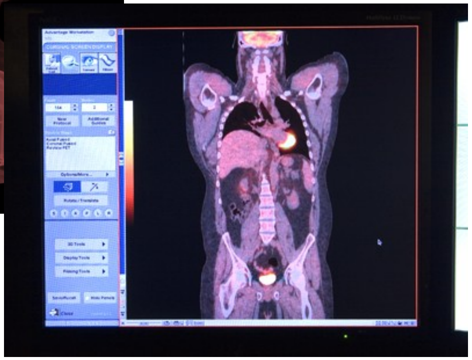
- How to achieve that?
  - The data is considered to represent a **semi-transparent light-emitting medium**
  - Approaches are based on the **laws of physics** (emission, absorption, scattering)
  - The volume data is **used as a whole** (look inside, see interior structures). Think of color plots in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

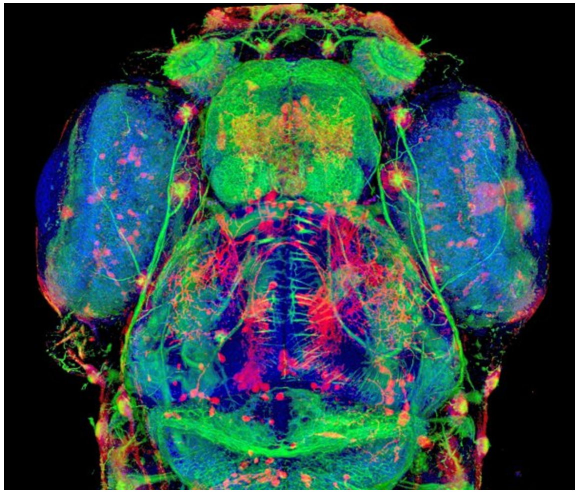
Volume Rendering is Useful

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



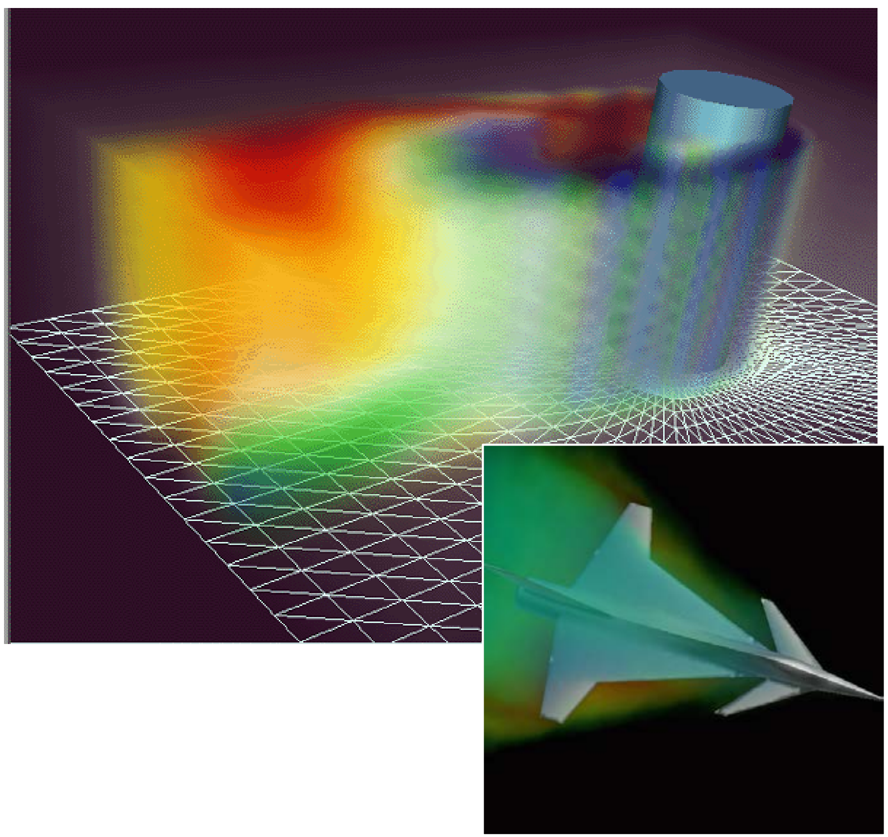


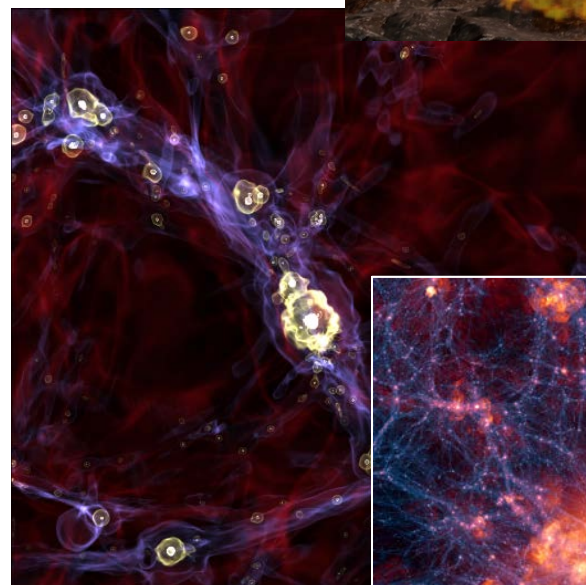


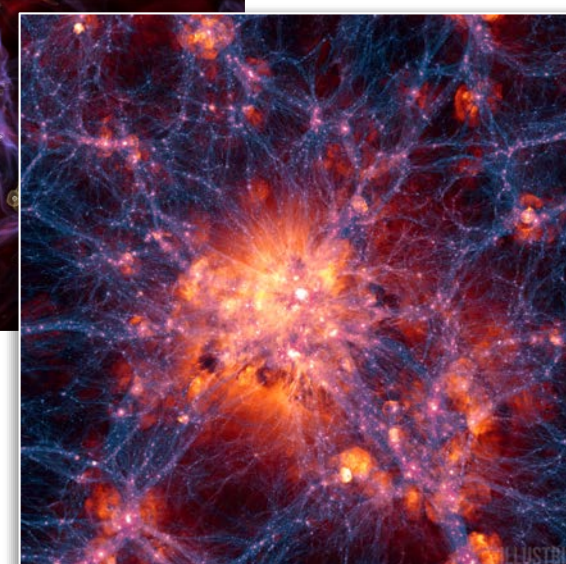


Volume Rendering is Useful

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







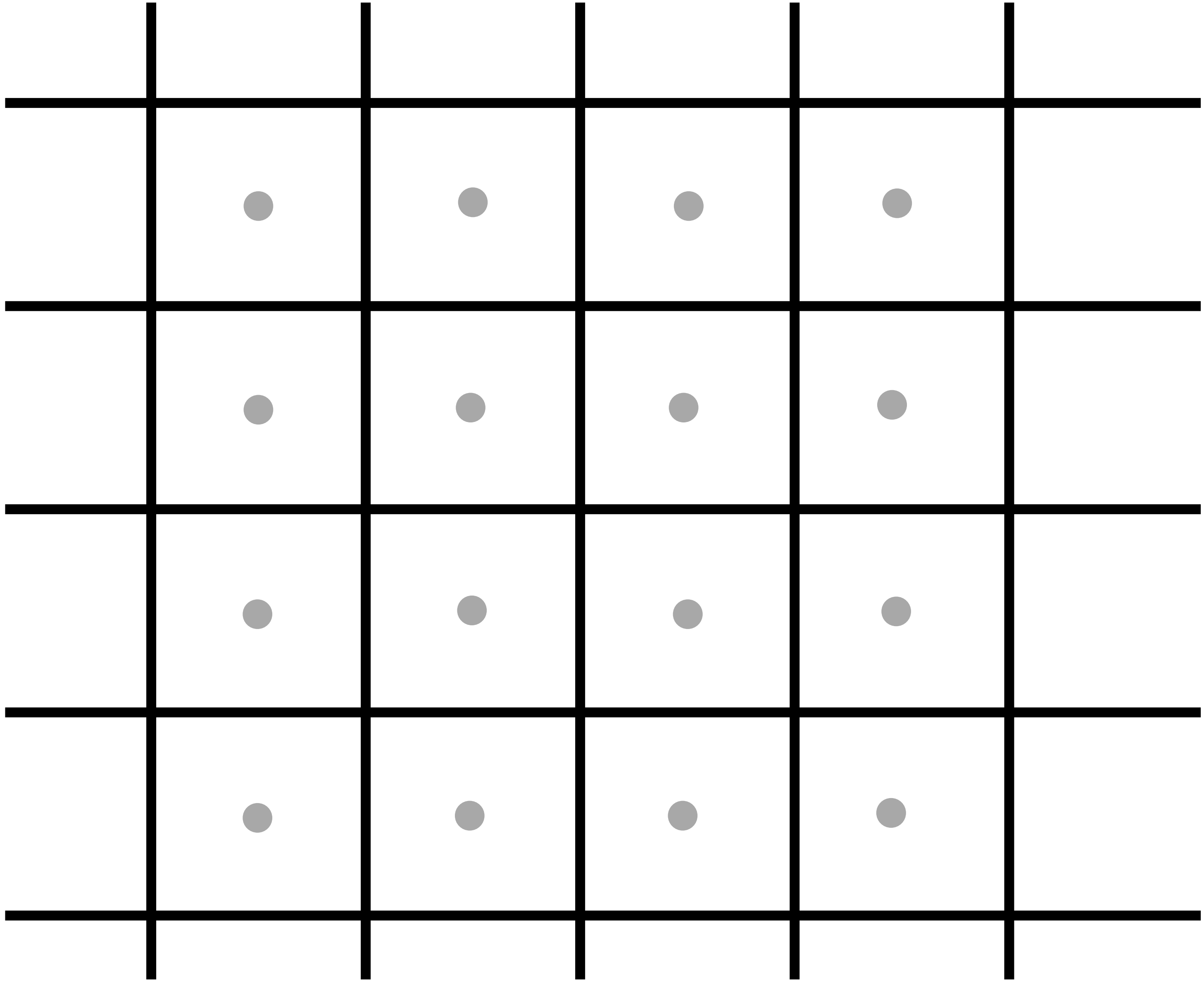
Data Representation

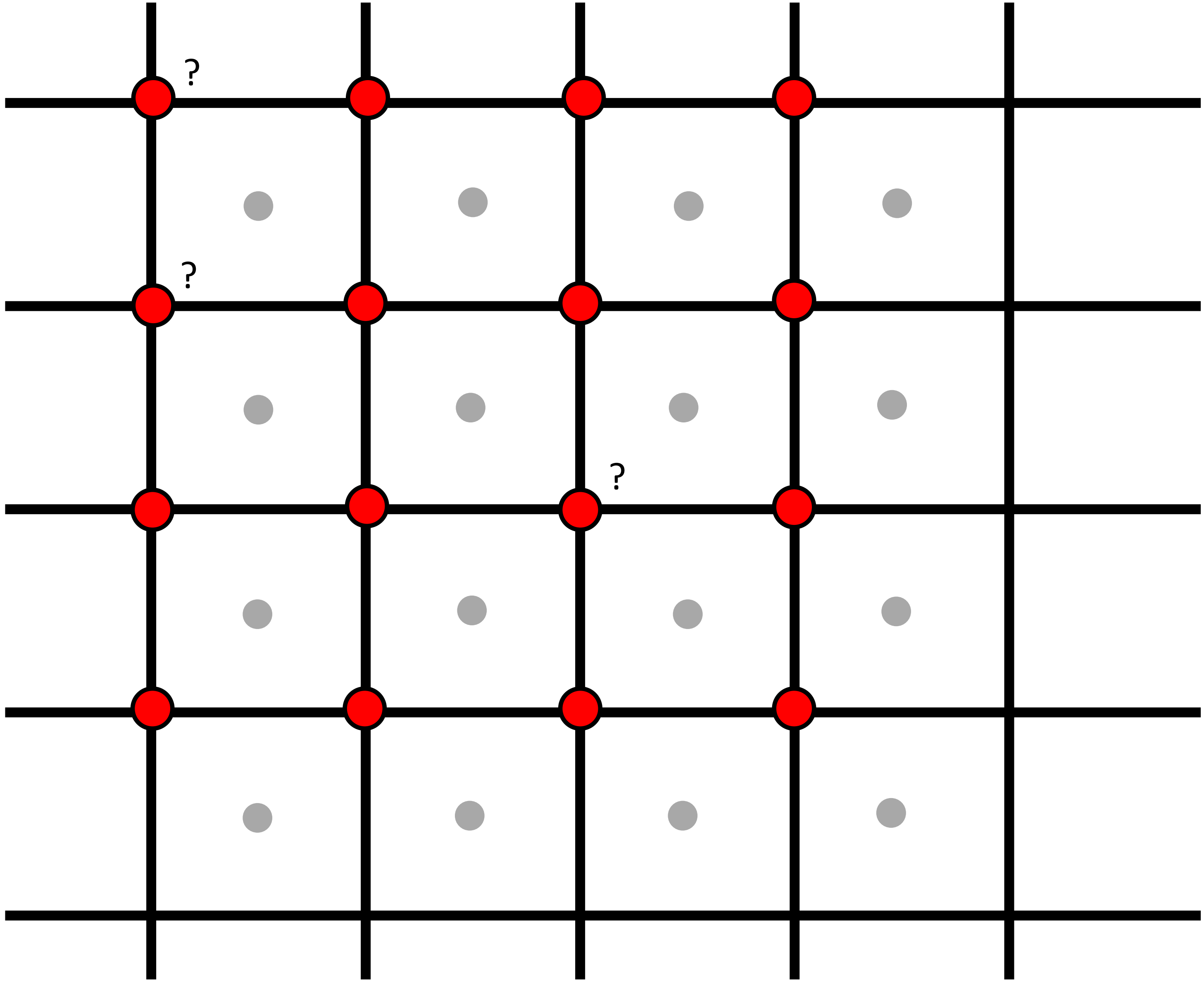
- Volume rendering techniques
  - depend strongly on the **grid type**
  - exist for both structured and unstructured grids
  - are predominantly applied to **uniform grids (3D images).**
  - for uniform grid, <u>**voxels**</u> are the basic unit

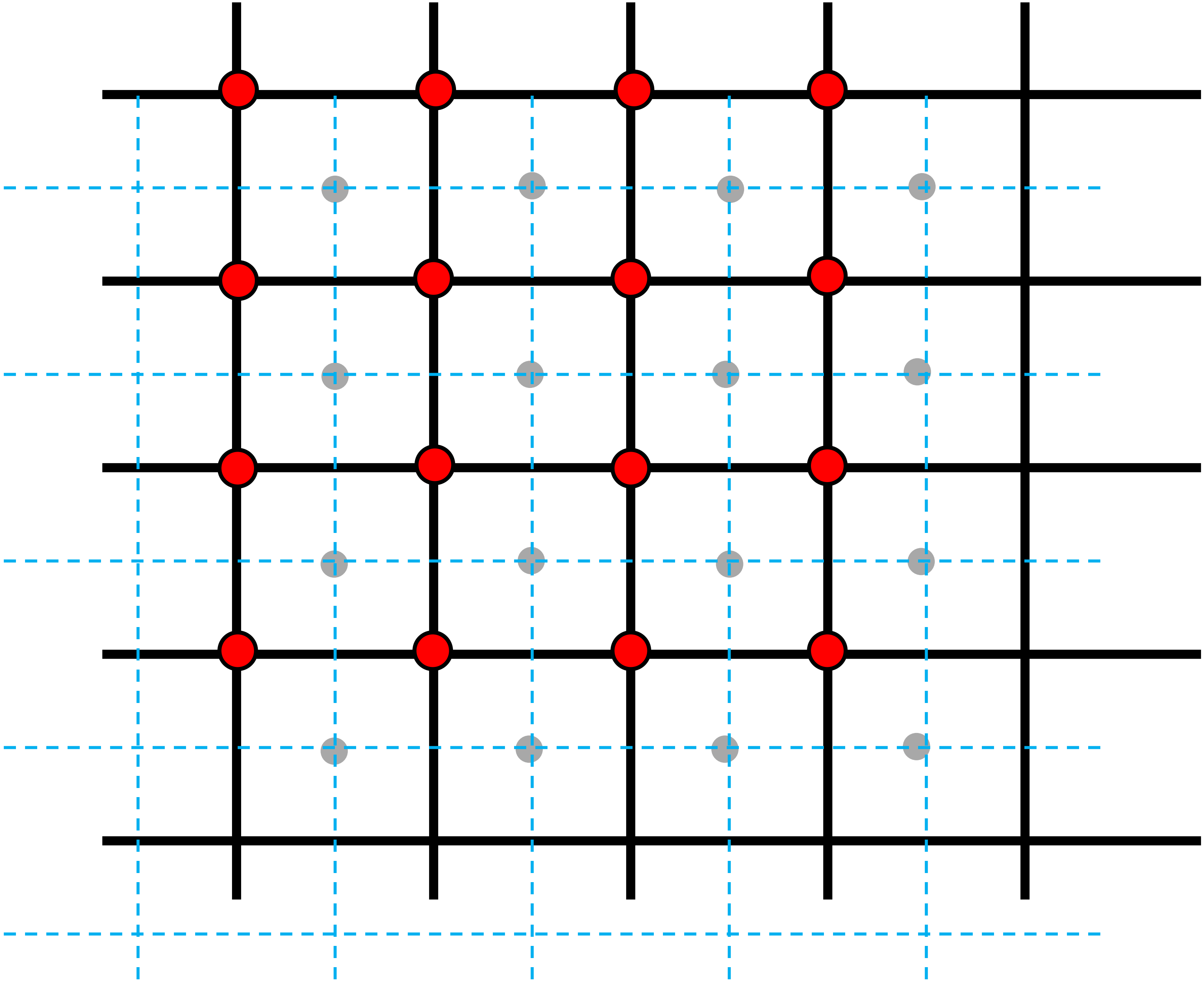
Data Representation

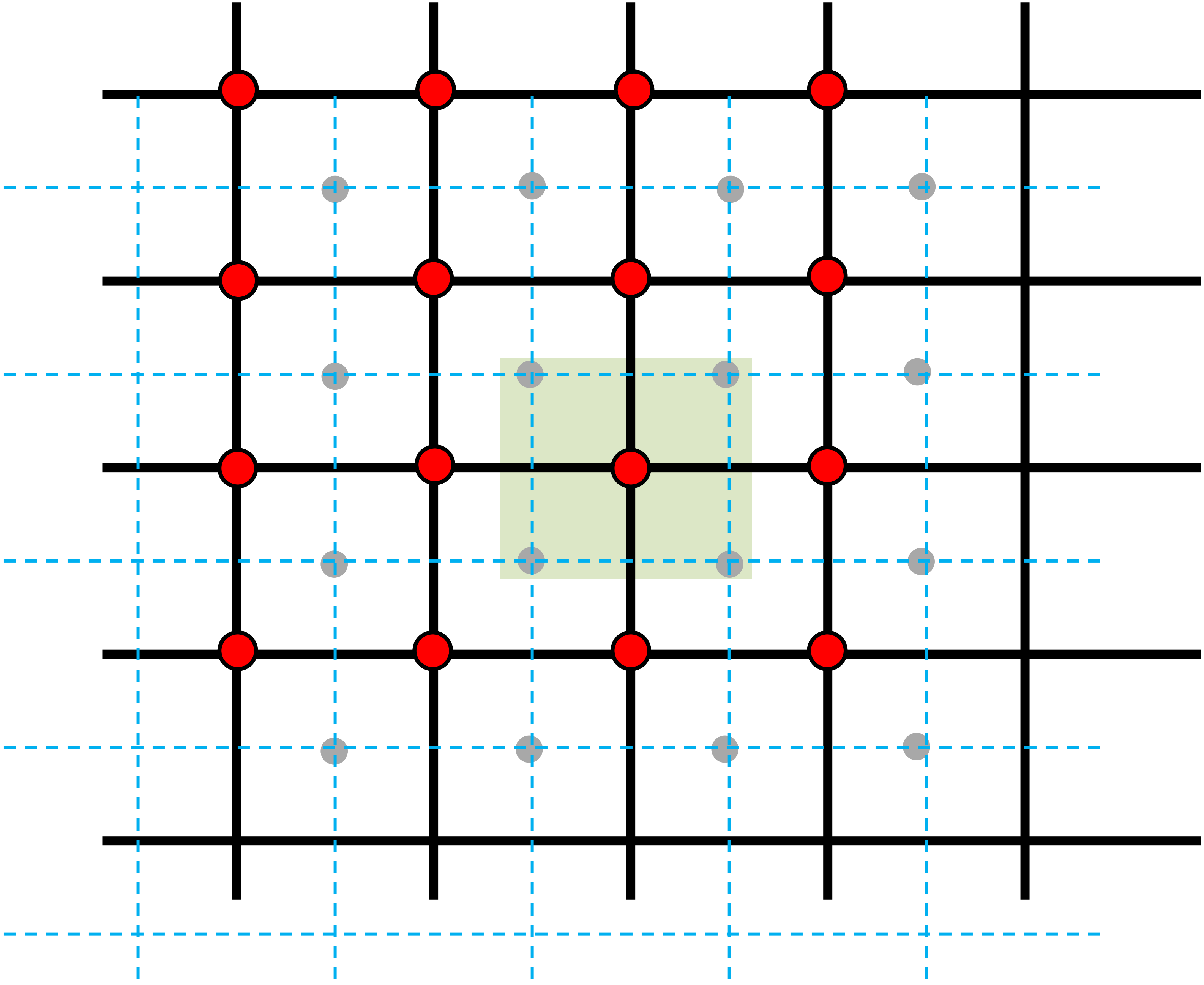
- Volume rendering techniques
  - depend strongly on the **grid type**
  - exist for both structured and unstructured grids
  - are predominantly applied to **uniform grids (3D images).**
  - for uniform grid, <u>**voxels**</u> are the basic unit

- <u>**Cell-centered data for uniform grids**</u>
  - 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.





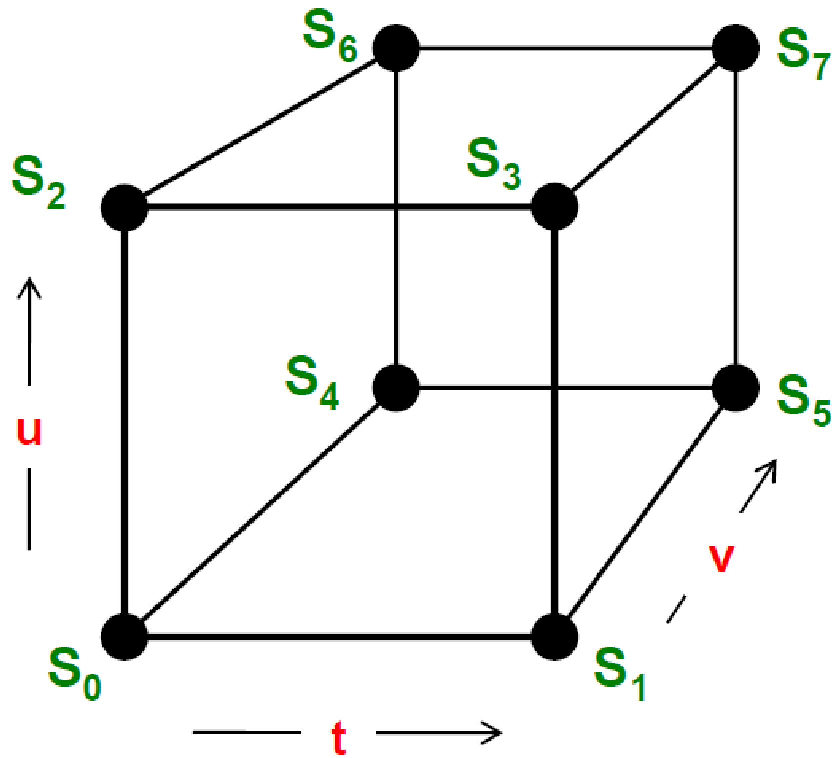




Important Concepts

- 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



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

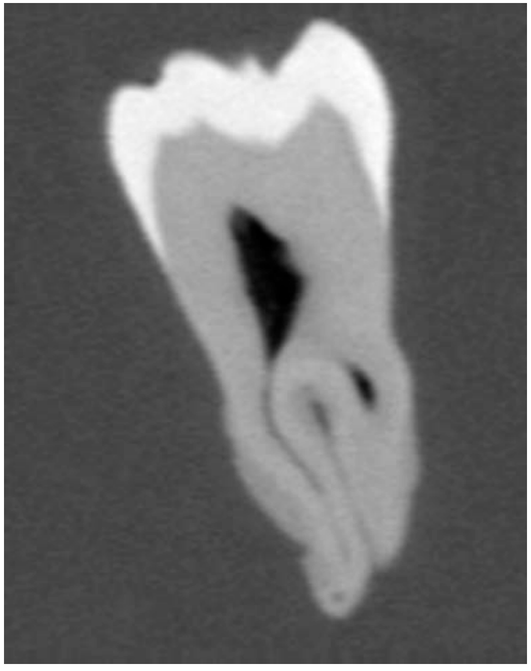
This is useful, for example, if we have passed an oblique cutting plane through a 3D mesh of points and are trying to interpolate scalar values from the 3D mesh to the 2D plane.

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 $N(p) = \mathrm{grad}(f)$
  - Derivatives of f
    - $C(f, \mathrm{grad}(f))$, $\alpha(f, \mathrm{grad}(f))$

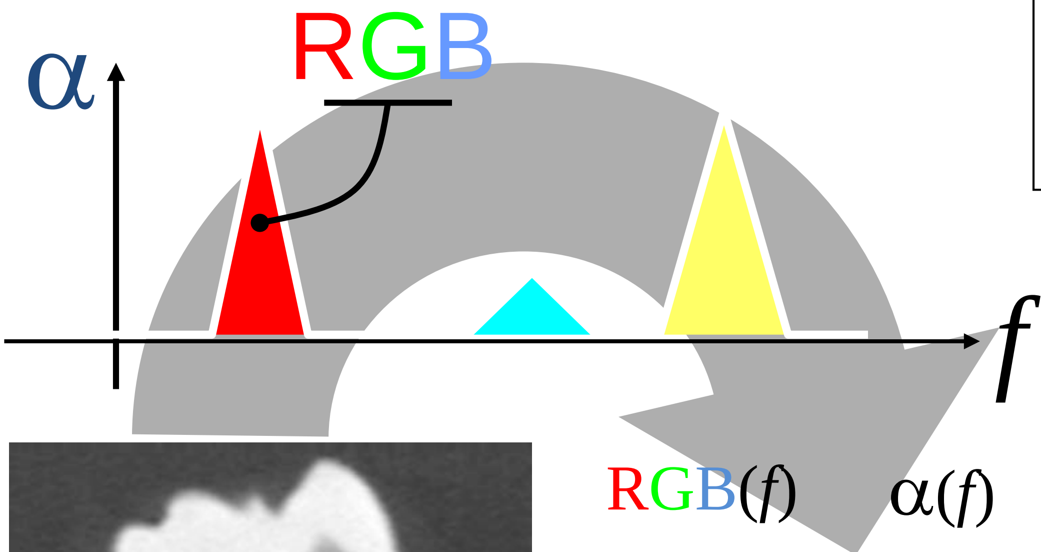
Transfer Functions (TFs)

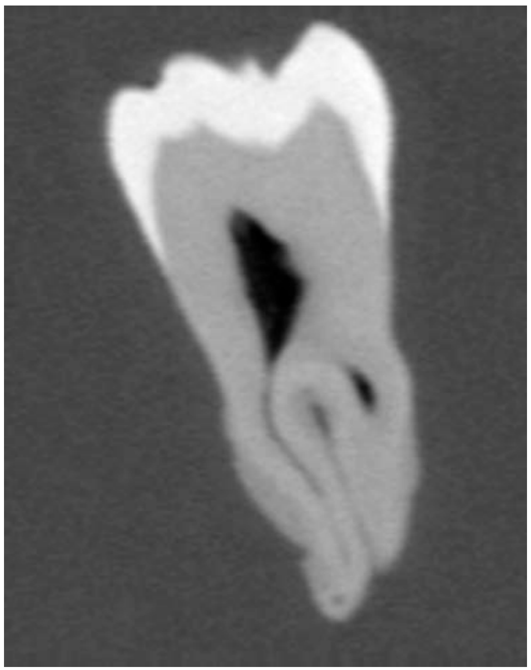


Human Tooth CT

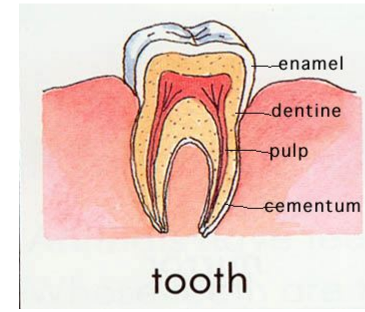
Transfer Functions (TFs)

Map data value f to color and opacity

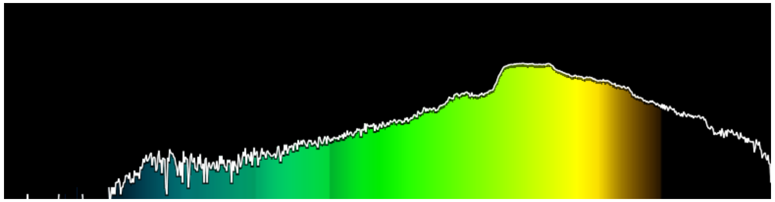




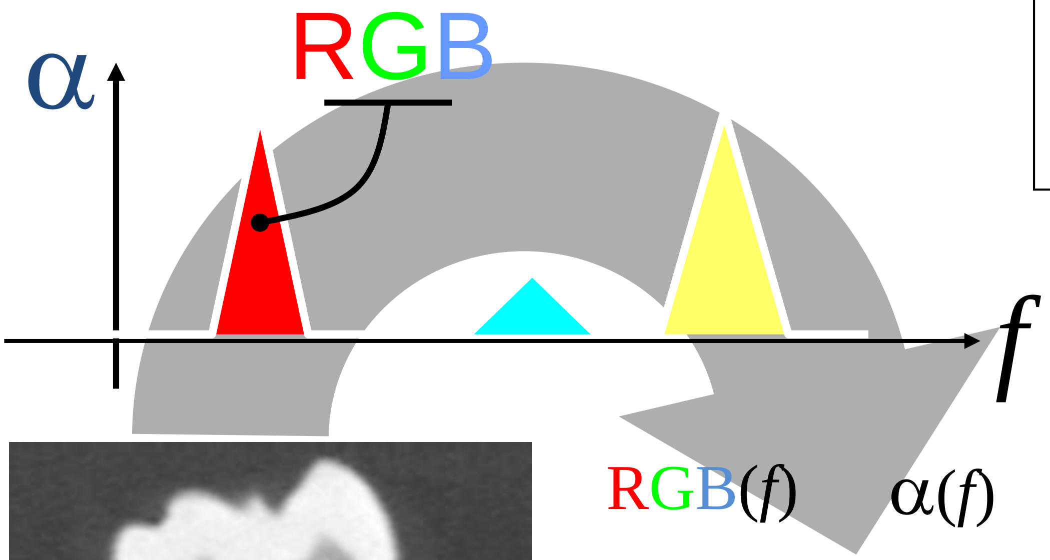
Human Tooth CT

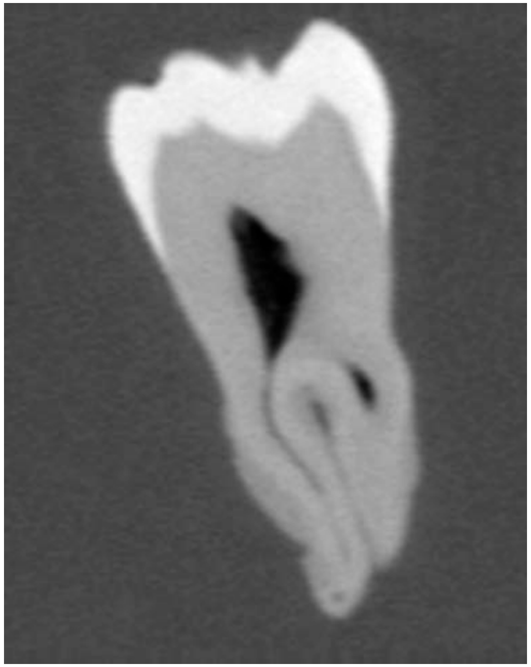


Transfer Functions (TFs)

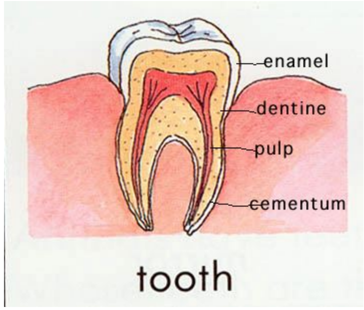


Map data value f to color and opacity

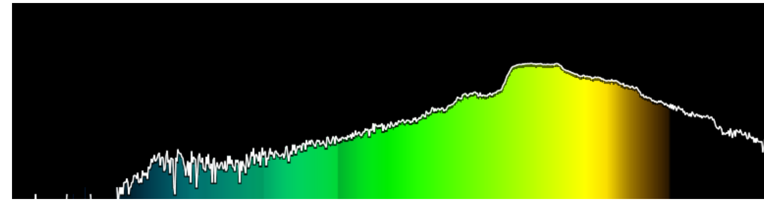




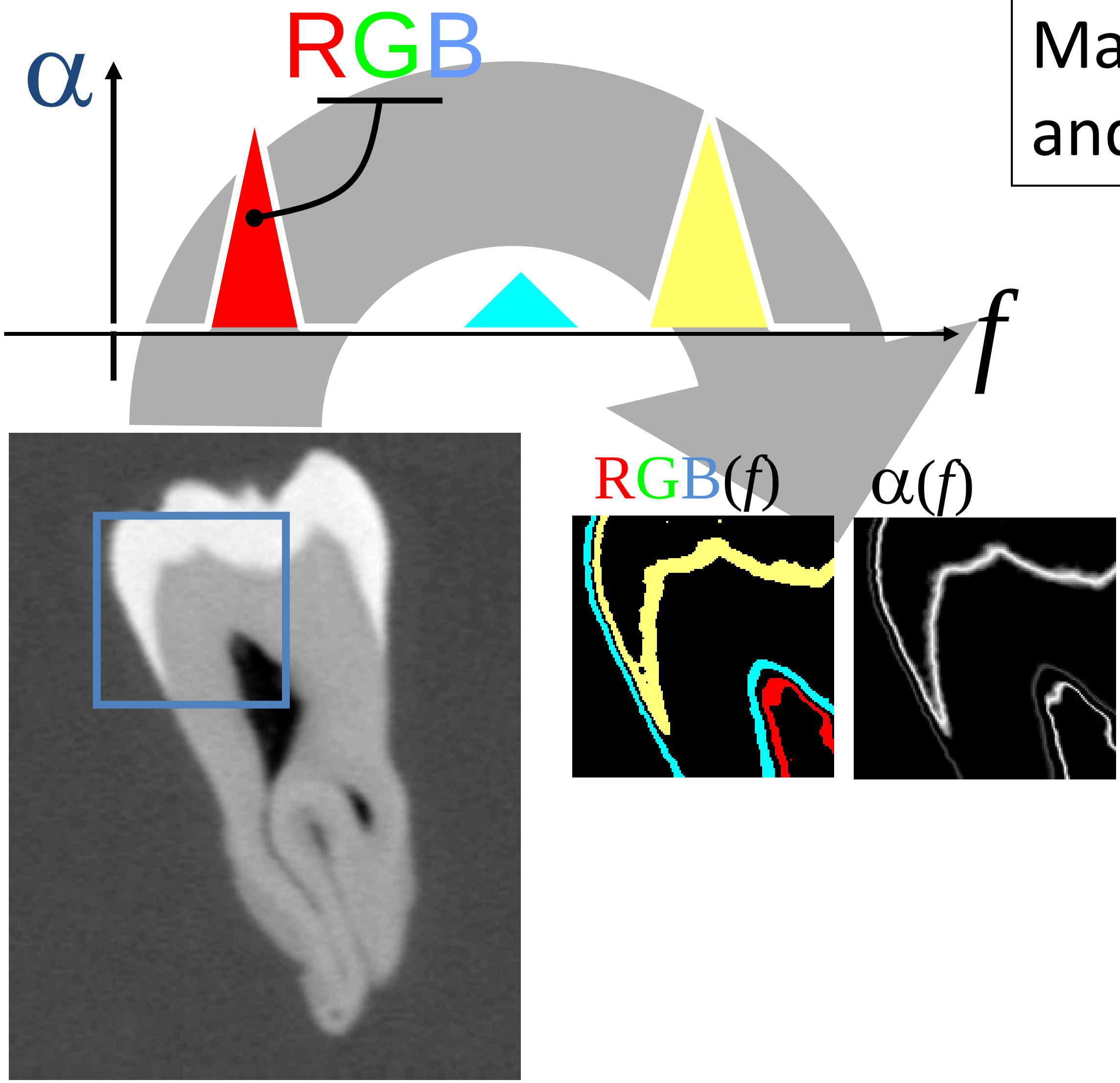
Human Tooth CT



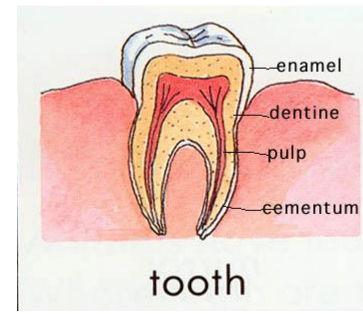
Transfer Functions (TFs)



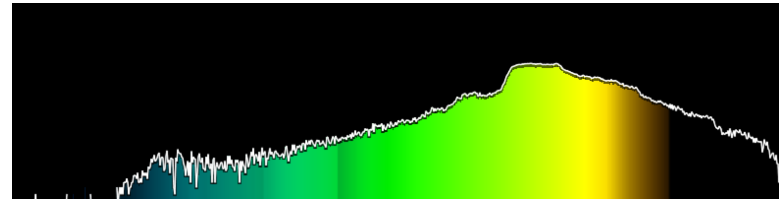
Map data value f to color and opacity



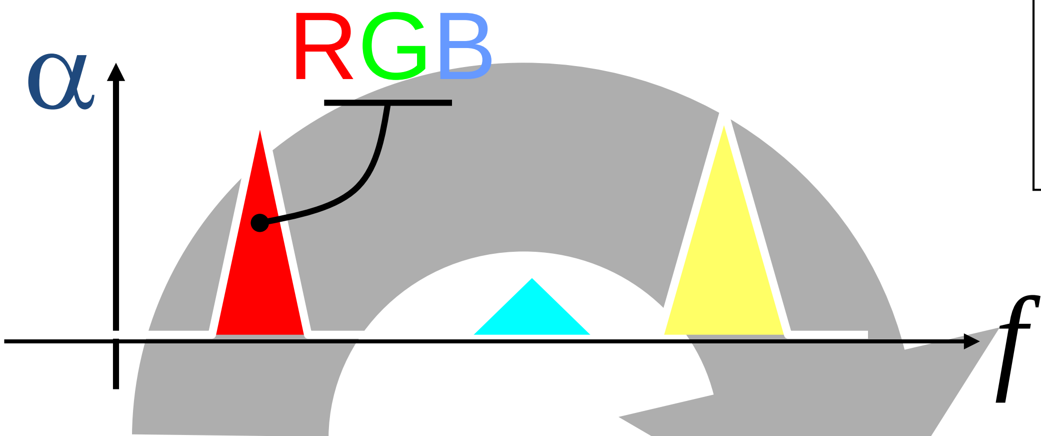
Human Tooth CT

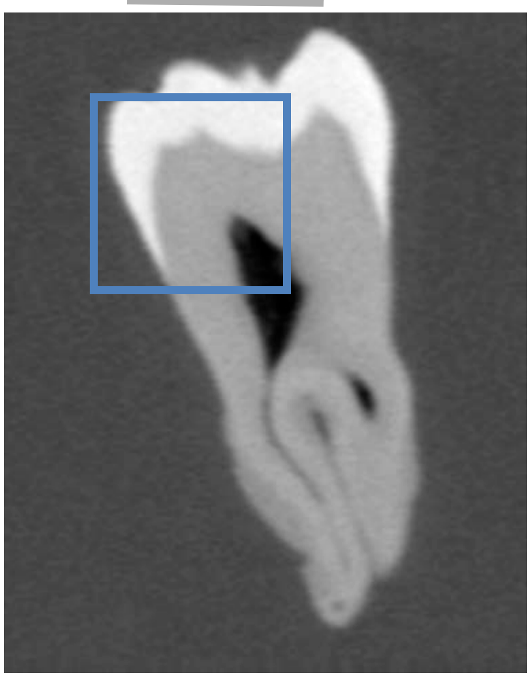


Transfer Functions (TFs)

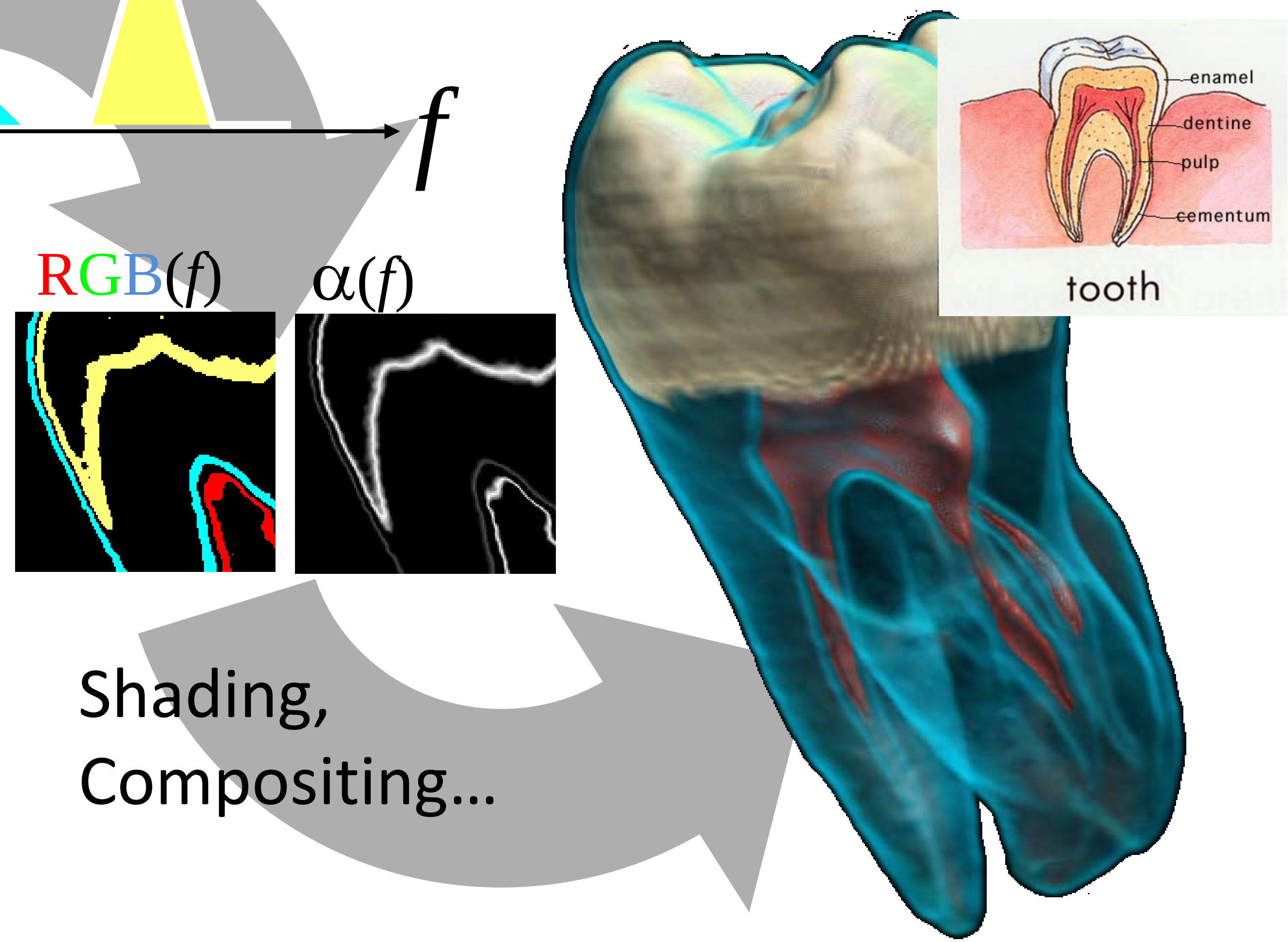


Map data value f to color and opacity





Human Tooth CT



Gradient

∇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,

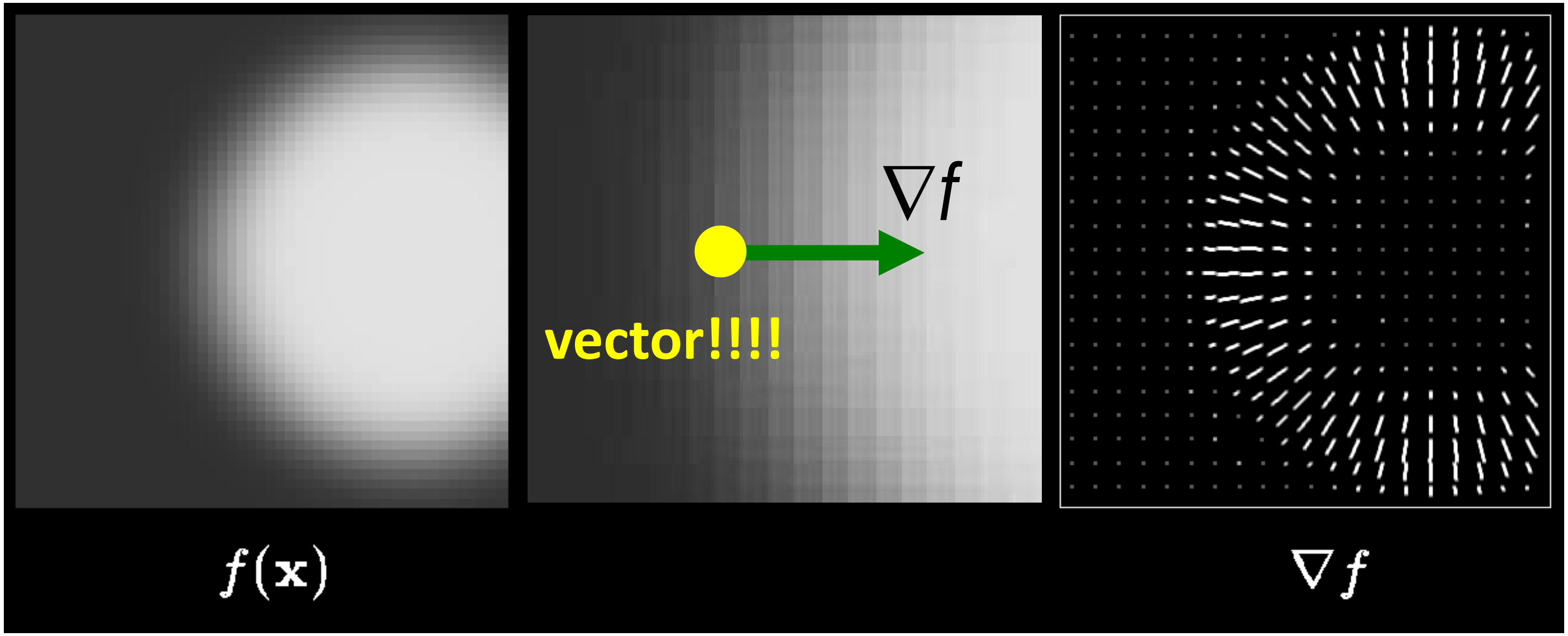
(f(0,0,1) - f(0,0,-1))/2)

Central difference

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

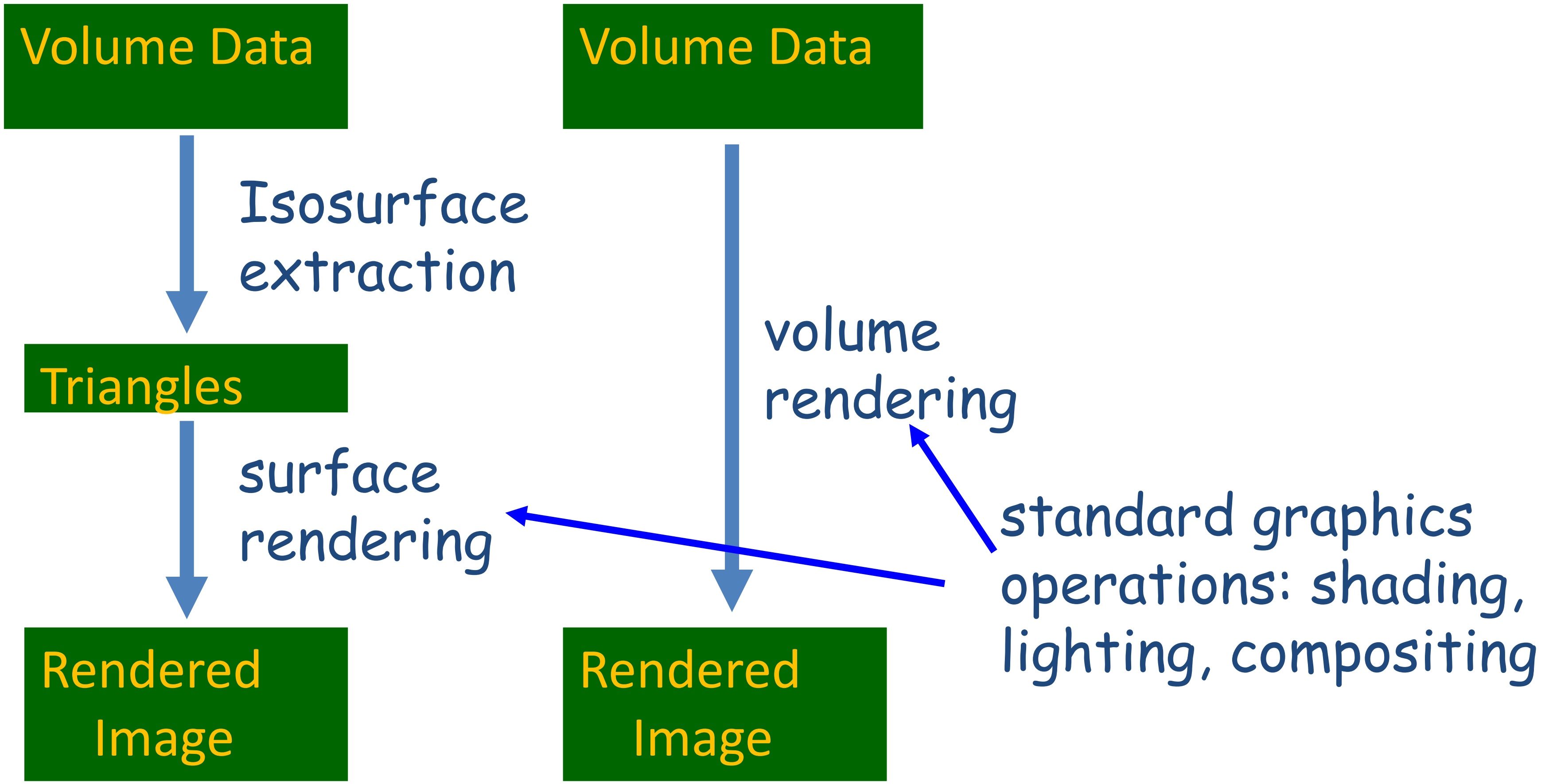
$$\frac{df}{dy} = \frac{f(y+h) - f(y-h)}{2h}$$

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>

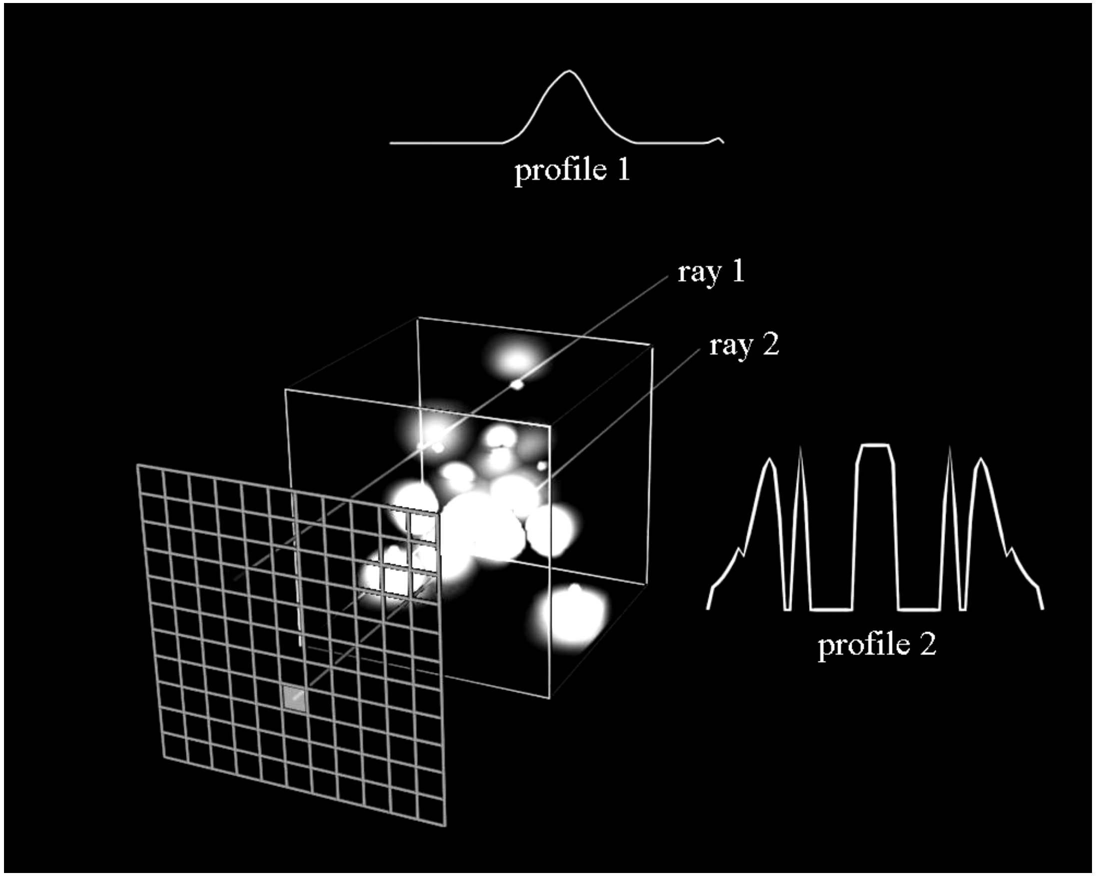
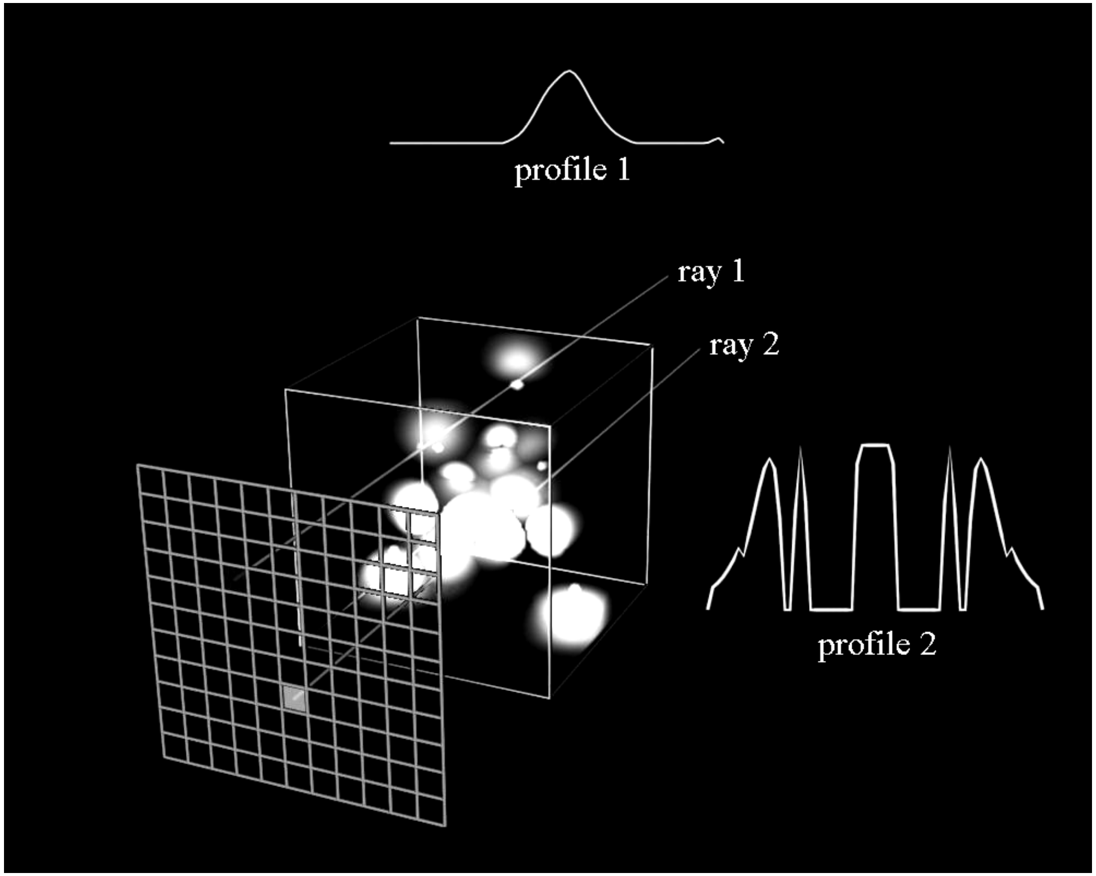


Image Order

- 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

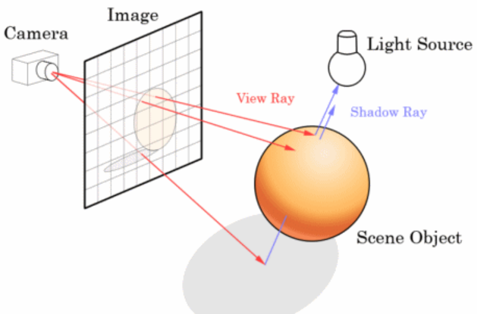


Image source: wikipedia

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

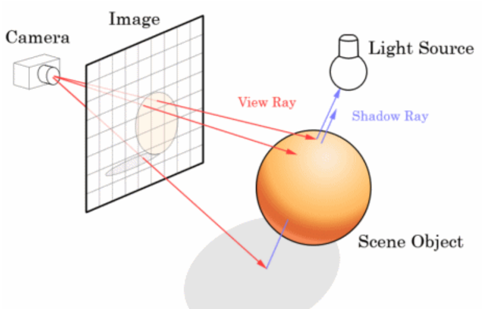
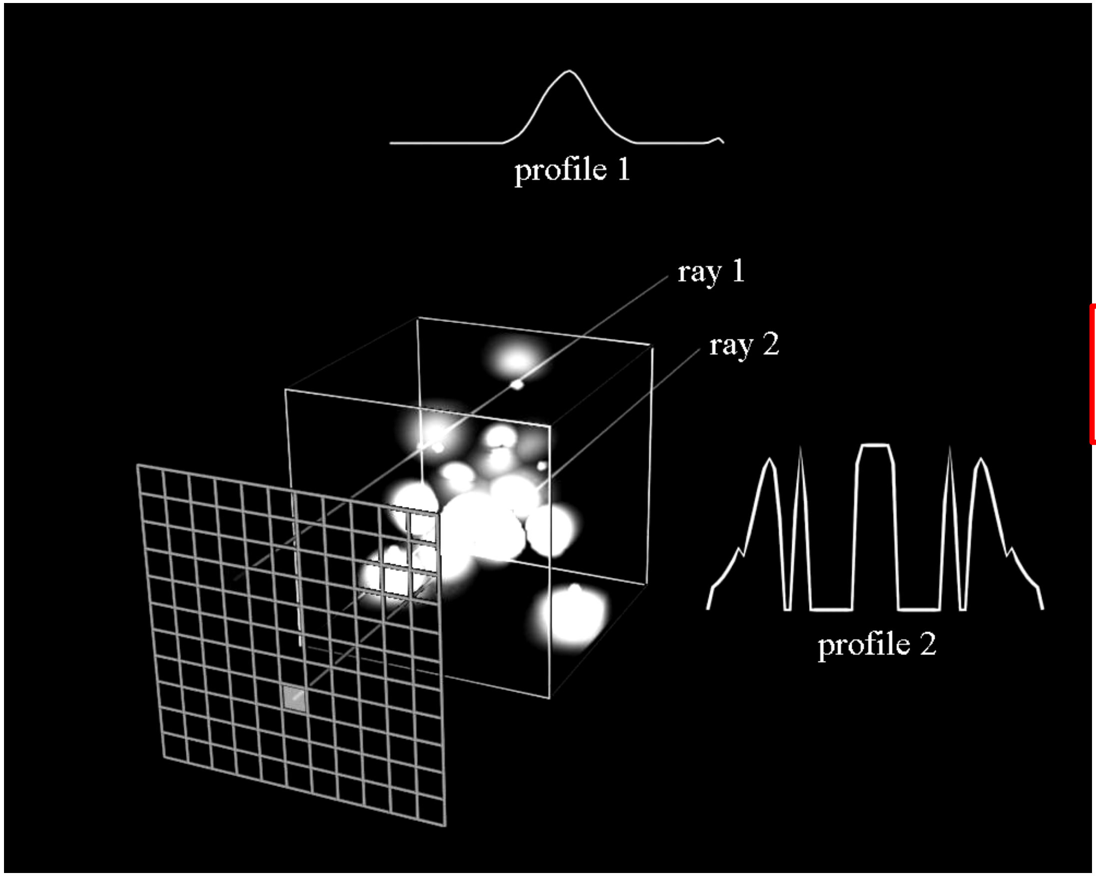


Image source: wikipedia

Image Order

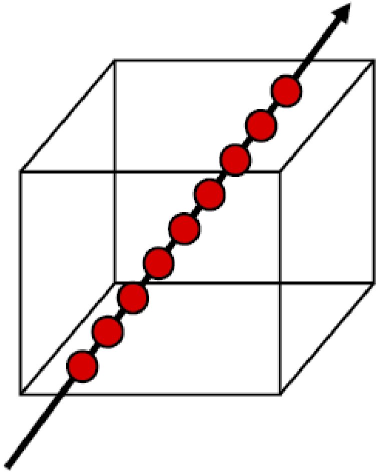
- Render image <u>one pixel at a time</u>



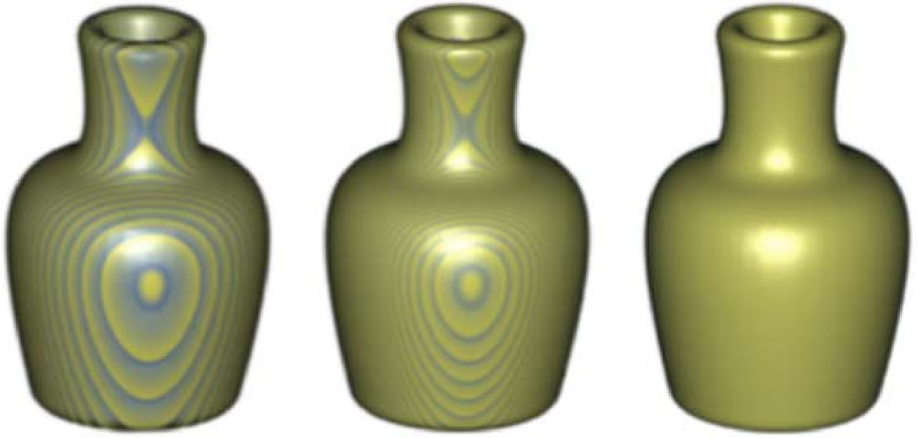
For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Raycasting

Sampling interval can be fixed or adjusted to voxels:



uniform sampling

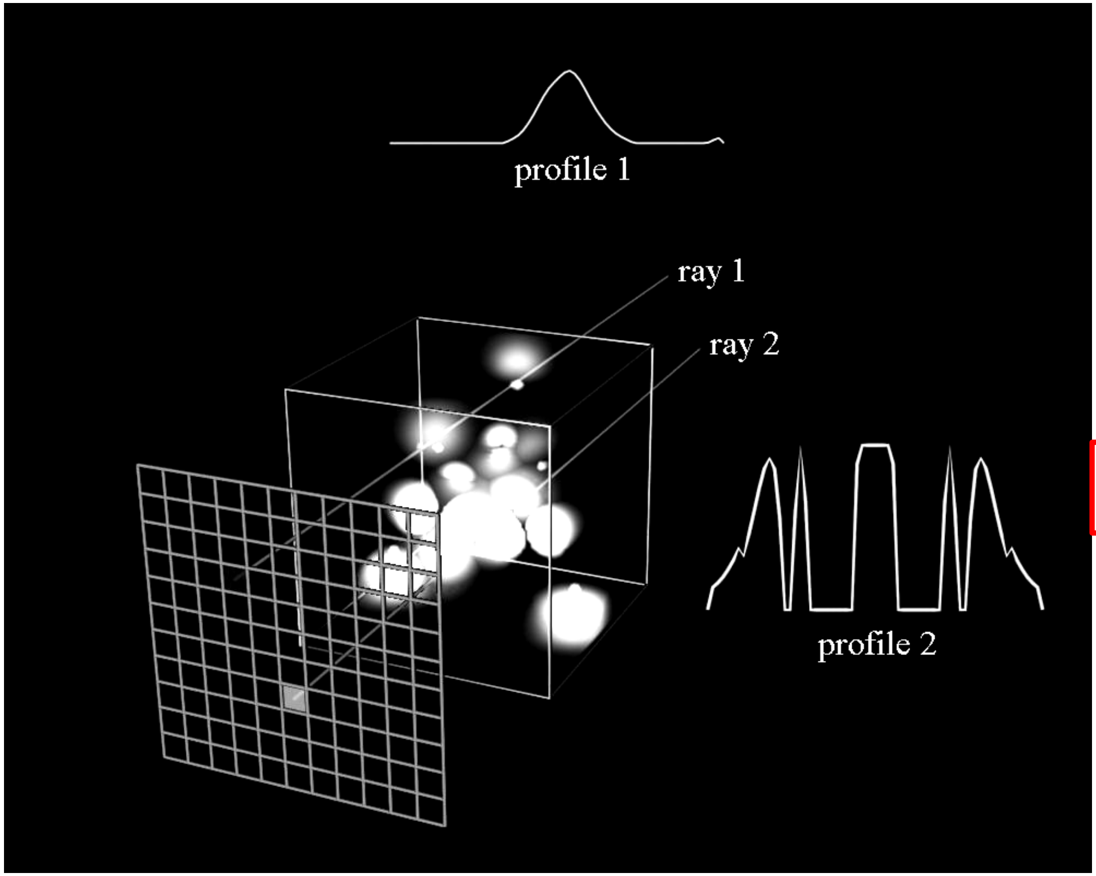


2.0 1.0 0.1

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

Image Order

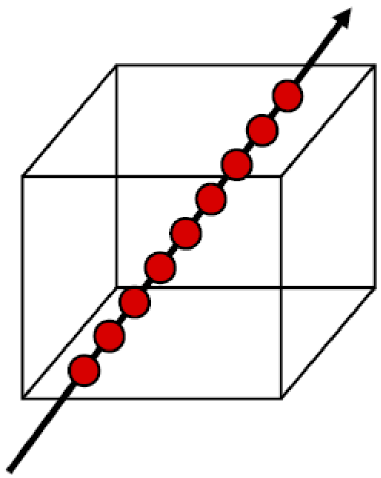
- Render image <u>one pixel at a time</u>



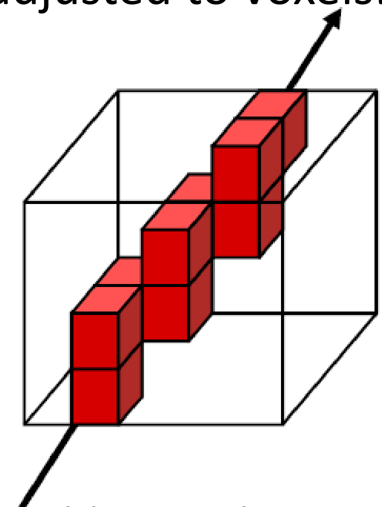
For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Raycasting

Sampling interval can be fixed or adjusted to voxels:



uniform sampling



voxel-by-voxel traversal
(faster)

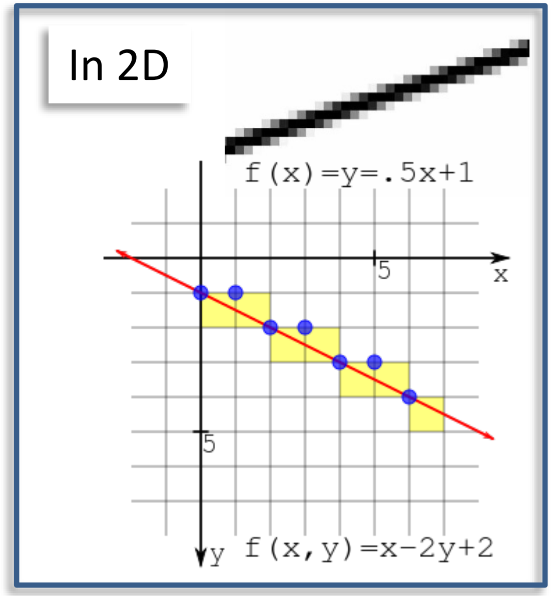
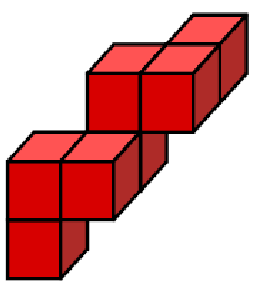
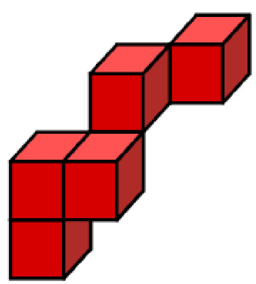


Image source: wikipedia

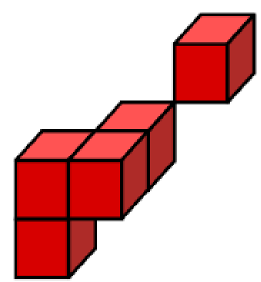
Connectedness of "voxelized" rays:



6-connected
(strongest)



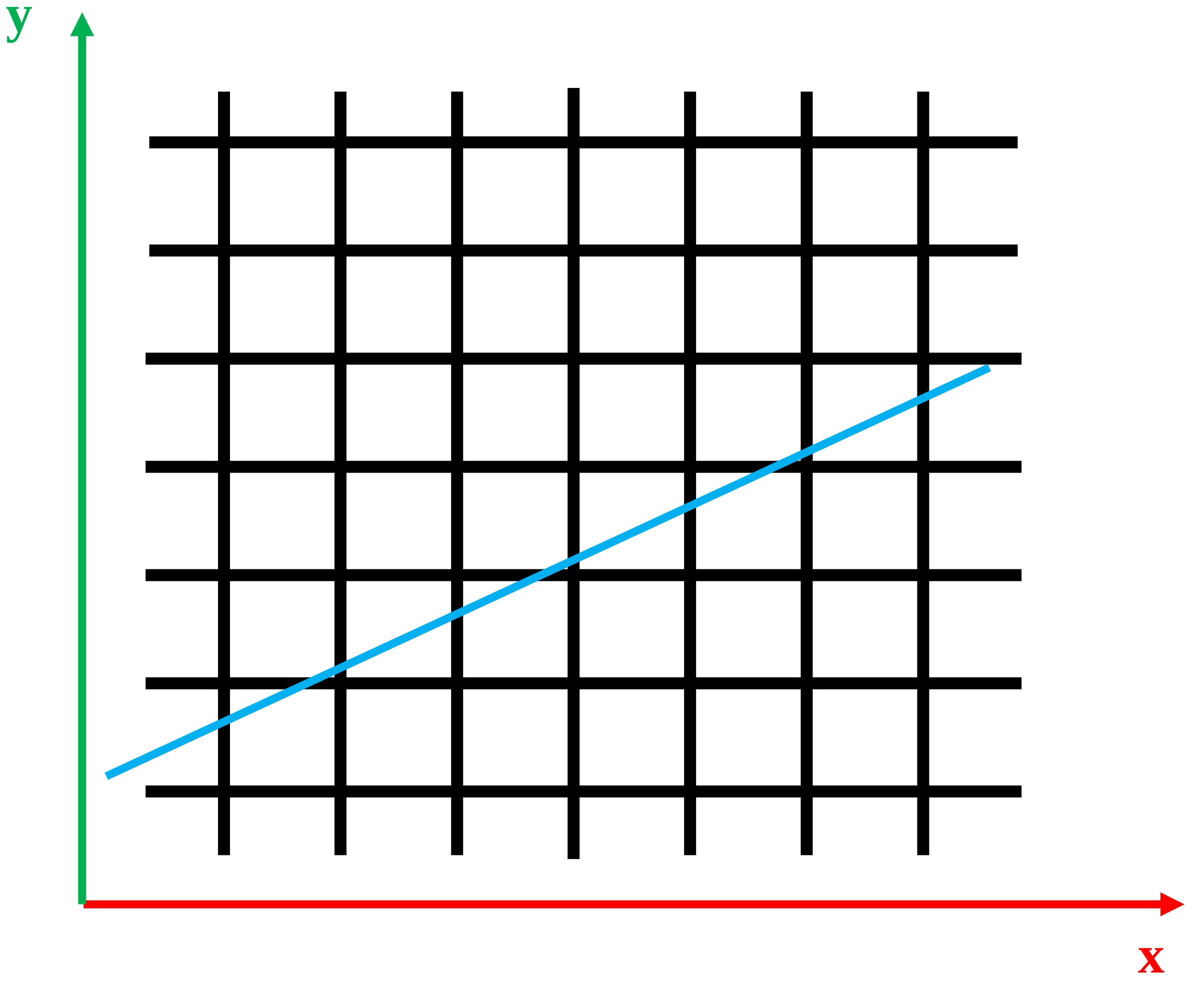
18-connected



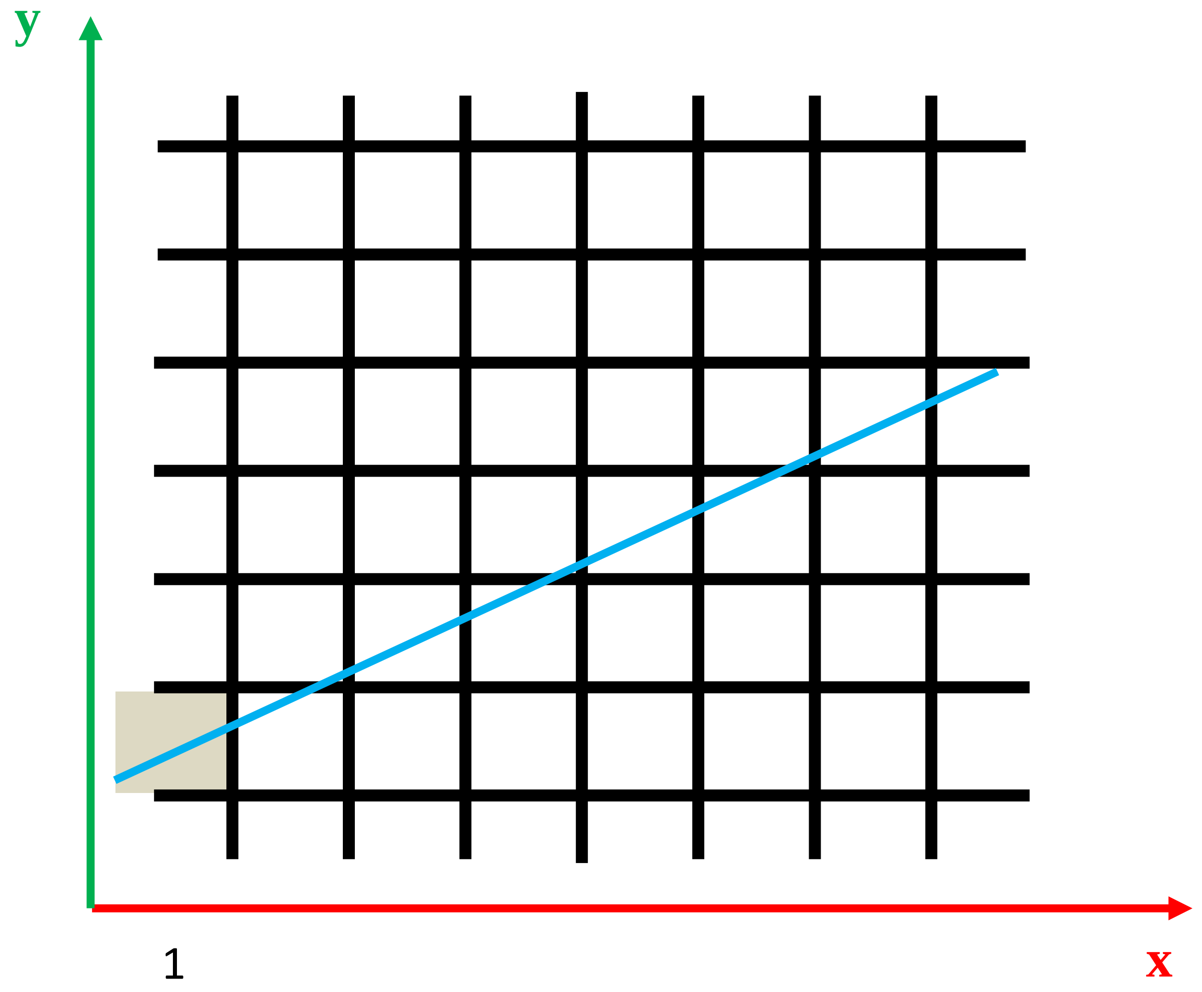
26-connected
(weakest)

Accelerate the sampling and interpolation

Line rasterization process

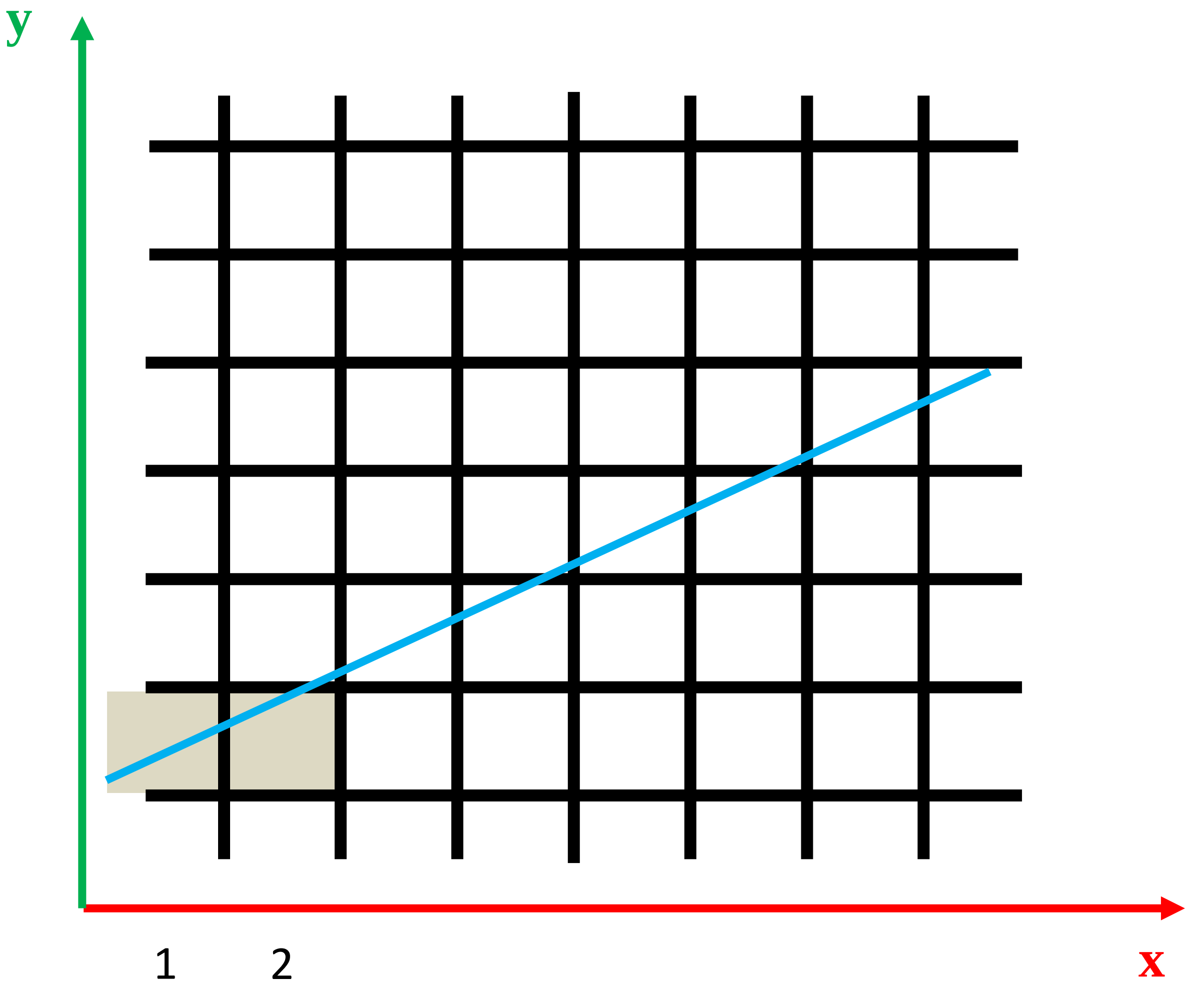


Line rasterization process



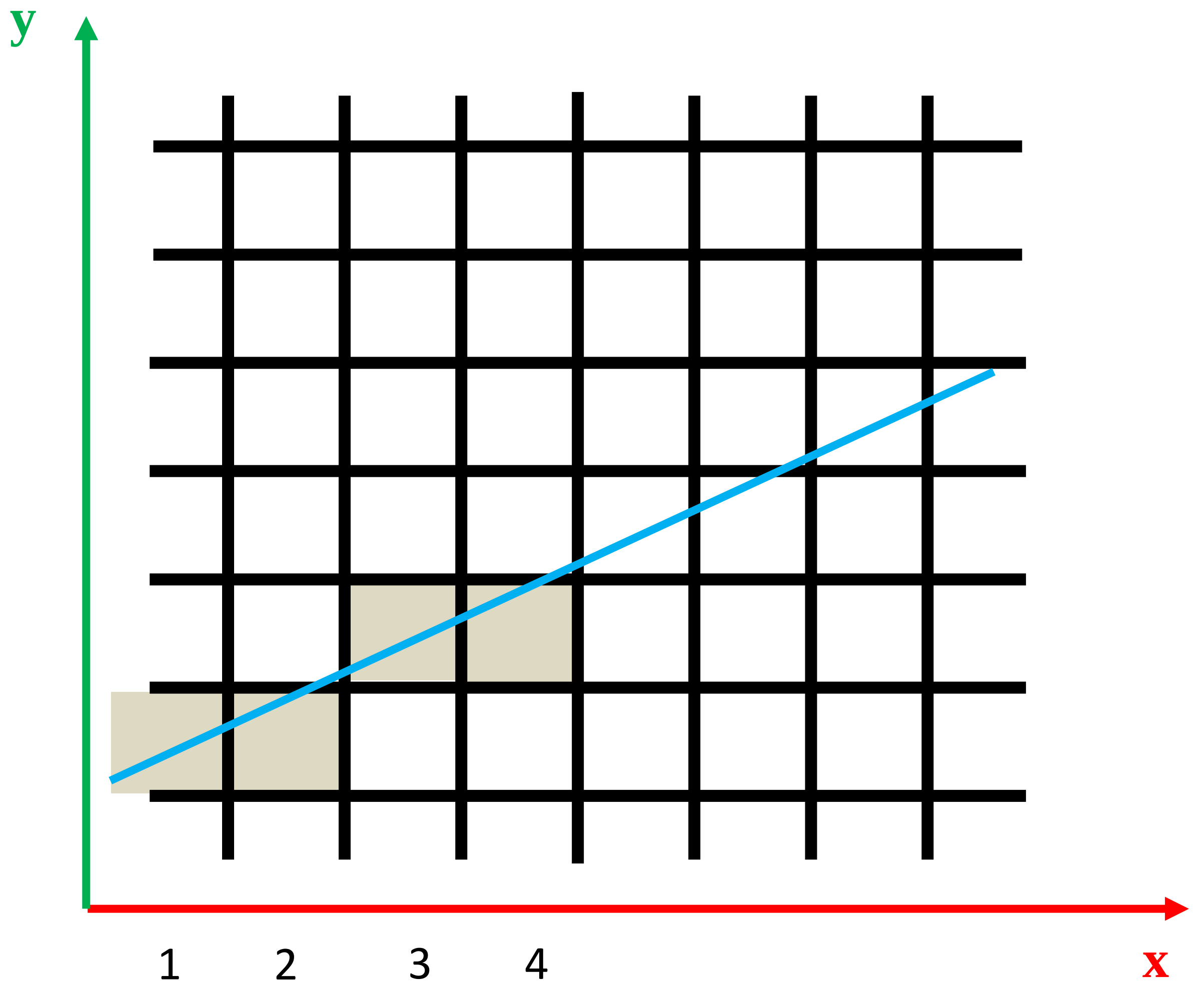
Let us increase along x

Line rasterization process



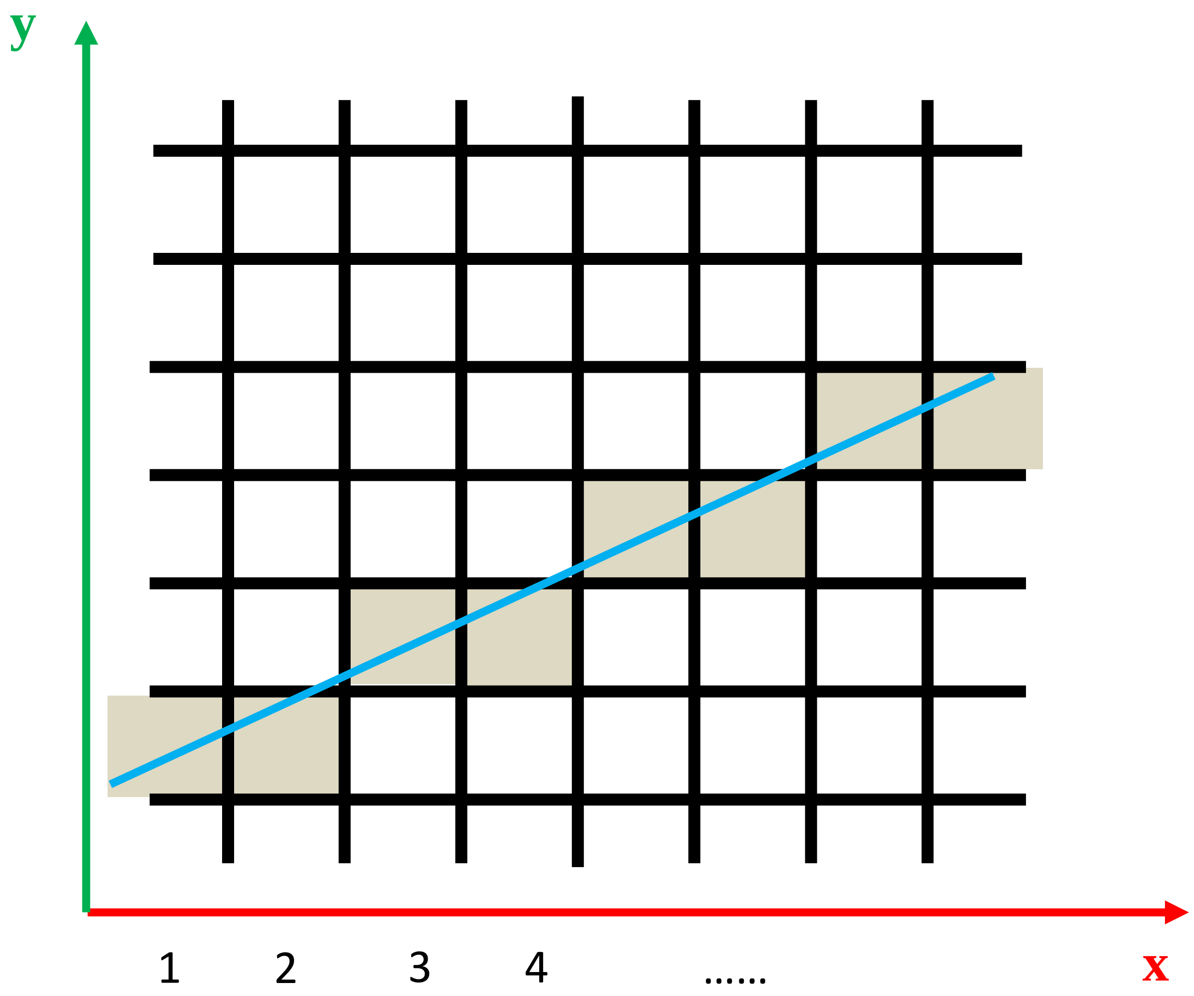
Let us increase along x

Line rasterization process



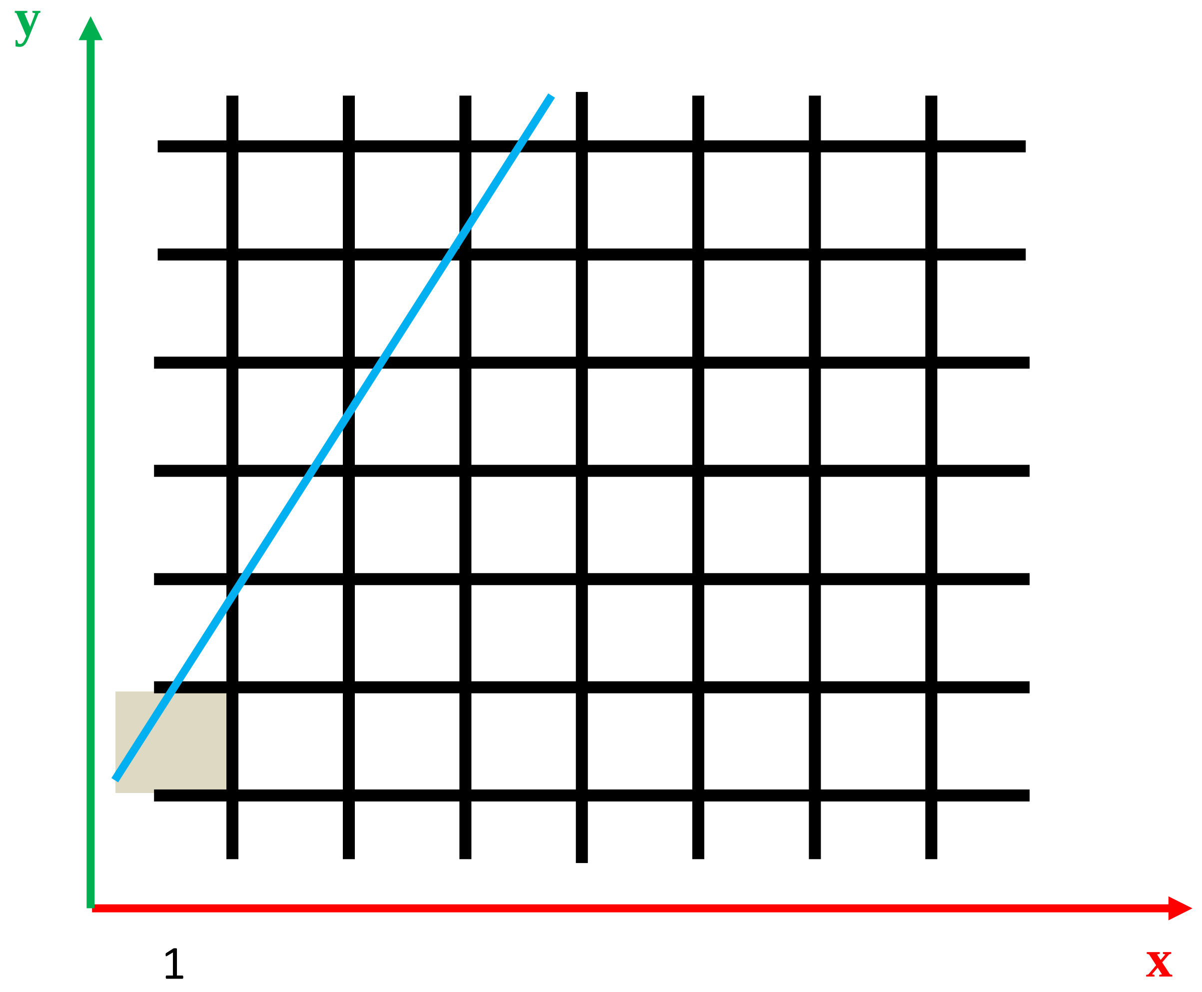
Let us increase along x

Line rasterization process



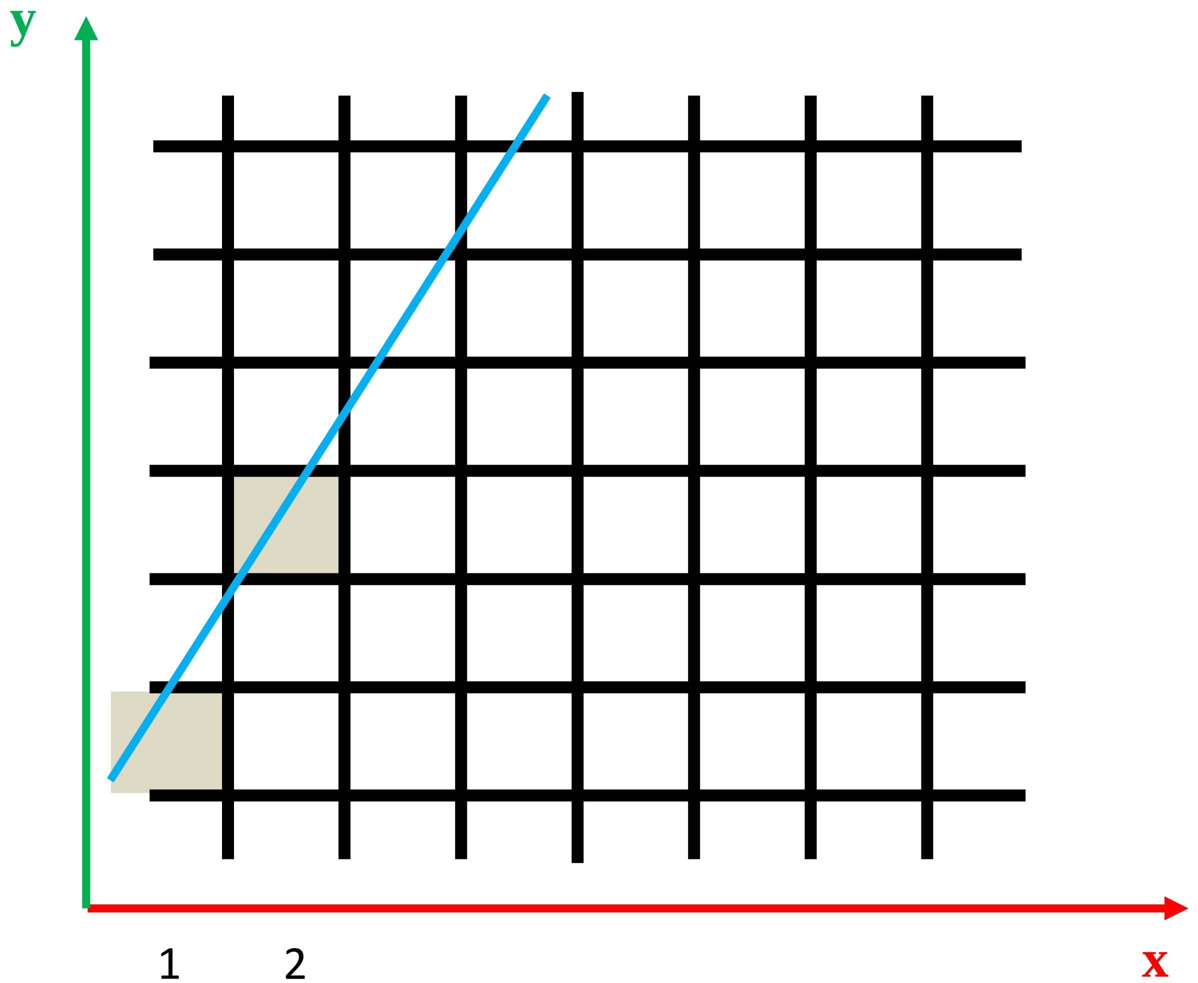
Let us increase along x

Line rasterization process



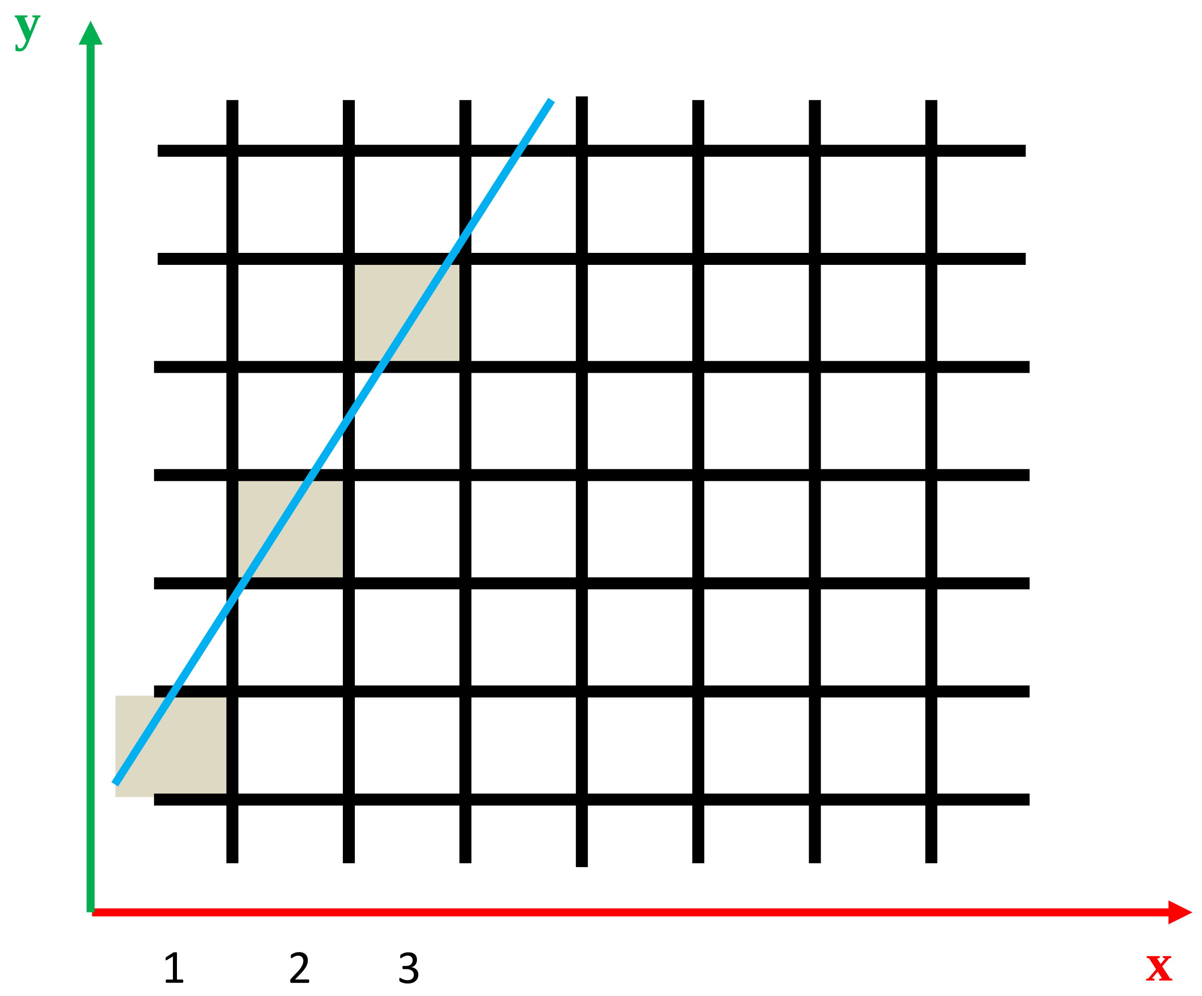
If we still increase along x, what will happen?

Line rasterization process



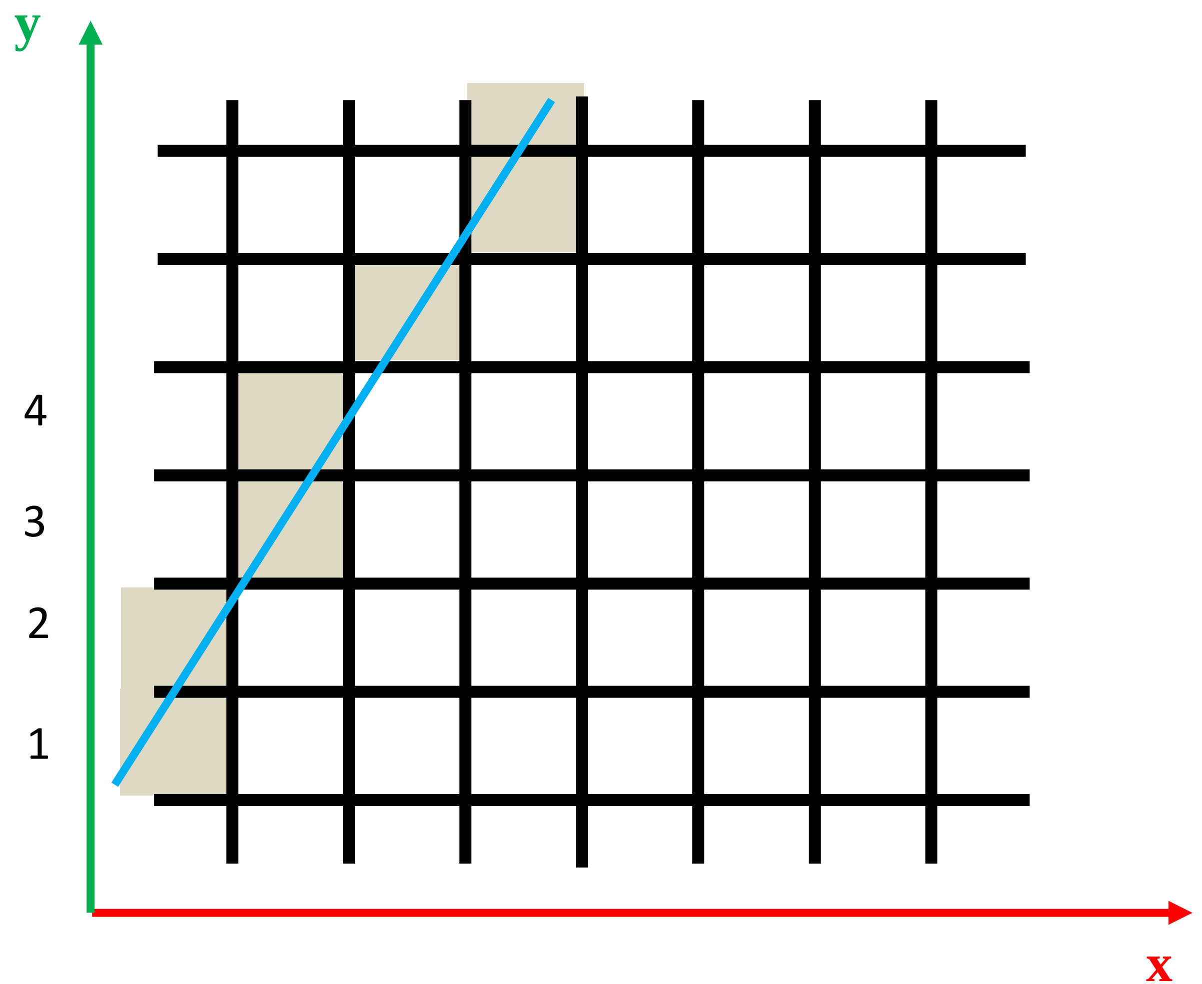
If we still increase along x, what will happen?

Line rasterization process



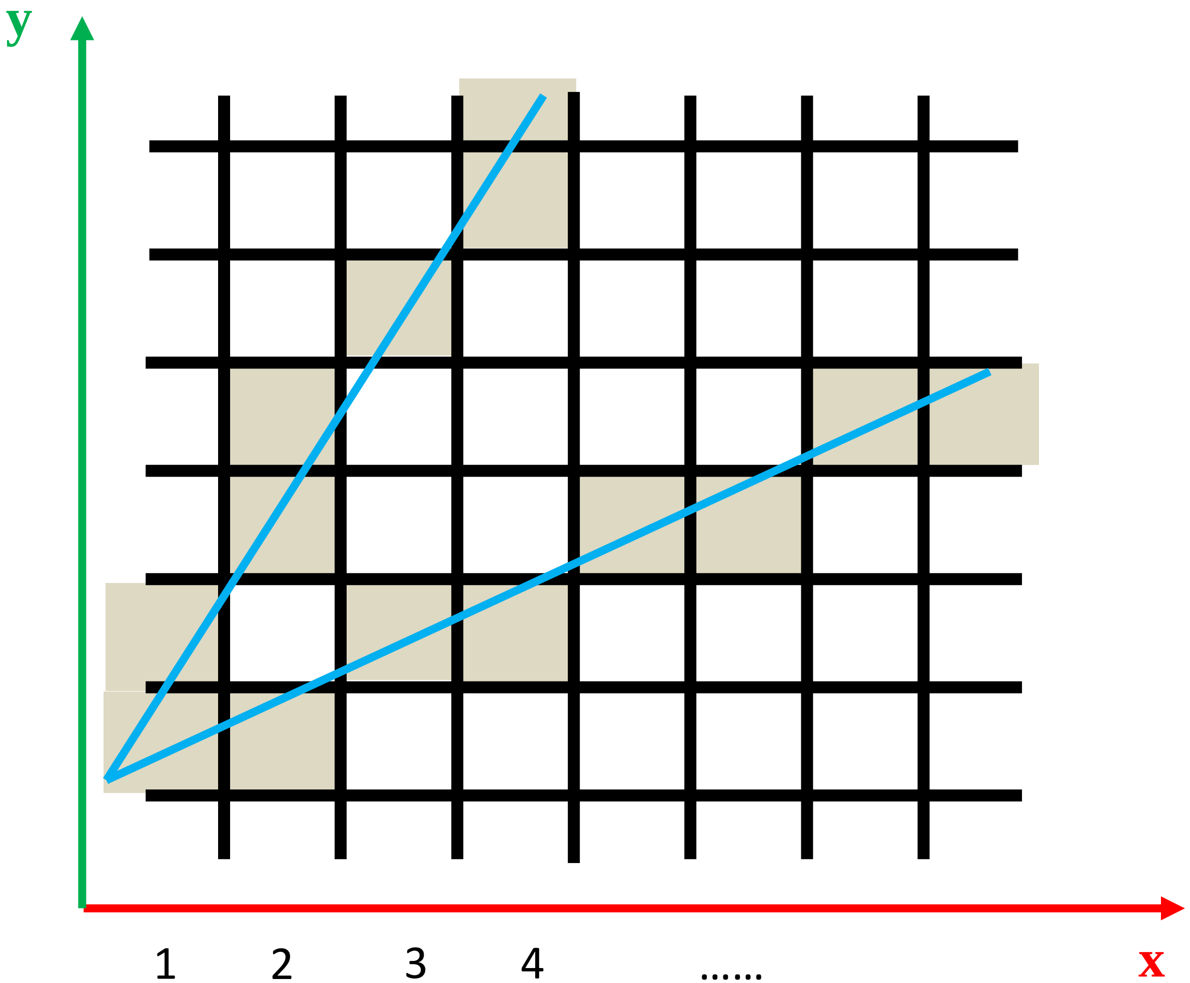
There will be many gaps between the pixels!!!

Line rasterization process



The correct way is to increase along y!!!

Line rasterization process

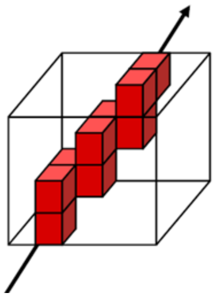


Lesson learned? We have to march along the axis that is most parallel to the line!

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

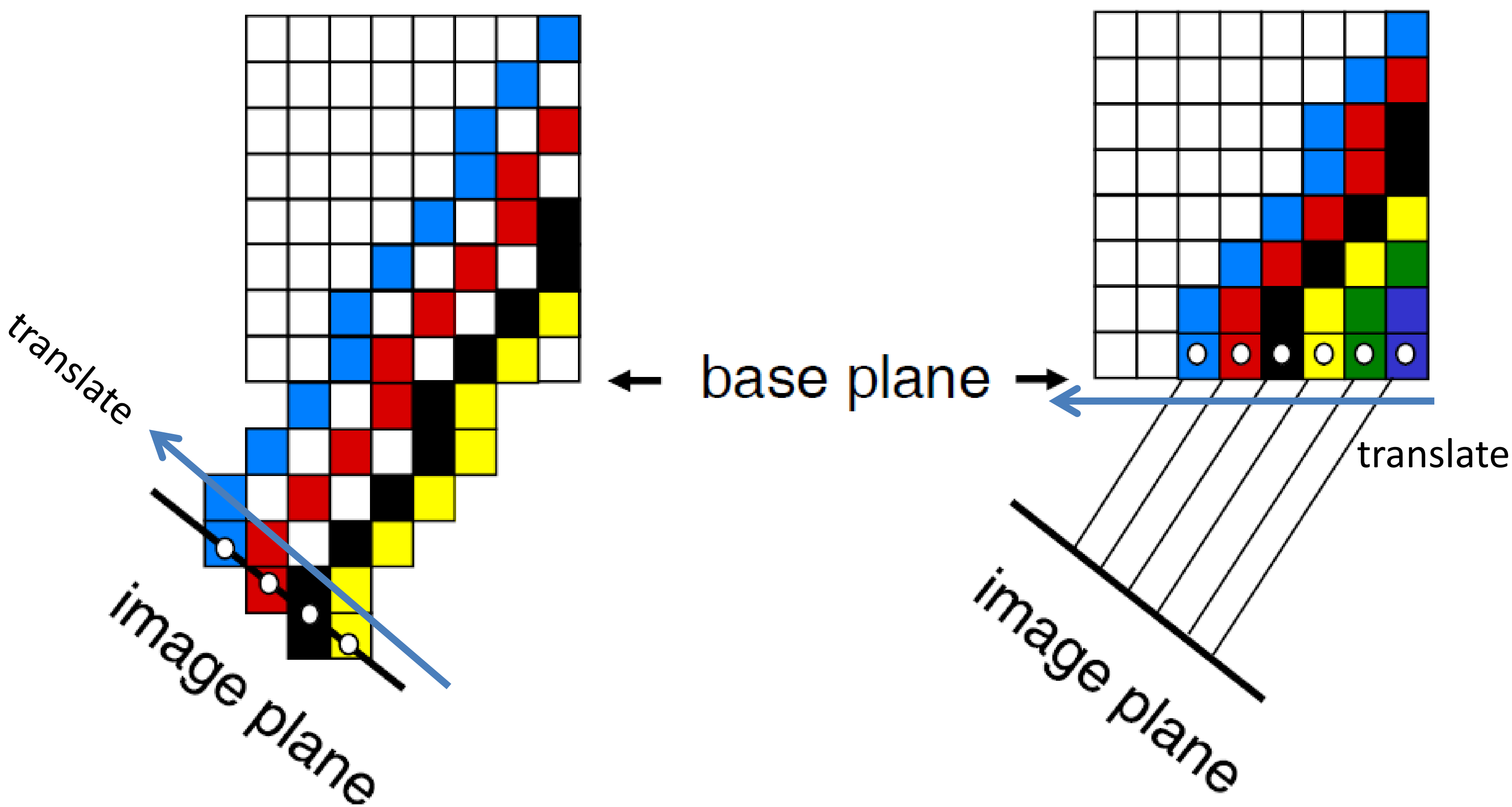
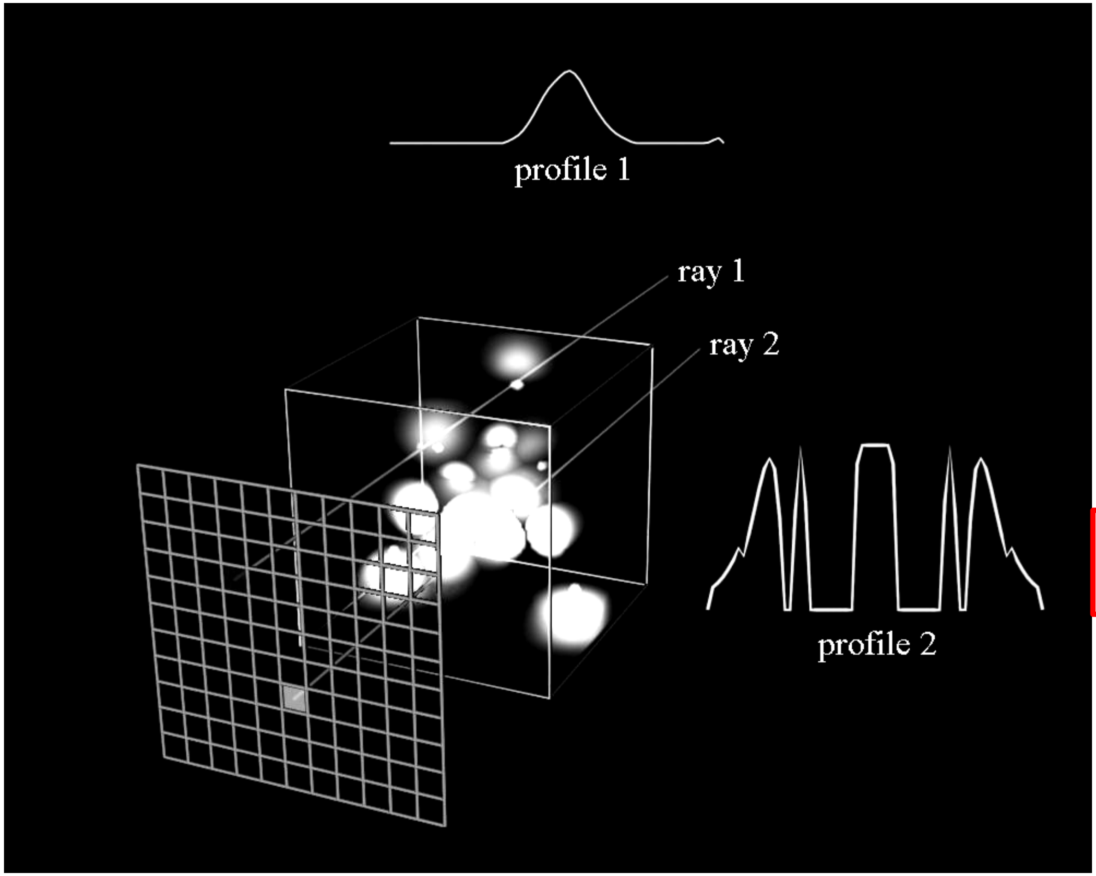


Image Order

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

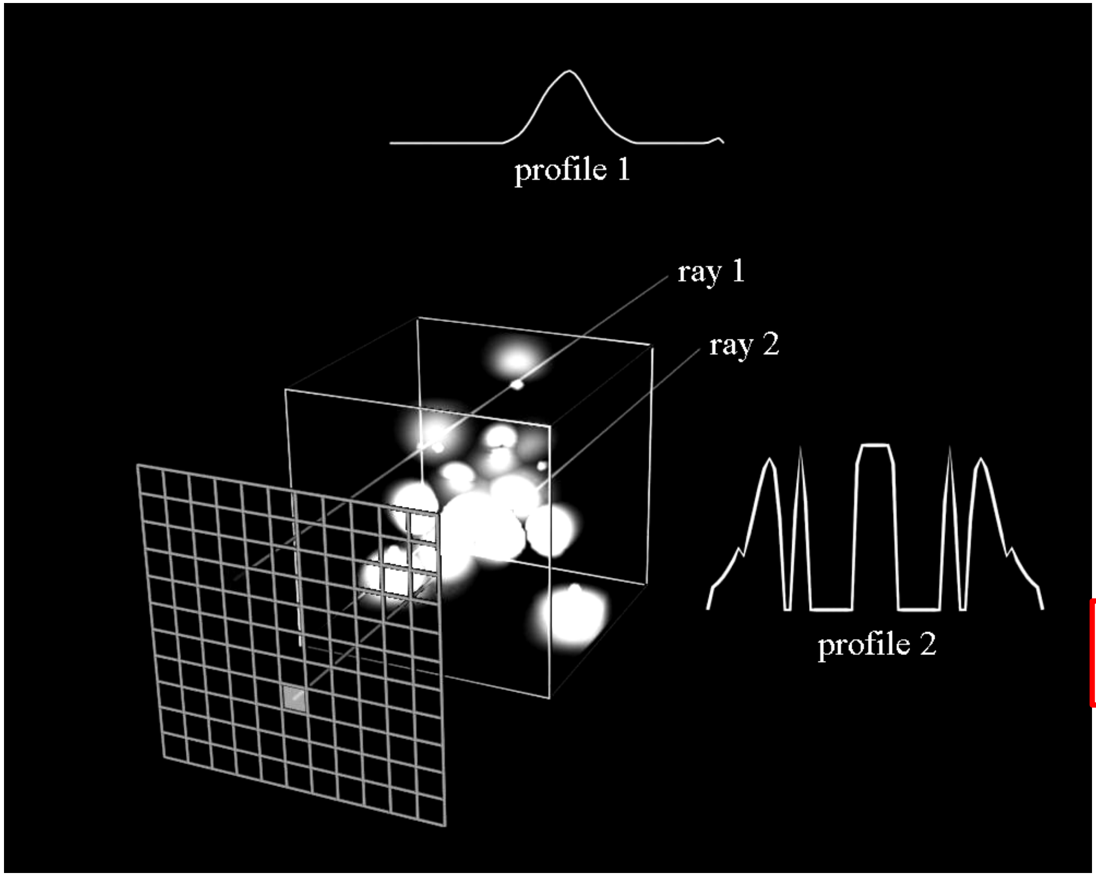


For each pixel …
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Important topic for the later

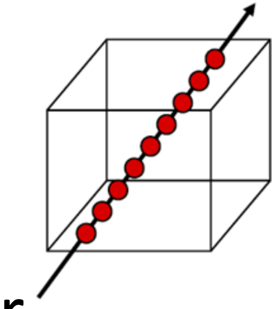
Image Order

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



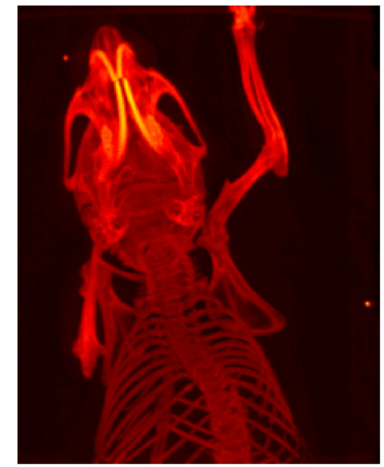
For each pixel ...
- cast ray
- sampling along ray
- interpolate
- get colors/opacity
- composite

Compositing



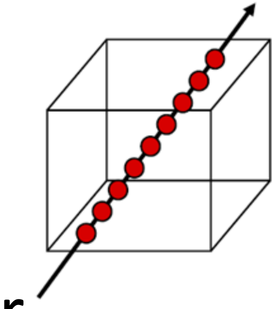
Two simple compositing functions can be used for **previewing**:

- **Maximum intensity projection (MIP)**:
  - maximum of sampled values
  - result resembles X-ray image

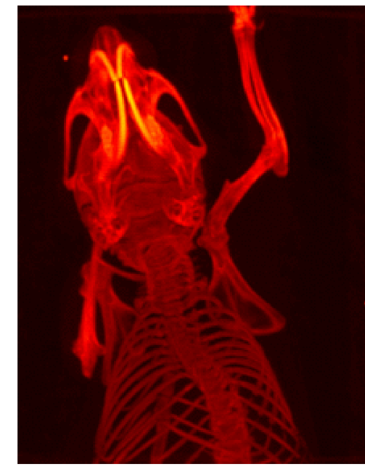


Source: wikipedia

Compositing

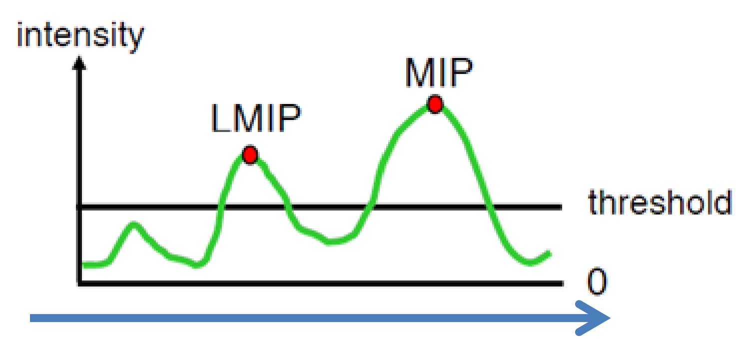


Two simple compositing functions can be used for **previewing**:



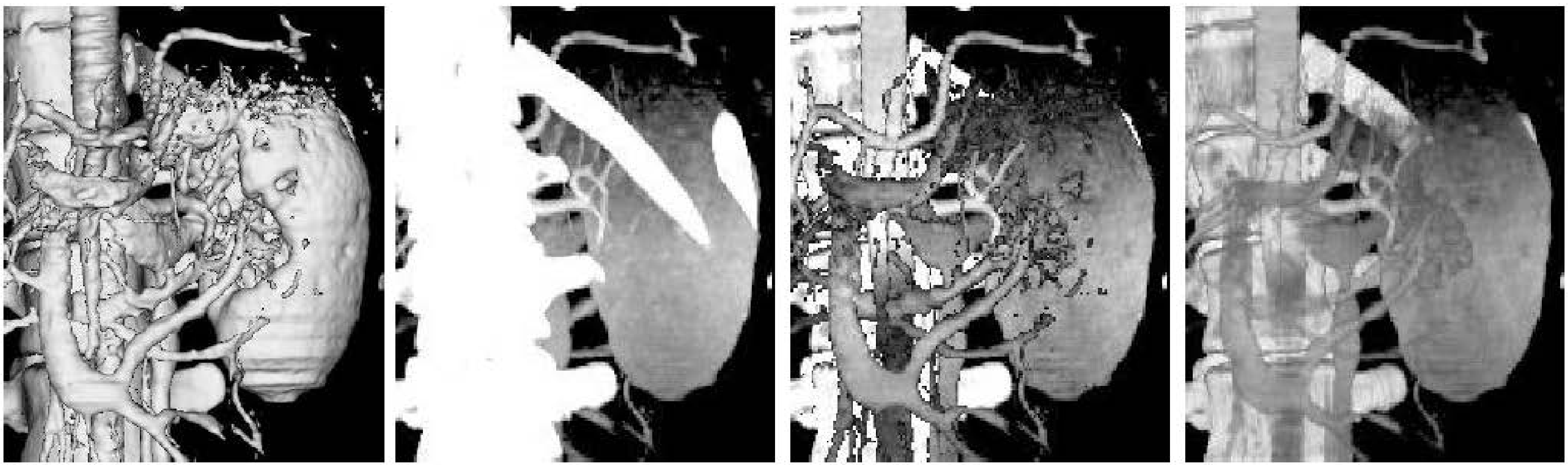
Source: wikipedia

- **Maximum intensity projection (MIP)**:
  - maximum of sampled values
  - result resembles X-ray image
- **Local maximum intensity projection (LMIP)**:
  - first local maximum which is above a prescribed threshold
  - approximates occlusion
  - faster & better(!)



Compositing

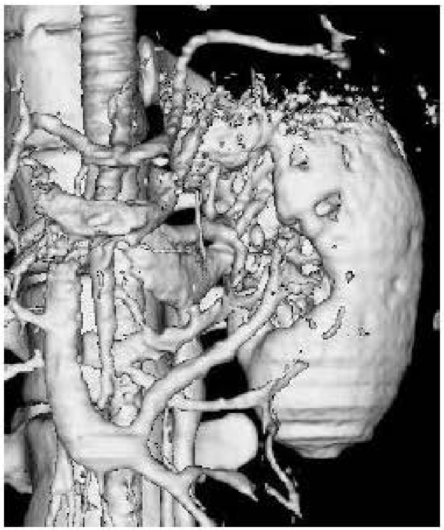
Comparison of techniques (Y. Sato, dataset of a left kidney):
Iso-surface vs. raycasting with **MIP**, **LMIP**, **α-compositing**



fast	fast	fast	-
(1 parameter)	parameter free	(1 parameter)	-
-	full data range	full data range	full data range
-	noise insensitive	noise insensitive	noise insensitive
lighting	-	-	lighting
occlusion	-	(occlusion)	occlusion
(transparency)	-	-	transparency

Compositing

Comparison of techniques (Y. Sato, dataset of a left kidney):
Iso-surface vs. raycasting with **MIP**, **LMIP**, **α-compositing**




fast	fast	fast	-
(1 parameter)	parameter free	(1 parameter)	-
-	full data range	full data range	full data range
-	noise insensitive	noise insensitive	noise insensitive
lighting	-	-	lighting
occlusion	-	(occlusion)	occlusion
(transparency)	-	-	transparency

α-compositing

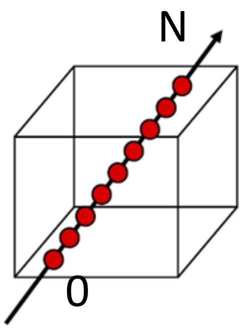
Assume that each sample on a view ray has color and opacity:

$$(C_0, \alpha_0), \cdots, (C_N, \alpha_N) \qquad C_i \in [0,1]^3, \alpha_i \in [0,1]$$

where the 0th sample is next to the camera

and the Nth one is a (fully opaque) background sample:

$$C_N = (r, g, b)_{\text{background}}$$
$$\alpha_N = 1$$



α-compositing

Assume that each sample on a view ray has color and opacity:

$$(C_0,\alpha_0),\cdots,(C_N,\alpha_N) \qquad C_i \in [0,1]^3, \alpha_i \in [0,1]$$

where the 0th sample is next to the camera

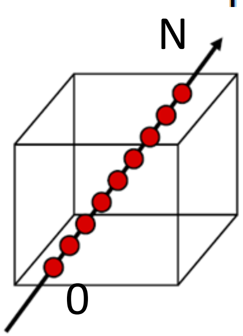
and the Nth one is a (fully opaque) background sample:

$$C_N = (r,g,b)_{\text{background}}$$
$$\alpha_N = 1$$

α-compositing can be defined recursively:

Let C_f^b denote the composite color of samples $f, f+1, \ldots, b$

Recursion formula for back-to-front compositing:



$$C_b^b = \alpha_b C_b$$ background color and opacity

Composite color of the current iteration

$$C_f^b = \alpha_f C_f + (1-\alpha_f) C_{f+1}^b$$

Composite color of the previous iteration

Composite opacity

$$\alpha_f{}^b = \alpha_f + (1-\alpha_f)\alpha_{f+1}{}^b$$

α-compositing

The first few generations, written with transparency $T_i = 1 - \alpha_i$

$$C_b^b = \alpha_b C_b$$

$$C_{b-1}^b = \alpha_{b-1} C_{b-1} + \alpha_b C_b T_{b-1}$$

$$C_{b-2}^b = \alpha_{b-2} C_{b-2} + \alpha_{b-1} C_{b-1} T_{b-2} + \alpha_b C_b T_{b-1} T_{b-2}$$

$$C_{b-3}^b = \alpha_{b-3} C_{b-3} + \alpha_{b-2} C_{b-2} T_{b-3} + \alpha_{b-1} C_{b-1} T_{b-2} T_{b-3} + \alpha_b C_b T_{b-1} T_{b-2} T_{b-3}$$

reveal the closed formula for α-compositing:

$$C_f^b = \sum_{i=f}^{b} \alpha_i C_i \prod_{j=f}^{i-1} T_j$$

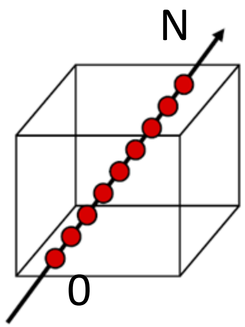
α-compositing

front-to-back compositing can be derived from the closed formula:

Let T_f^b denote the composite transparency of samples $f, f+1, \ldots, b$

$$T_f^b = \prod_{j=f}^{b} T_j$$

Then the simultaneous recursion for front-to-back compositing is:



$$\begin{aligned} C_f^f &= \alpha_f C_f \\ T_f^f &= 1 - \alpha_f \\ C_f^{b+1} &= C_f^b + \alpha_{b+1} C_{b+1} T_f^b \\ T_f^{b+1} &= \left(1 - \alpha_{b+1}\right) T_f^b \end{aligned}$$

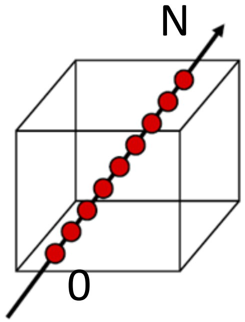
α-compositing

front-to-back compositing can be derived from the closed formula:

Let T_f^b denote the composite transparency of samples $f, f+1, \ldots, b$

$$T_f^b = \prod_{j=f}^{b} T_j$$

Then the simultaneous recursion for front-to-back compositing is:

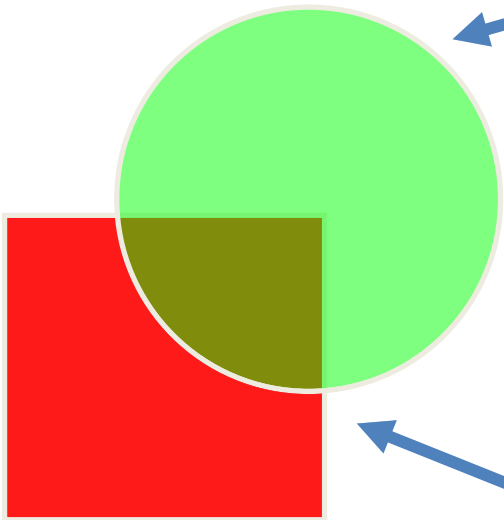


$$\begin{aligned} C_f^f &= \alpha_f C_f \\ T_f^f &= 1 - \alpha_f \\ C_f^{b+1} &= C_f^b + \alpha_{b+1} C_{b+1} T_f^b \\ T_f^{b+1} &= \left(1 - \alpha_{b+1}\right) T_f^b \end{aligned}$$

Advantage of front-to-back compositing: early ray termination when composite transparency falls below a threshold.

Compositing Example I

$c_f = (0,1,0)$
$a_f = 0.4$



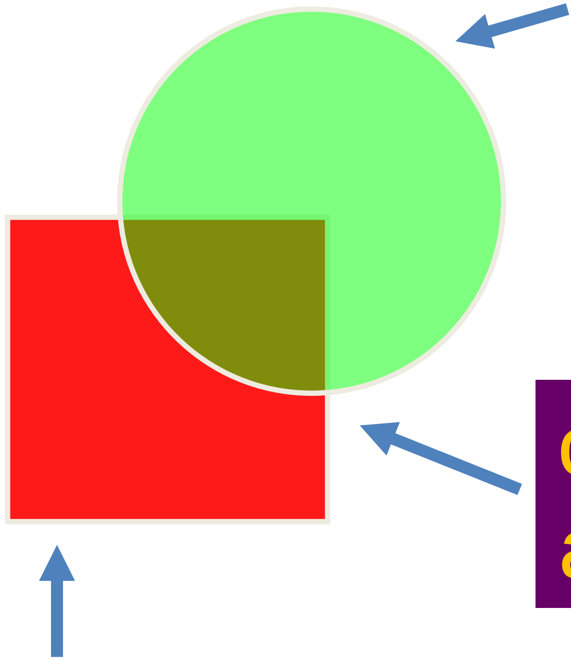
$c = a_f * c_f + (1 - a_f) * a_b * c_b$
$a = a_f + (1 - a_f) * a_b$

$c_b = (1,0,0)$
$a_b = 0.9$

Compositing Example I

$c_f = (0,1,0)$

$a_f = 0.4$



$c = a_f * c_f + (1 - a_f) * a_b * c_b$

$a = a_f + (1 - a_f) * a_b$

$c_b = (1,0,0)$

$a_b = 0.9$

$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$

$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$

$c_{blue} = 0.4*0 + (1-0.4)*0.9*0 = 0$

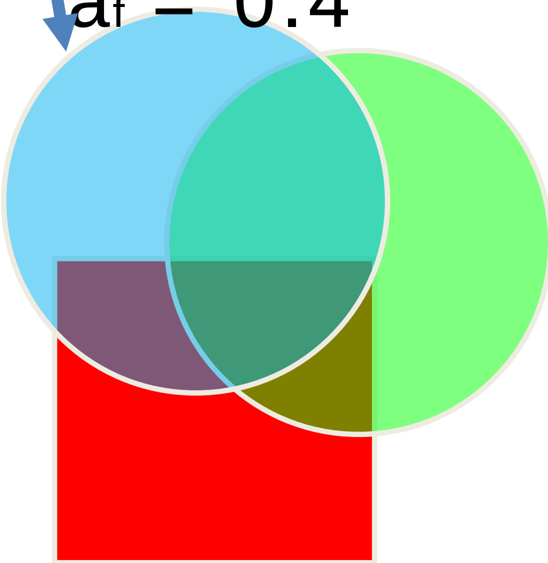
$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9$

$c = (0.54, 0.4, 0)$

$a = 0.94$

Compositing Example II

$c_f = (0,1,1)$
$a_f = 0.4$



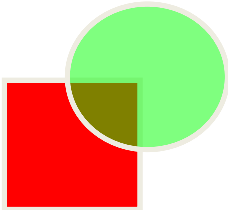
$c_b = (1,0,0)$
$a_b = 0.9$

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

$c_f = (0,1,0)$
$a_f = 0.4$

$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$
$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$c_{blue} = 0.4*0 + (1-0.4)*0.9*0 = 0$
$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9)$

$c_b = (0.54,0.4,0)$
$a_b = 0.94$

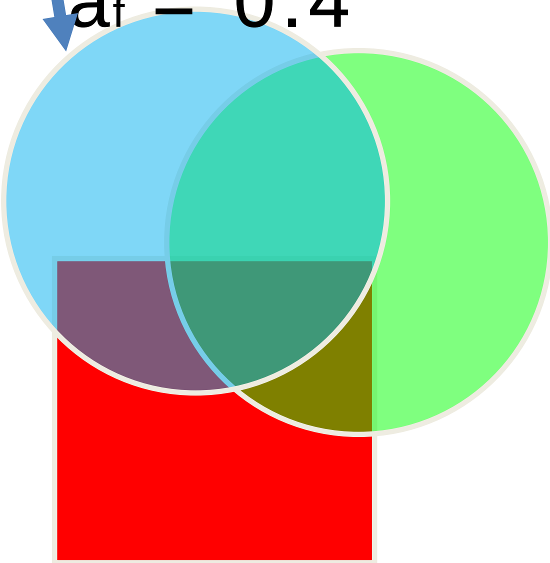


Compositing Example II

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

$c_f = (0,1,1)$

$a_f = 0.4$

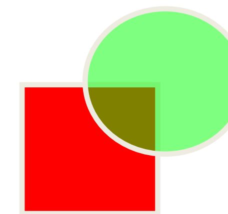


$c_b = (1,0,0)$

$a_b = 0.9$

$c_f = (0,1,0)$

$a_f = 0.4$



$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$

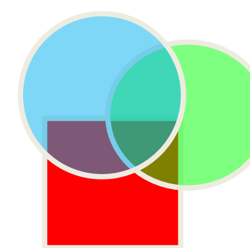
$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$

$c_{blue} = 0.4*0 + (1-0.4)*0.9*0 = 0$

$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9)$

$c_b = (0.54,0.4,0)$

$a_b = 0.94$



$c_{red} = 0.4*0 + (1-0.4)*0.94*0.54 = 0.6*0.94*.54 = 0.30$

$c_{green} = 0.4*1 + (1-0.4)*0.94*0.4 = 0.6*0.94*.4 = 0.23$

$c_{blue} = 0.4*1 + (1-0.4)*0.94*0 = .4$

$a = 0.4 + (1 - 0.4)*(0.94) = 0.4 + 0.6*0.94) = .964$

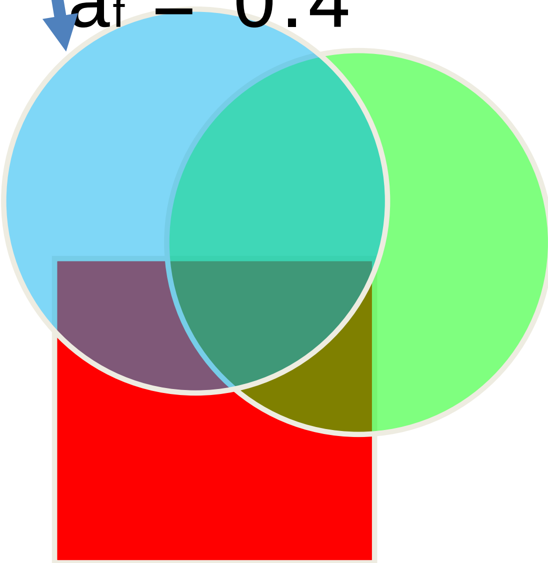
$c = (0.3,0.23,0.4)$

$a = 0.964$

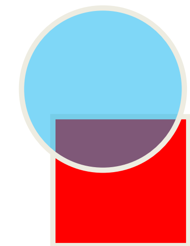
Compositing Example II

$c_f = (0,1,1)$
$a_f = 0.4$

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$



$c_f = (0,1,0)$
$a_f = 0.4$



$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$
$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$c_{blue} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9)$

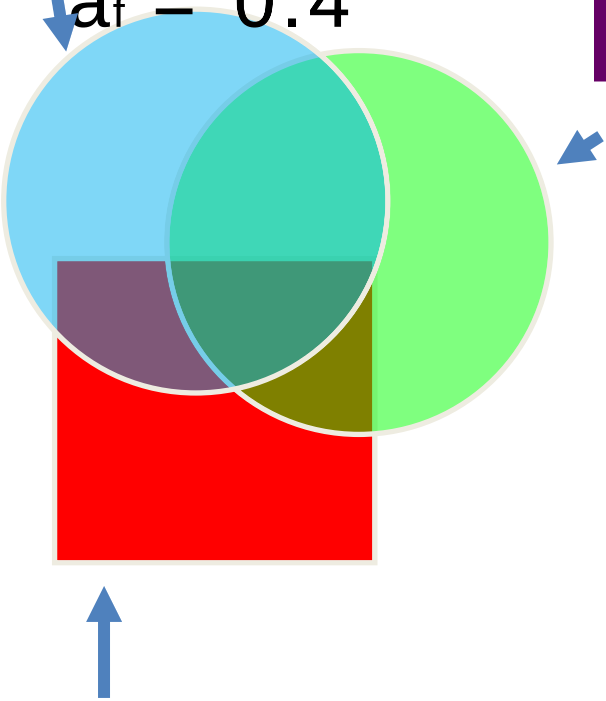
$c_b = (0.54,0.4,0.4)$
$a_b = 0.94$

$c_b = (1,0,0)$
$a_b = 0.9$

Compositing Example II

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

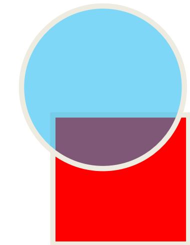
$c_f = (0,1,1)$
$a_f = 0.4$



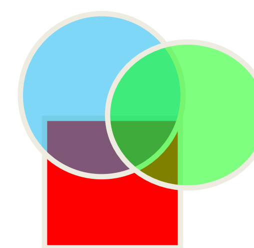
$c_b = (1,0,0)$
$a_b = 0.9$

$c_f = (0,1,0)$
$a_f = 0.4$

$c_{red} = 0.4*0 + (1-0.4)*0.9*1 = 0.6*0.9 = 0.54$
$c_{green} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$c_{blue} = 0.4*1 + (1-0.4)*0.9*0 = 0.4$
$a = 0.4 + (1 - 0.4)*(0.9) = 0.4 + 0.6*0.9)$



$c_b = (0.54,0.4,0.4)$
$a_b = 0.94$



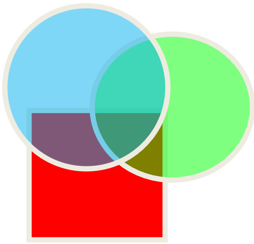
$c_{red} = 0.4*0 + (1-0.4)*0.94*0.54 = 0.6*0.94*.54 = 0.30$
$c_{green} = 0.4*1 + (1-0.4)*0.94*0.4 = 0.6*0.94*.4 = 0.23$
$c_{blue} = 0.4*1 + (1-0.4)*0.94*0.4 = .23$
$a = 0.4 + (1 - 0.4)*(0.94) = 0.4 + 0.6*0.94) = .964$

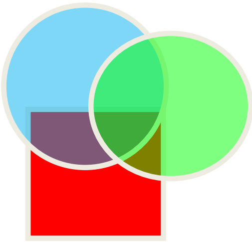
$c = (0.3,0.23,0.23)$
$a = 0.964$

Compositing Orders

$$c = a_f * c_f + (1 - a_f) * a_b * c_b$$
$$a = a_f + (1 - a_f) * a_b$$

Order Matters!





$c = (0.3, 0.23, \mathbf{0.4})$
$a = 0.964$

$c = (0.3, 0.23, \mathbf{0.23})$
$a = 0.964$

The Emission-Absorption Model

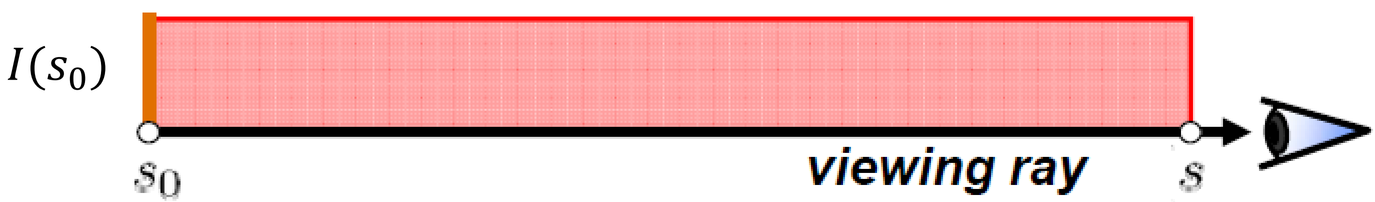
How realistic is α-compositing?

The emission-absorption model (Sabella 1988)

The Emission-Absorption Model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988)



$I(s) = I(s_0)$

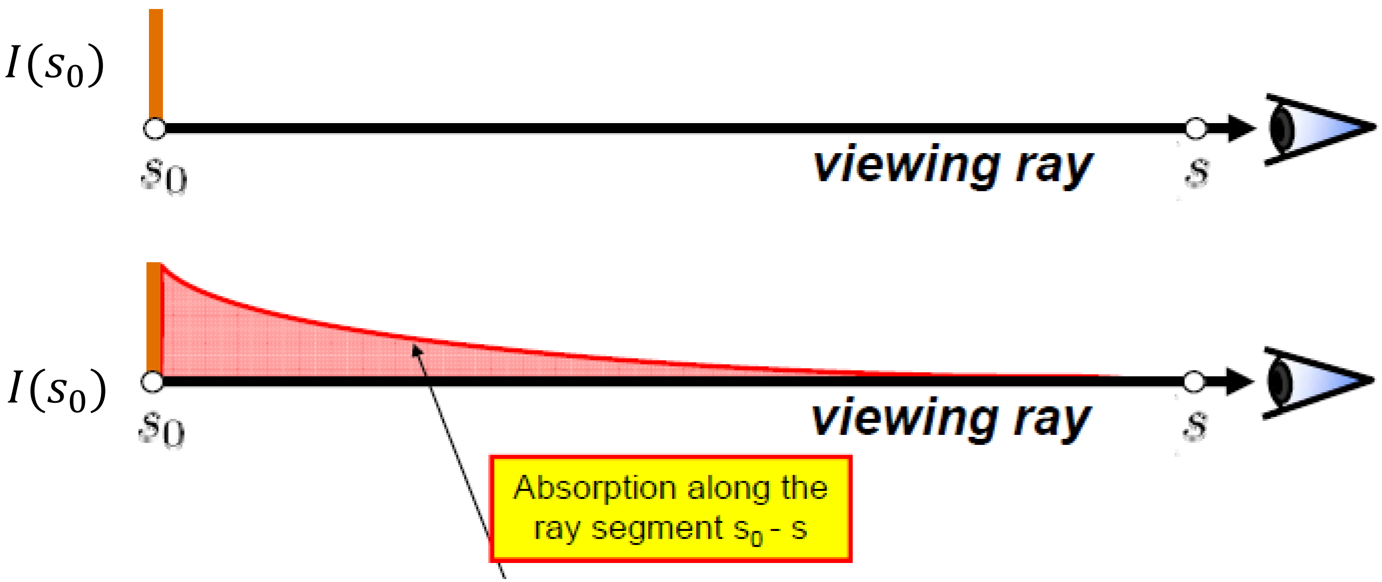
Initial intensity at s_0

Without absorption all the initial radiant energy would reach the point ***s.***

The Emission-Absorption Model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988)



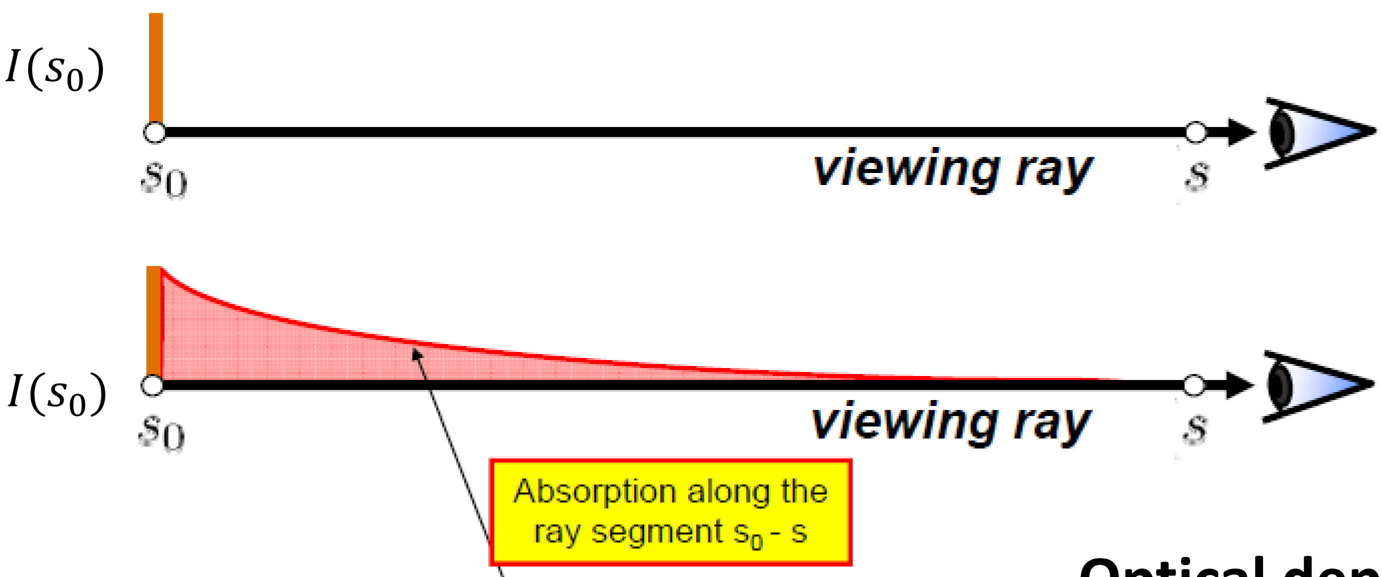
Initial intensity at s_0

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

The Emission-Absorption Model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988)



Initial intensity at s_0

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

Optical depth $\boldsymbol{\tau}$

Absorption $\boldsymbol{\kappa}$

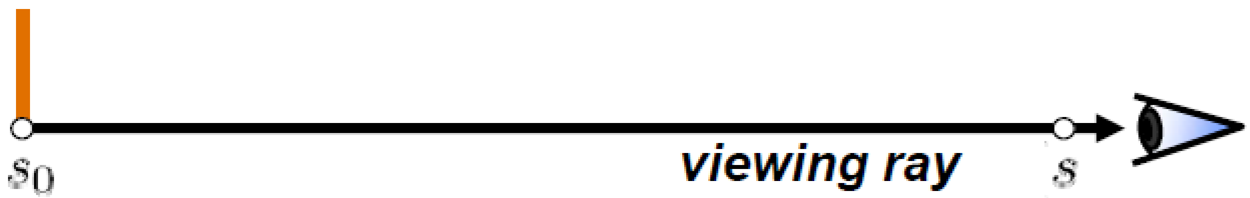
$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)ds$$

The Emission-Absorption Model

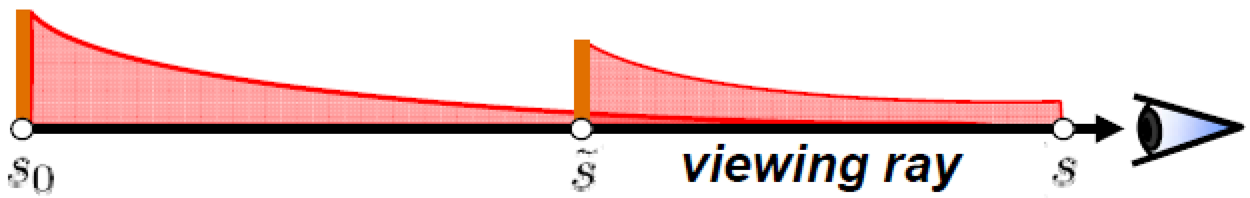
How realistic is α-compositing?

The emission-absorption model (Sabella 1988)

$I(s_0)$



$I(s_0)$



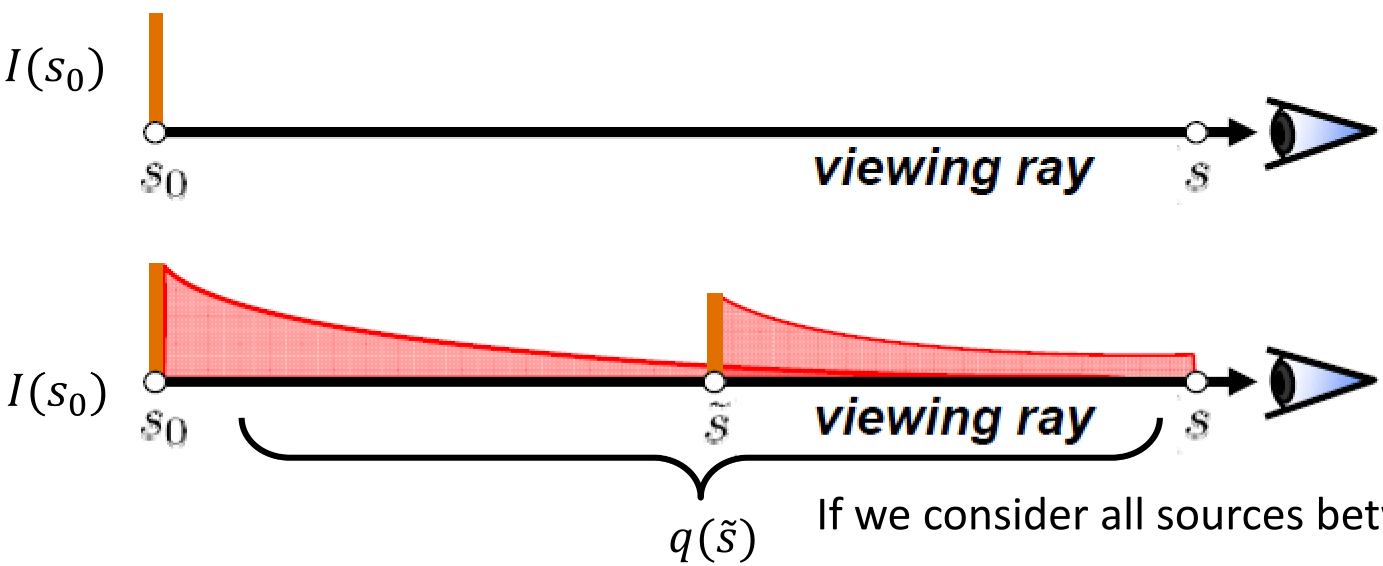
$q(\tilde{s})$ If we have a second source at $\tilde{s}$

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

The Emission-Absorption Model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988)



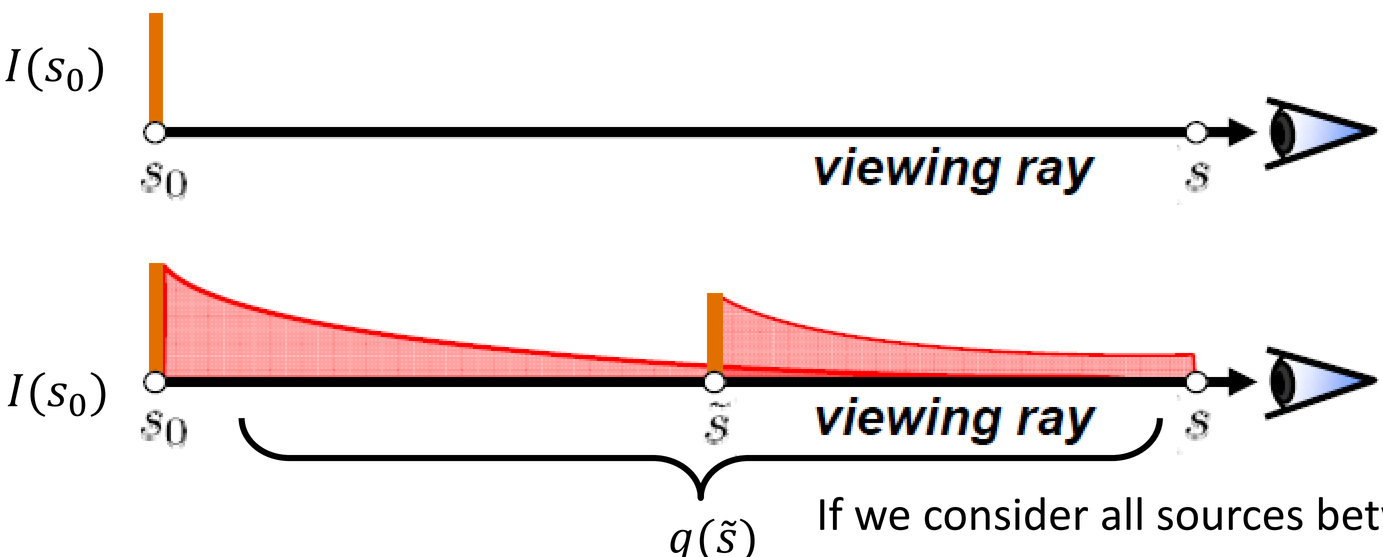
If we consider all sources between s0 and s…

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

The Emission-Absorption Model

How realistic is α-compositing?

The emission-absorption model (Sabella 1988)



If we consider all sources between s0 and s...

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

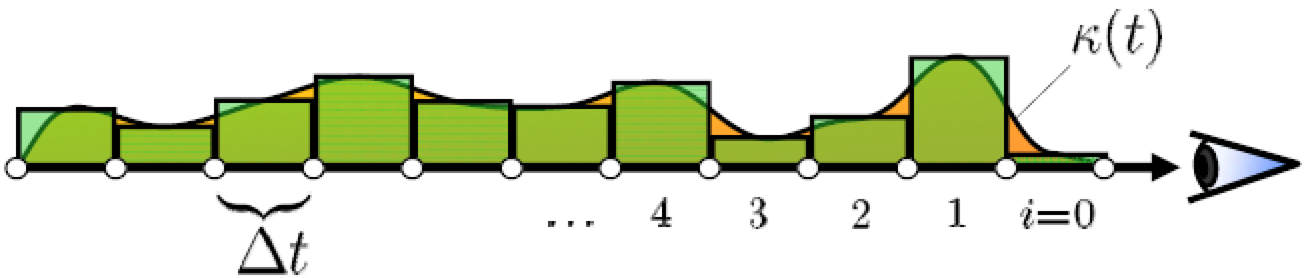
$$\tau(s_1, s_2) = \int_{s_1}^{s_2} \kappa(s)ds$$

Numerical Solution

We need to first estimate the optical depth by taking into account the absorption property of the material that the ray is traveling through.

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

Numerical Solution



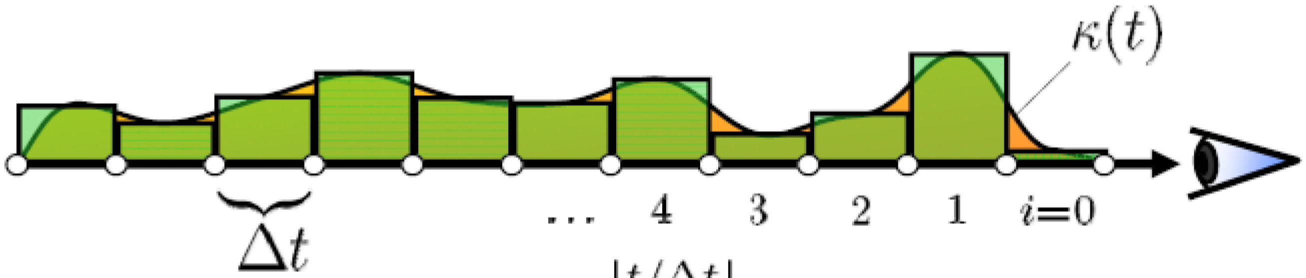
Optical depth: $\tau(0,t) \;=\; \int_0^t \kappa(\hat{t})\, d\hat{t}$

Approximate Integral by Riemann sum:

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

The area underneath the smooth curve $\kappa(t)$ can be approximated as the sum of the areas of the individual rectangles.

Numerical Solution

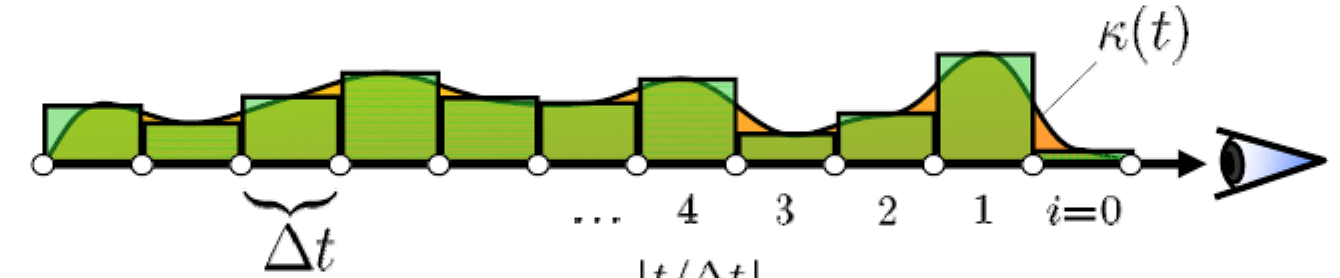


$$\tau(0,t) \approx \tilde{\tau}(0,t) = \sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\,\Delta t$$

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

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

Numerical Solution



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

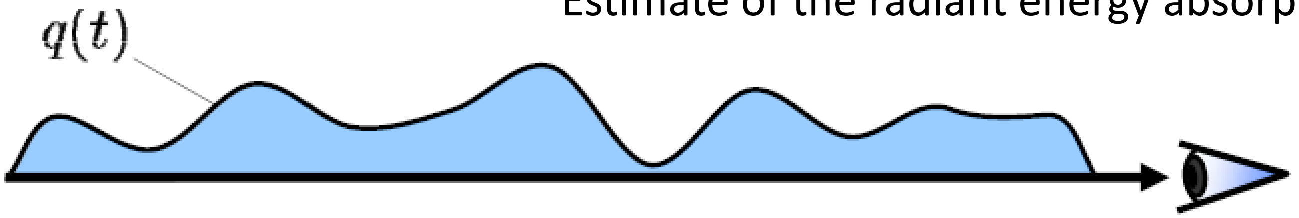
$$e^{-\tilde{\tau}(0,t)} = e^{-\sum_{i=0}^{\lfloor t/\Delta t \rfloor} \kappa(i \cdot \Delta t)\, \Delta t}$$

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

Now we introduce opacity $1 - A_i = e^{-\kappa(i \cdot \Delta t)\Delta t}$

Numerical Solution

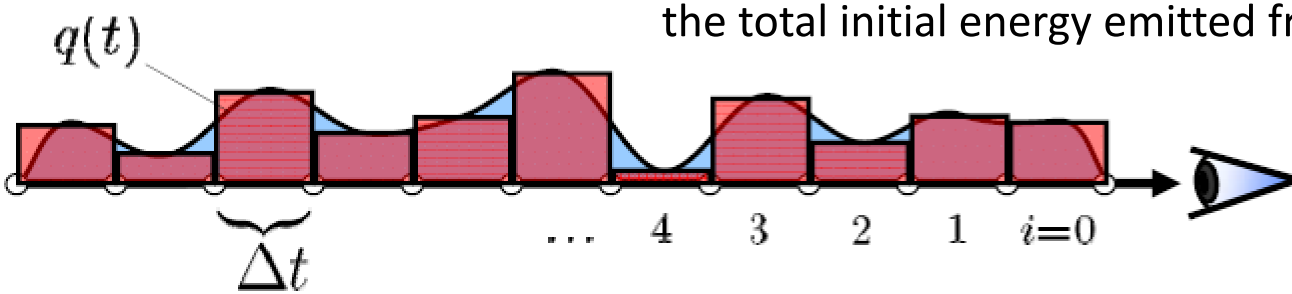
Estimate of the radiant energy absorption



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

Numerical Solution

The area of each rectangle approximates the total initial energy emitted from q(t)



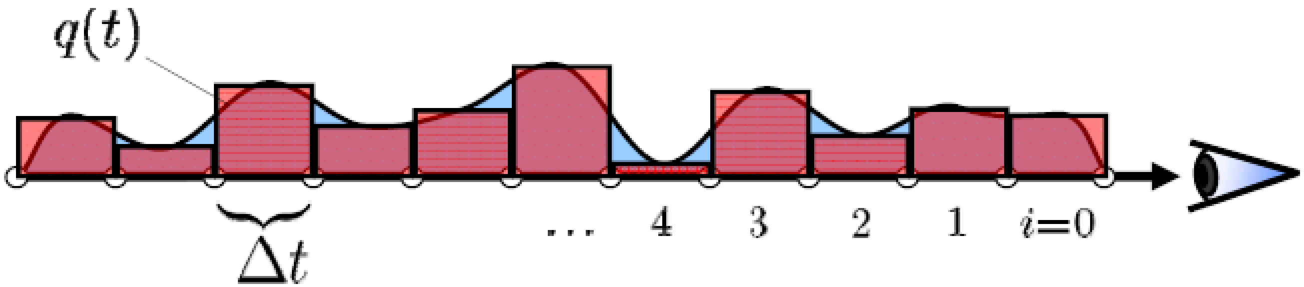
$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

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

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

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

Numerical Solution



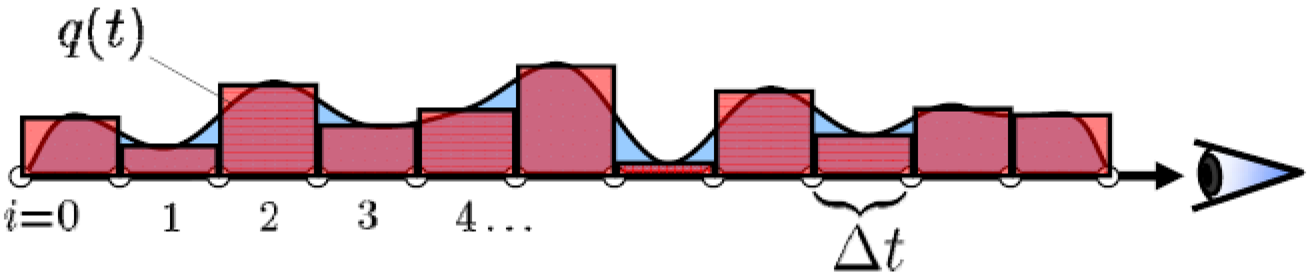
$$e^{-\tilde{\tau}(0,t)} = \prod_{i=0}^{\lfloor t/\Delta t \rfloor} (1 - A_i)$$

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

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

can be computed recursively/iteratively!

Numerical Solution



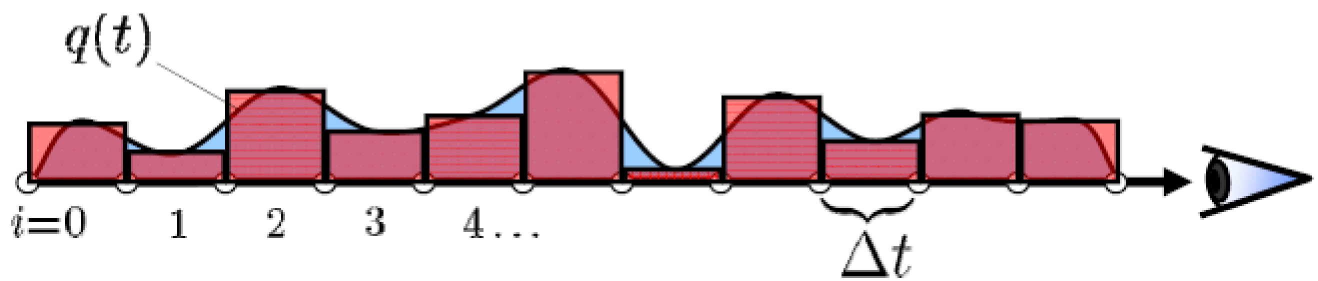
Note: we just changed the convention from i=0 is at the front of the volume (previous slides) to i=0 is at the back of the volume !

can be computed recursively/iteratively:

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

$$C_i = A_i C_{pure\_color}$$

Numerical Solution



can be computed recursively/iteratively:

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

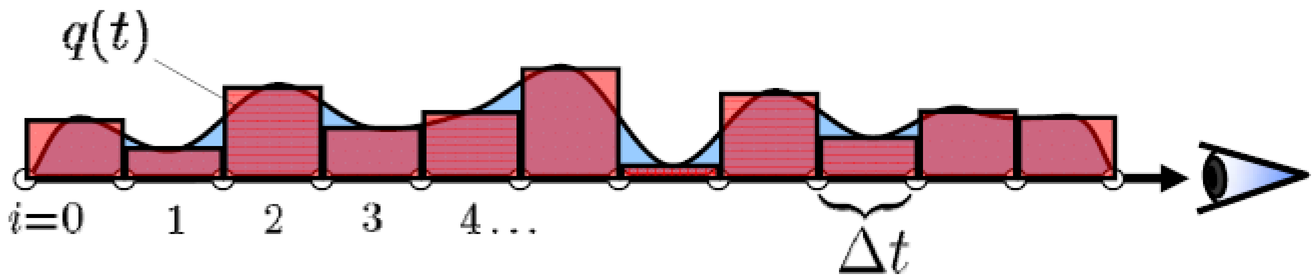
Radiant energy observed at position ***i***

Radiant energy emitted at position ***i***

Absorption at position ***i***

Radiant energy observed at position ***i*–1**

Numerical Solution



Back-to-front compositing

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

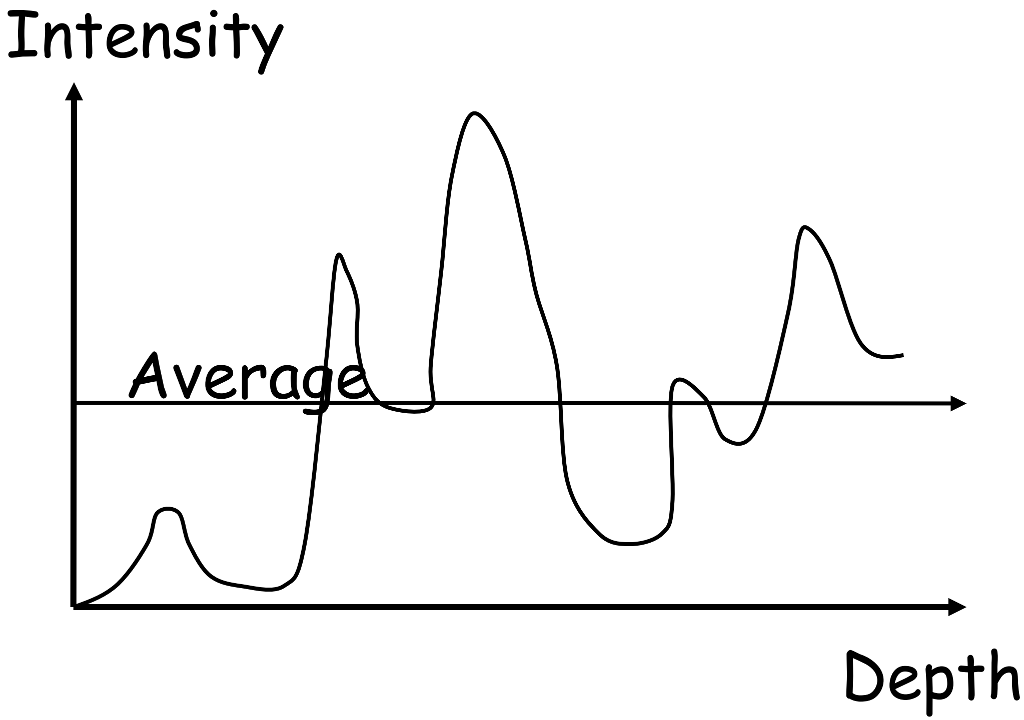
Iterate from *i*=0 (back) to *i*=max (front): *i* increases

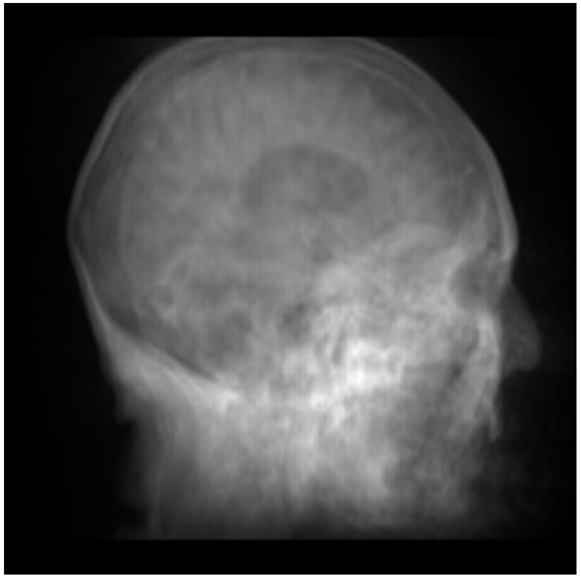
Front-to-back compositing

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

Iterate from *i*=max (front) to *i*=0 (back) : *i* decreases

Other Compositing - Average





Synthetic Reprojection