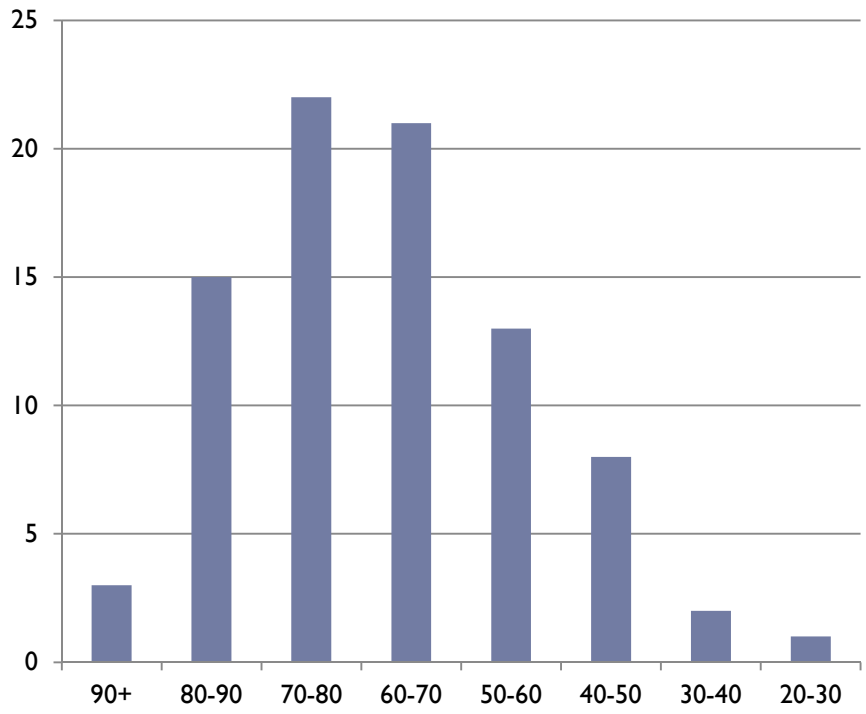
CSE 167:
Introduction to Computer Graphics
Lecture #17: Collisions

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2013

Midterm #2 Statistics

	Midterm 2
# Submissions	85
Average	67.3
Median	68
Highest	96.5
Lowest	24



Announcements

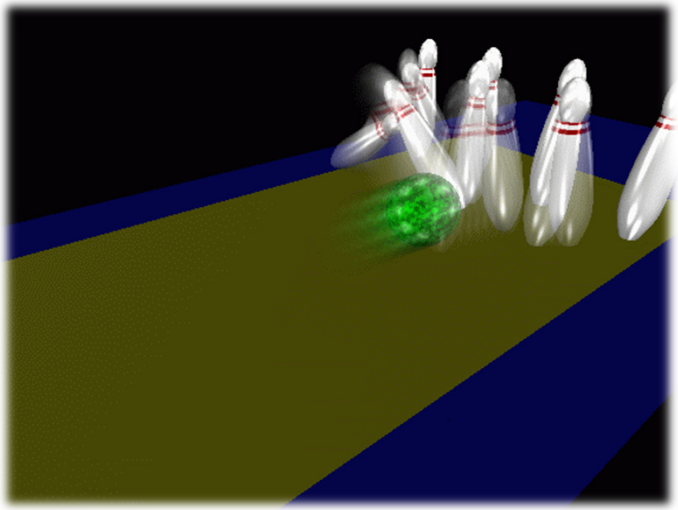
- Midterms
  - Double check total score
  - Cross-check score with Ted
  - If exam kept past end of lecture, cannot dispute grade later
  - If you disagree with a score, write note on front page

Lecture Overview

- Collision Detection
- Volume Rendering

Collision Detection

- Goals:
  - Physically correct simulation of collision of objects
    - Not covered here
  - Determine if two objects intersect
- Slow calculation because of exponential growth $O(n^2)$:
  - # collision tests = $n*(n-1)/2$



Intersection Testing

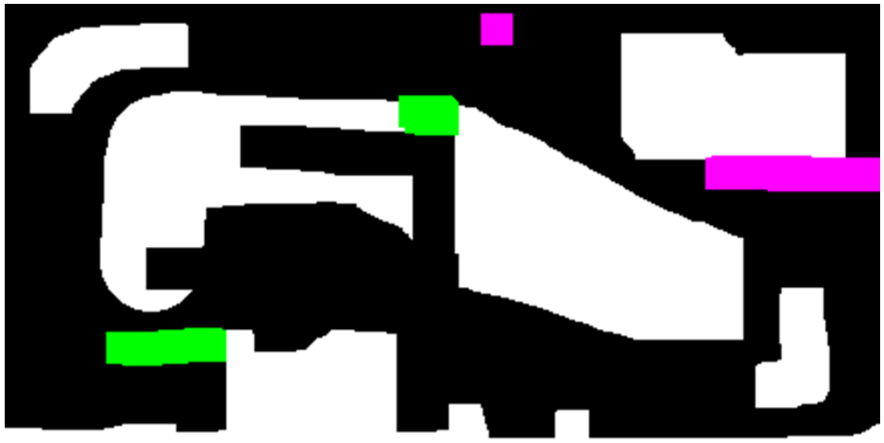
- Purpose:
  - Keep moving objects on the ground
  - Keep moving objects from going through walls, each other, etc.
- Goal:
  - Believable system, does not have to be physically correct
- Priority:
  - Computationally inexpensive
- Typical approach:
  - Spatial partitioning
  - Object simplified for collision detection by one or a few
    - Points
    - Spheres
    - Axis aligned bounding box (AABB)
  - Pairwise checks between points/spheres/AABBs and static geometry

Sweep and Prune Algorithm

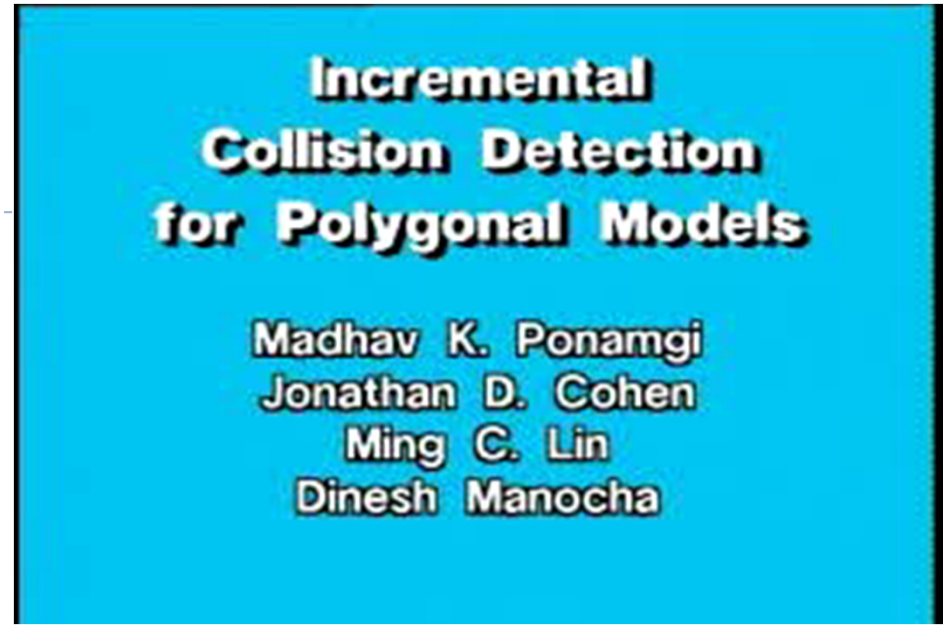
- Sorts bounding boxes
- Not intuitively obvious how to sort bounding boxes in 3-space
- Dimension reduction approach:
  - Project each 3-dimensional bounding box onto the x,y and z axes
  - Find overlaps in 1D: a pair of bounding boxes can overlap if and only if their intervals overlap in all three dimensions
    - Construct 3 lists, one for each dimension
    - Each list contains start/end point of intervals corresponding to that dimension
    - By sorting these lists, we can determine which intervals overlap
    - Reduce sorting time by keeping sorted lists from previous frame, changing only the interval endpoints
- Alternative: project bounding boxes onto coordinate axis planes and look for overlaps in 2D

Collision Map (CM)

- 2D map with information about where objects can go and what happens when they go there
- Colors indicate different types of locations
- Map can be computed from 3D model, or hand drawn with paint program
- Granularity: defines how much area (in object space) one CM pixel represents



References



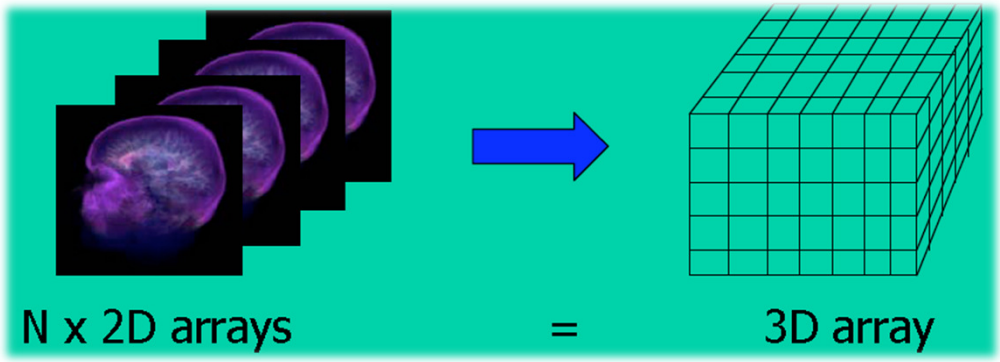
- I-Collide:
  - Interactive and exact collision detection library for large environments composed of convex polyhedra
  - http://gamma.cs.unc.edu/I-COLLIDE/
- OZ Collide:
  - Fast, complete and free collision detection library in C++
  - Based on AABB tree
  - http://www.tsarevitch.org/ozcollide/

Lecture Overview

- Volume Rendering
  - Overview
  - Transfer Functions
  - Rendering

What is a Volume Data Set?

- A *Volume Data Set* is a regular 3D array of data points, typically discrete sample points of a 3D object or simulation.
- Analogous to pixels (for **pic**ture **el**ements), the sample points in a volume data set are called *voxels*.
- Stacks of 2D images produced by CT, MRI or 3D ultrasound devices are naturally represented as volume data sets.



- Typical volume sizes range from 64^3 to 1024^3 voxels.
- Volume sizes do not have to be cubic; for example, 512x512x30 slices is typical for CT or MRI because the acquisition time increases with the number of cross-sections.

Volume Data Types

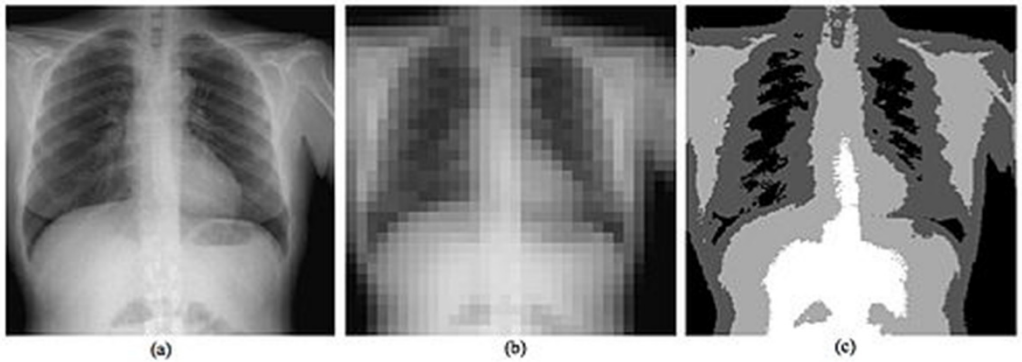


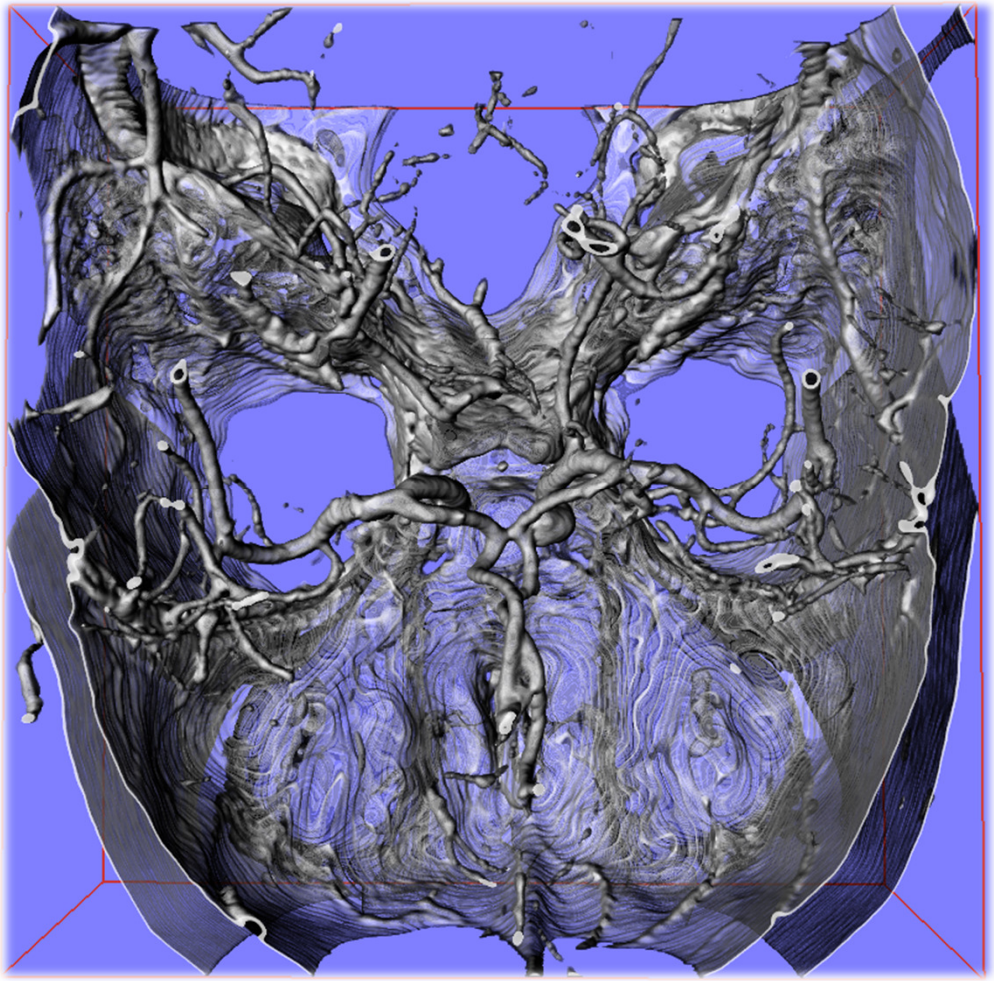
Image from Wikibooks.org:
a) 256 x 256 pixels, 8 bits per pixel
b) 32 x 32 pixels, 8 bits per pixel
c) 256 x 256 pixels, 2 bits per pixel

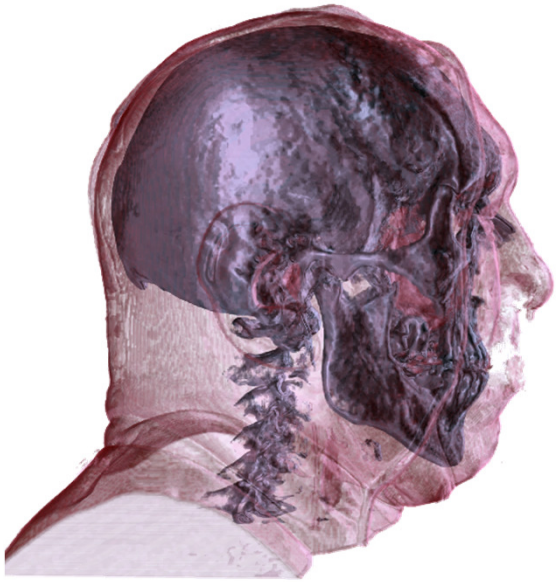
- Each voxel stores a data value
  - Single bit: binary data set, iso-surface
  - Typical: 8 or 16 bit integers
  - Simulations often generate one or more floating point numbers for each voxel
  - If multiple numbers are stored per voxel, data set is called multi-valued or multi-channel
    - Example: RGB colors in multi-channel confocal microscopy

What is Volume Rendering?

- *Volume Rendering* is a set of techniques used to display a 2D projection of a volume data set.
- User specifies viewpoint, rendering algorithm and transfer function
  - Transfer function defines mapping of color and opacity to voxels
- Wide spectrum of application domains

Medicine



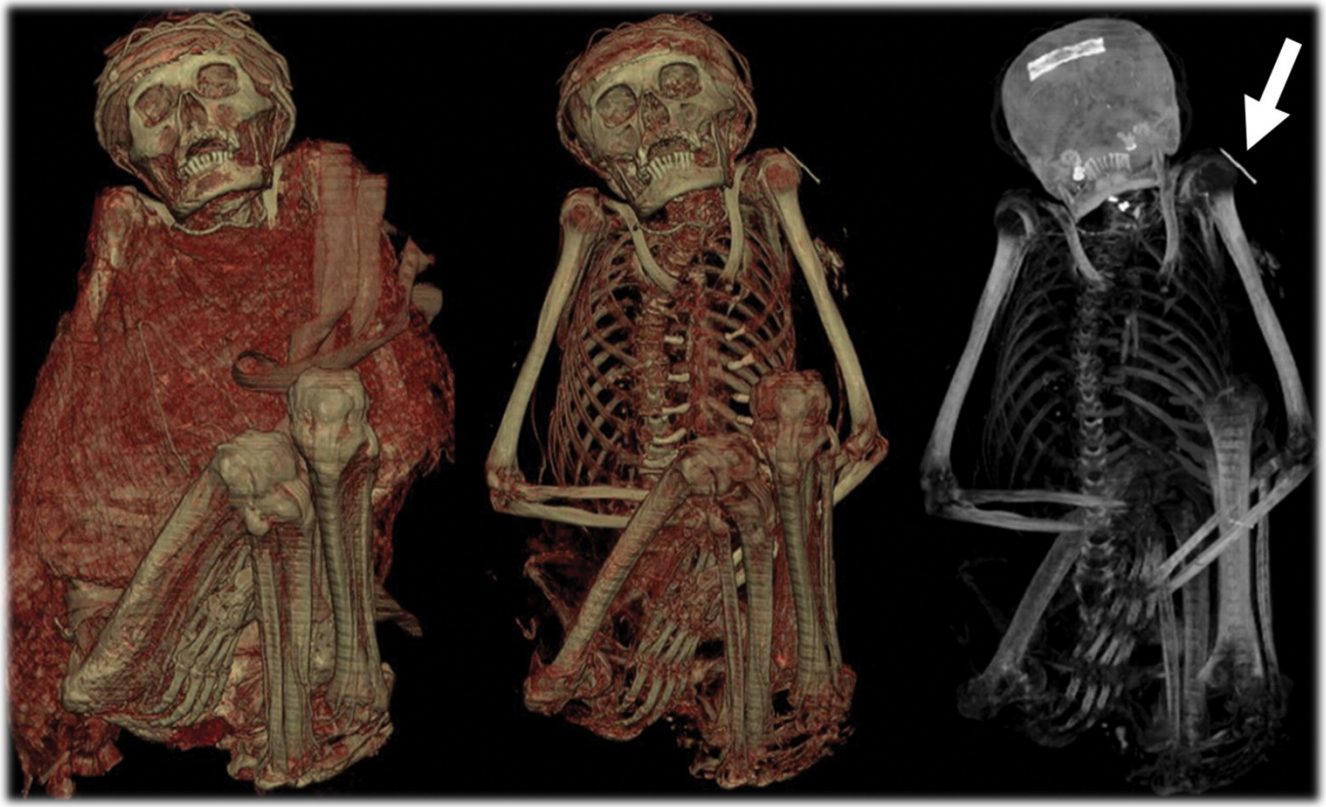


CT Human Head:
Visible Human Project, US National Library of Medicine, Maryland, USA

CT Angiography:
Dept. of Neuroradiology University of Erlangen, Germany

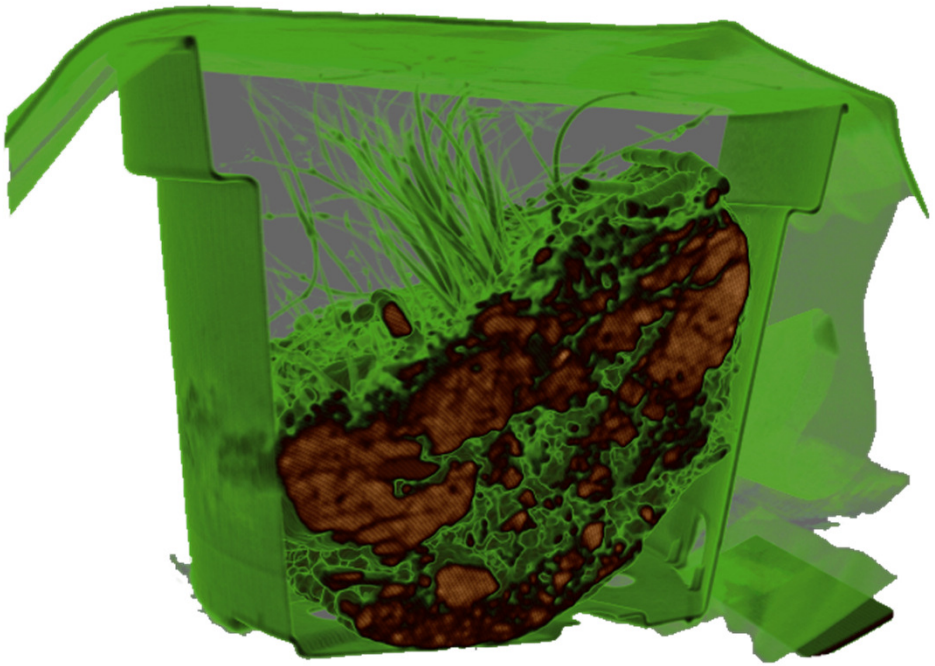
This and some of the following slides are from a Eurographics 2006 course by Dr. Christof Rezk Salama, Computer Graphics and Multimedia Group, University of Siegen, Germany

Archaeology



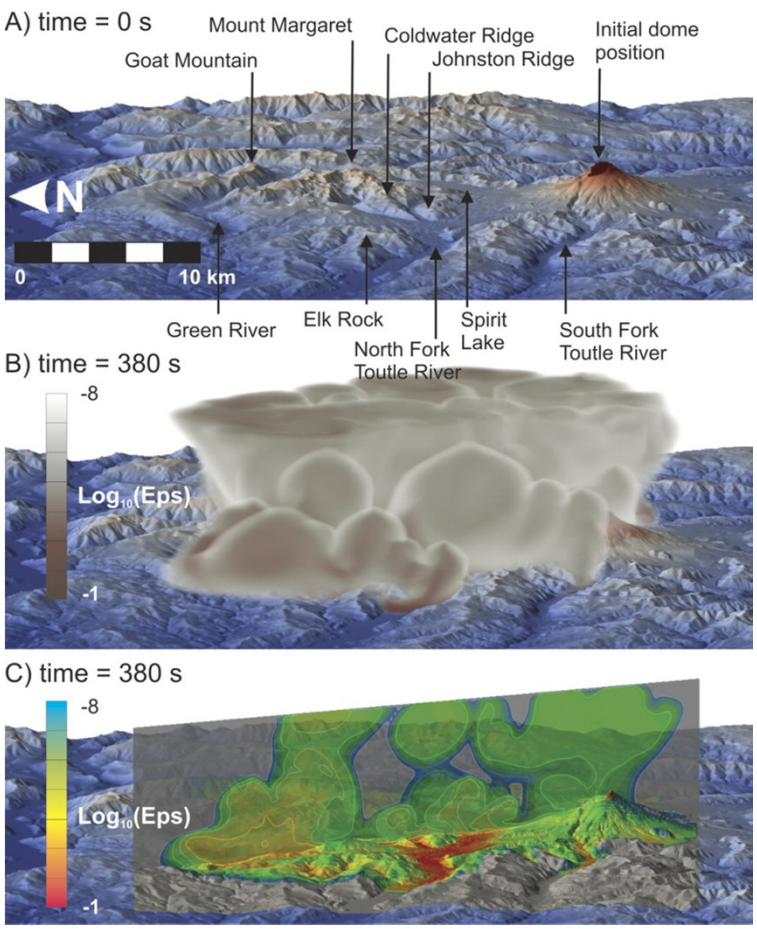
Mummified young man from the highlands of the Andes in South America
From: Jackowski C et al. Radiographics 2008;28:1477-1492

Biology



CT scan of biological sample of soil
Virtual Reality Group, University of Erlangen

Geology



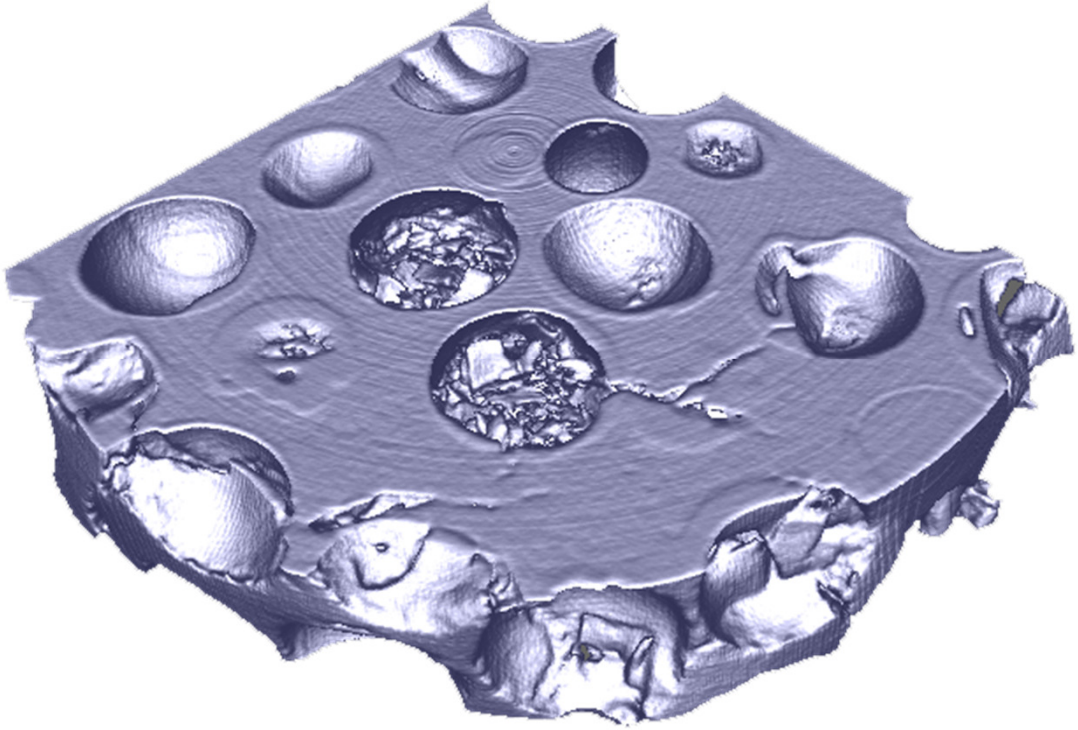
A) Initial position of exploding source and topographic setting.

B) Three-dimensional volume rendering of total volumetric fraction of pyroclasts in the atmosphere at 380 seconds.

C) Two-dimensional rendering of volumetric fraction of pyroclasts along northeast section.

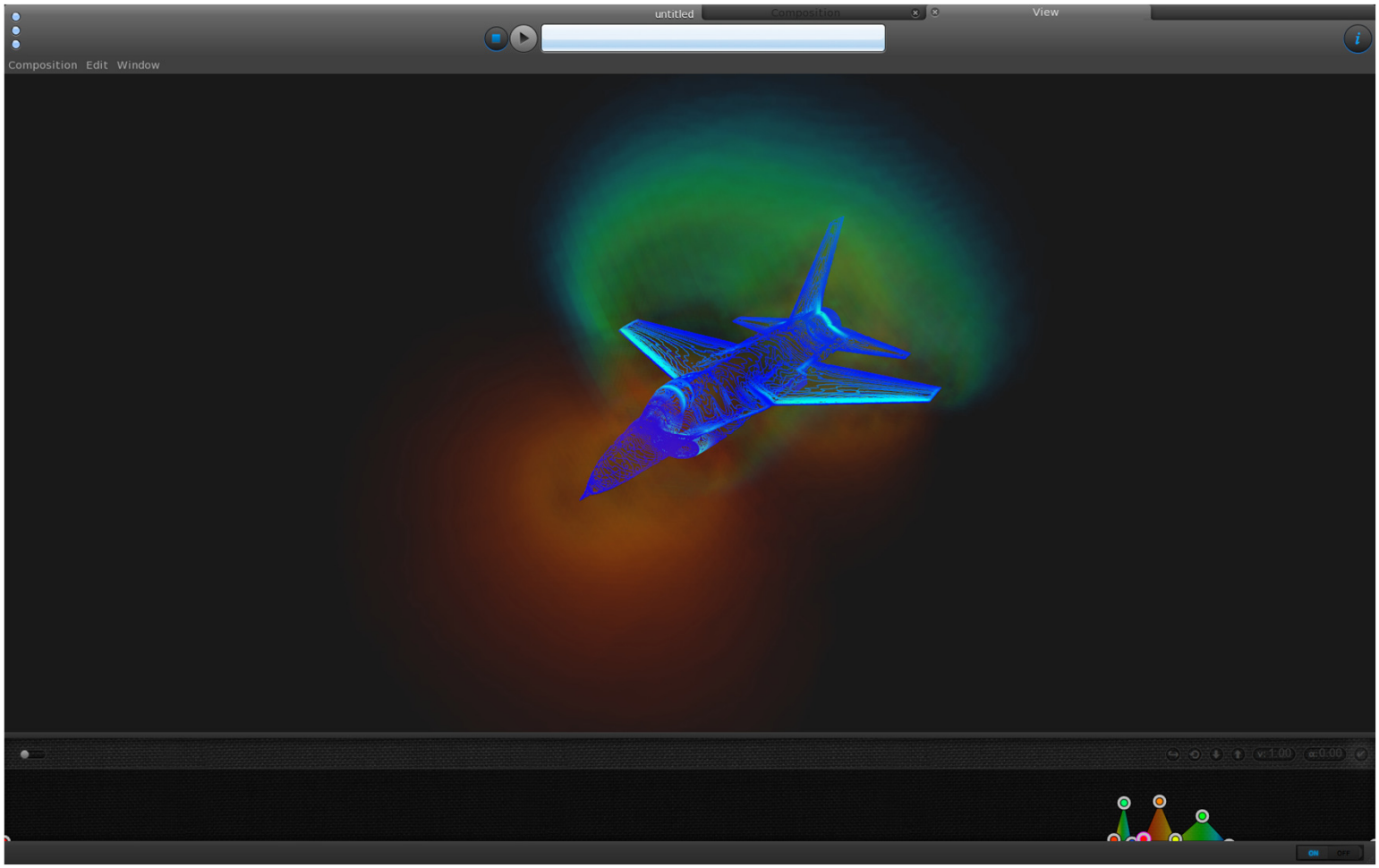
From: Ongaro T E et al. Geology 2011;39:535-538

Material Science



Micro CT of Compound Material
Material Science Department, University of Erlangen

Mechanical Engineering



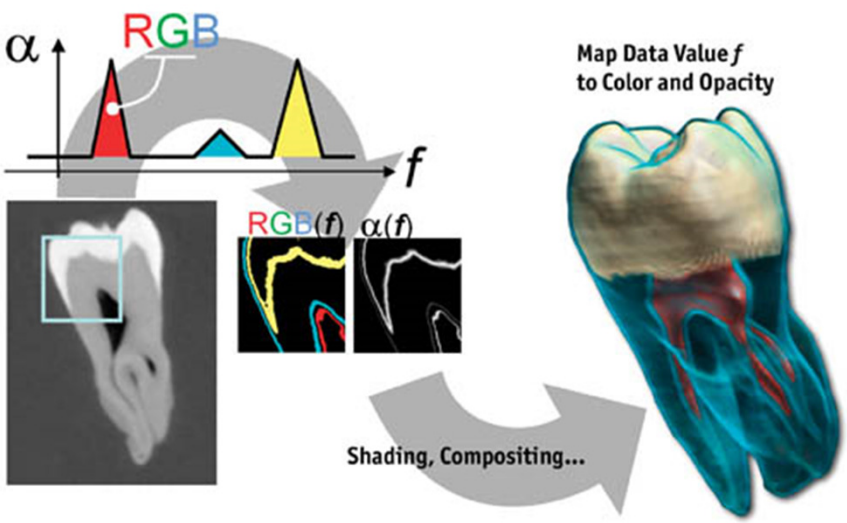
Volume rendering of shock waves around F16 aircraft (Thibaud Kloczko)

Lecture Overview

- Volume Rendering
  - Overview
  - Transfer Functions
  - Rendering

Transfer Functions

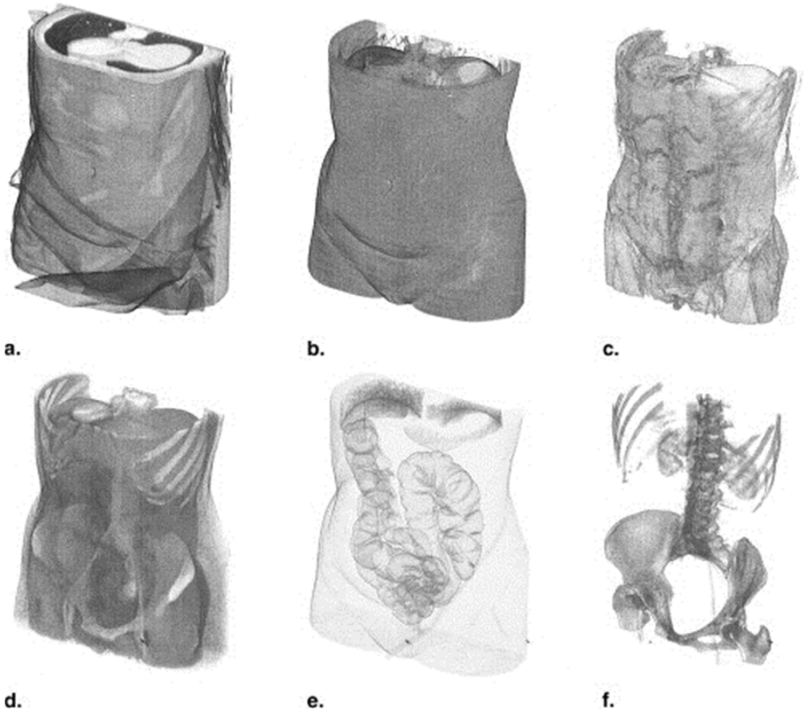
- Most volume data sets consist of a single data value per voxel, for instance the measured material density.
- This data value can be interpreted as:
  - Luminance and rendered as a gray value on a scale from black to white
  - An index into a color look-up table
- Another look-up table maps opacity to data values.



From GPU Gems

Opacity Transfer Function

- Modifying the mapping of data value to opacity exposes different parts of the volume.



From Shin et al. 2004: Images a-f show decreasing opacity.

Volume Filtering

- Applying a filter to the volume data set can improve image quality
- Filtering operation defined by filter kernel
- Filter kernels on right:
  - (a) blur filter
  - (b) sharpen filter
  - (c) edge filter
- In 3D, filter kernels typically use a 6-, 18- or 26-voxel neighborhood

1	1	1
1	1	1
1	1	1

(a)

0	-1	0
-1	10	-1
0	-1	0

(b)

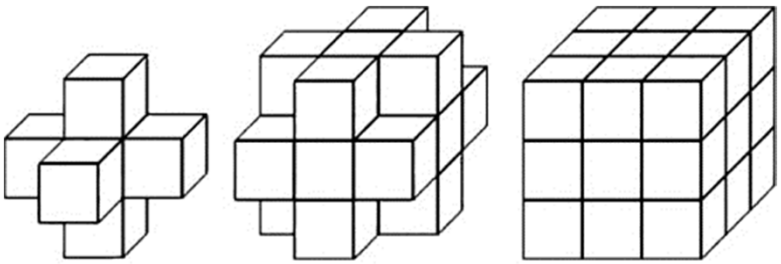
0	1	0
1	-4	1
0	1	0

(c)

(d)

(e)

(f)



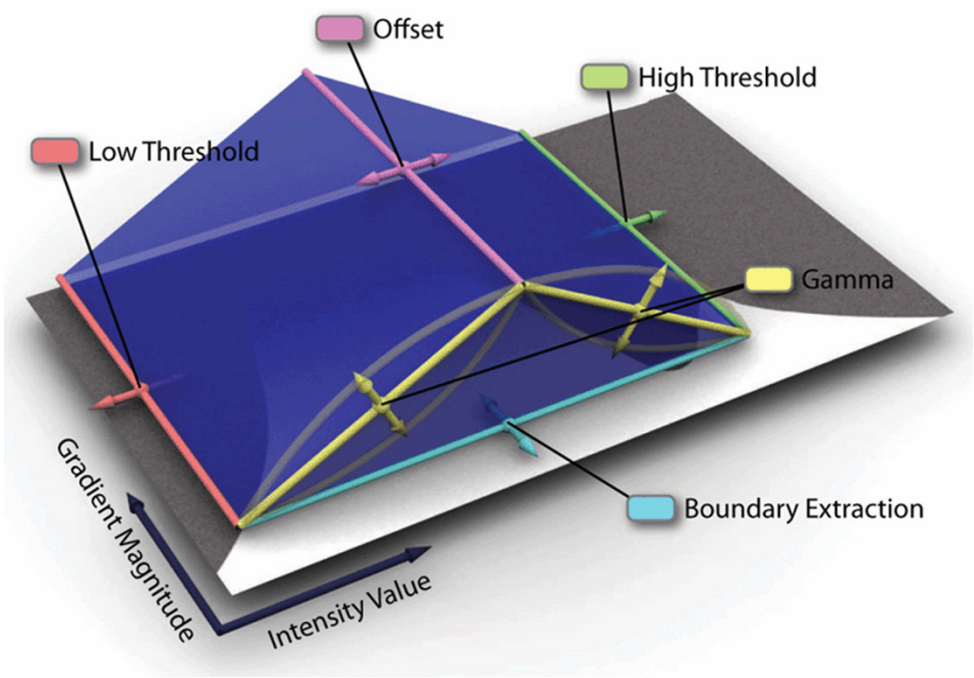
6-Neighborhood 18-Neighborhood 26-Neighborhood

Derived Voxel Data

- Applying a filter on the volume data set generates a new, derived volume data set.
- Derived volume data can be stored with original volume in a multi-channel volume data set.
- Example:
  - Channel 1: original density data
  - Channel 2: gradient magnitude

2D Transfer Functions

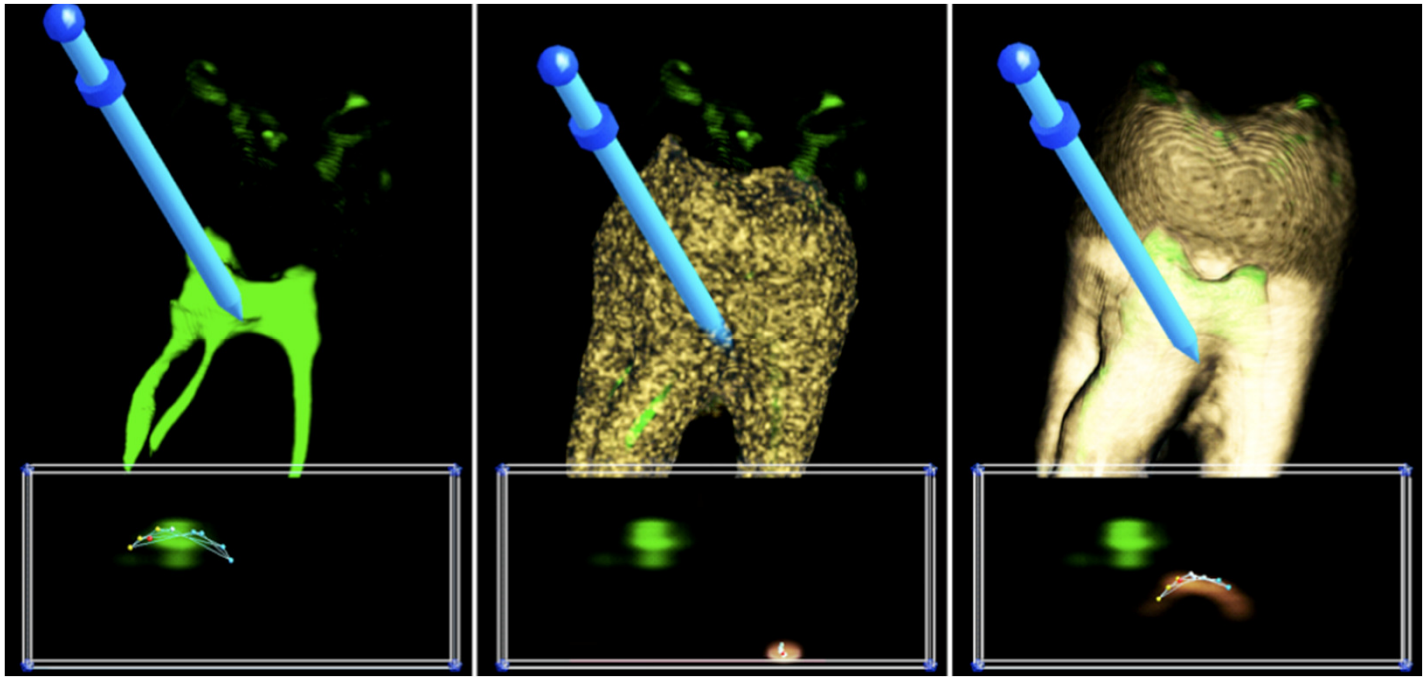
- A 2D transfer function can map RGBA values separately to every combination of density data and gradient magnitude



2D Transfer function and its parameters (Wan et al. 2009)

2D Transfer Functions

- Example: Rectangular 2D transfer function editor



Images by Gordon Kindleman and Joe Kniss

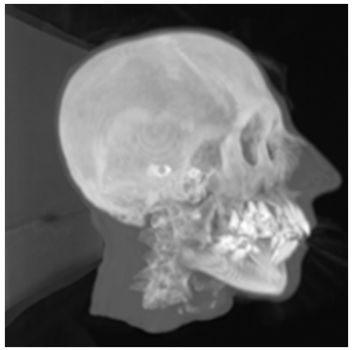
Lecture Overview

- Volume Rendering
  - Overview
  - Transfer Functions
  - Rendering

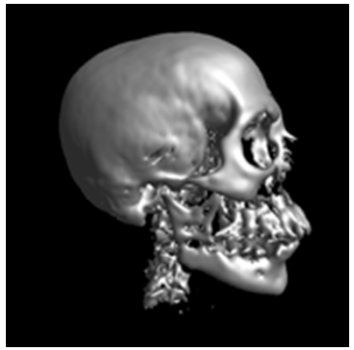
Volume Rendering Techniques

- Iso-surface
- Cross-sections
- Direct volume rendering (DVR)
  - Slicing with 2D textures
  - Translucent textures with image plane-aligned 3D textures
  - MIP
- Spatial constraints
  - Region of interest (cubic, spherical, slab)
  - Clipping plane

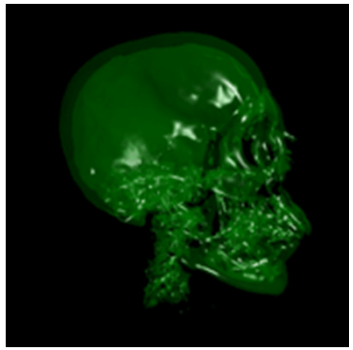
Volume Rendering Techniques



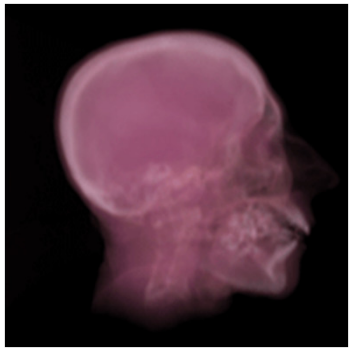
Maximum Intensity Projection



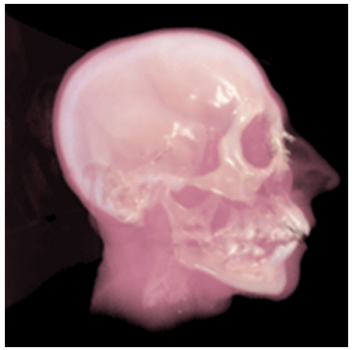
Iso-Surface



Transparent Iso-Surfaces



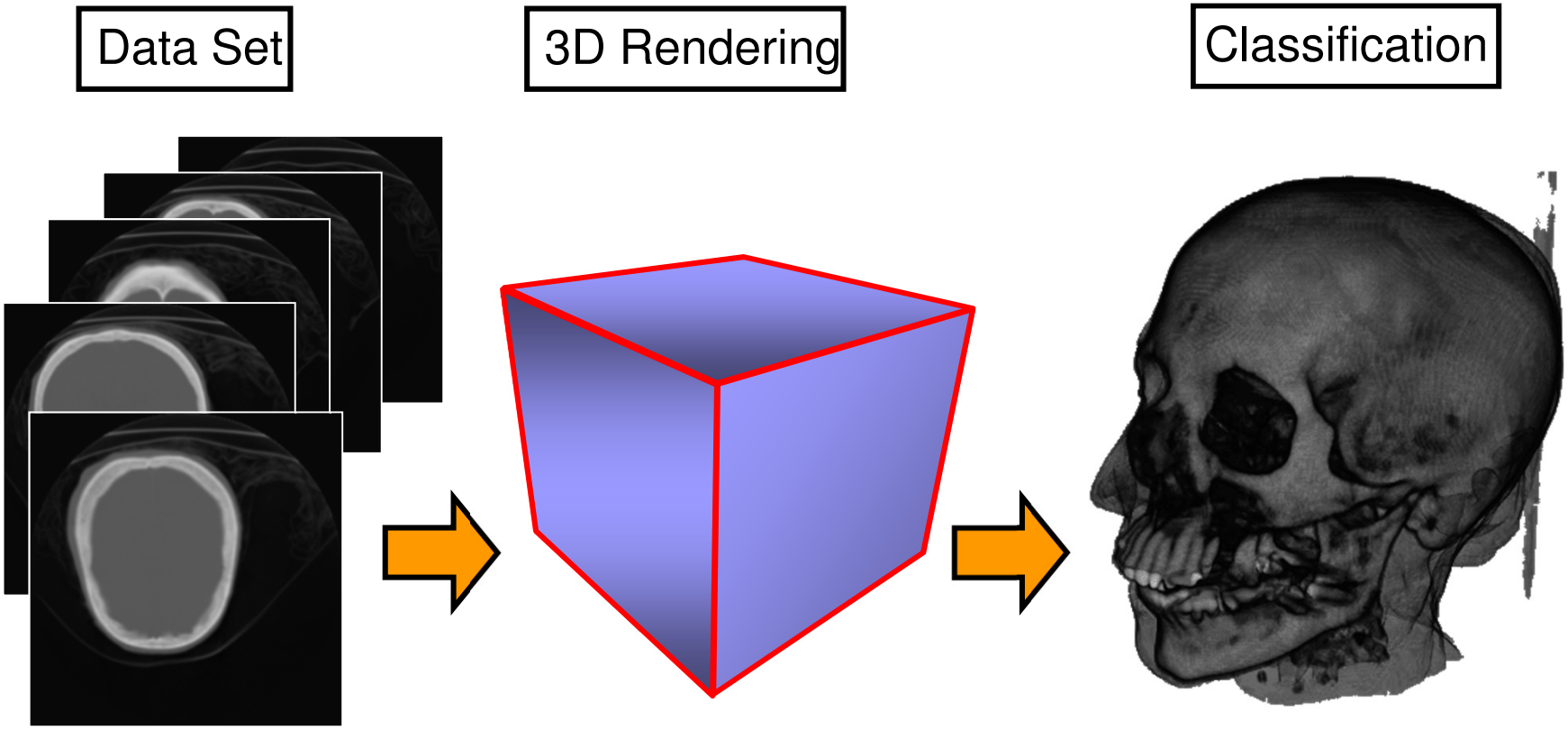
Raycasting (DVR)



Raycasting and Iso-Surface

Images from: A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting, Stegmaier et al. 2005

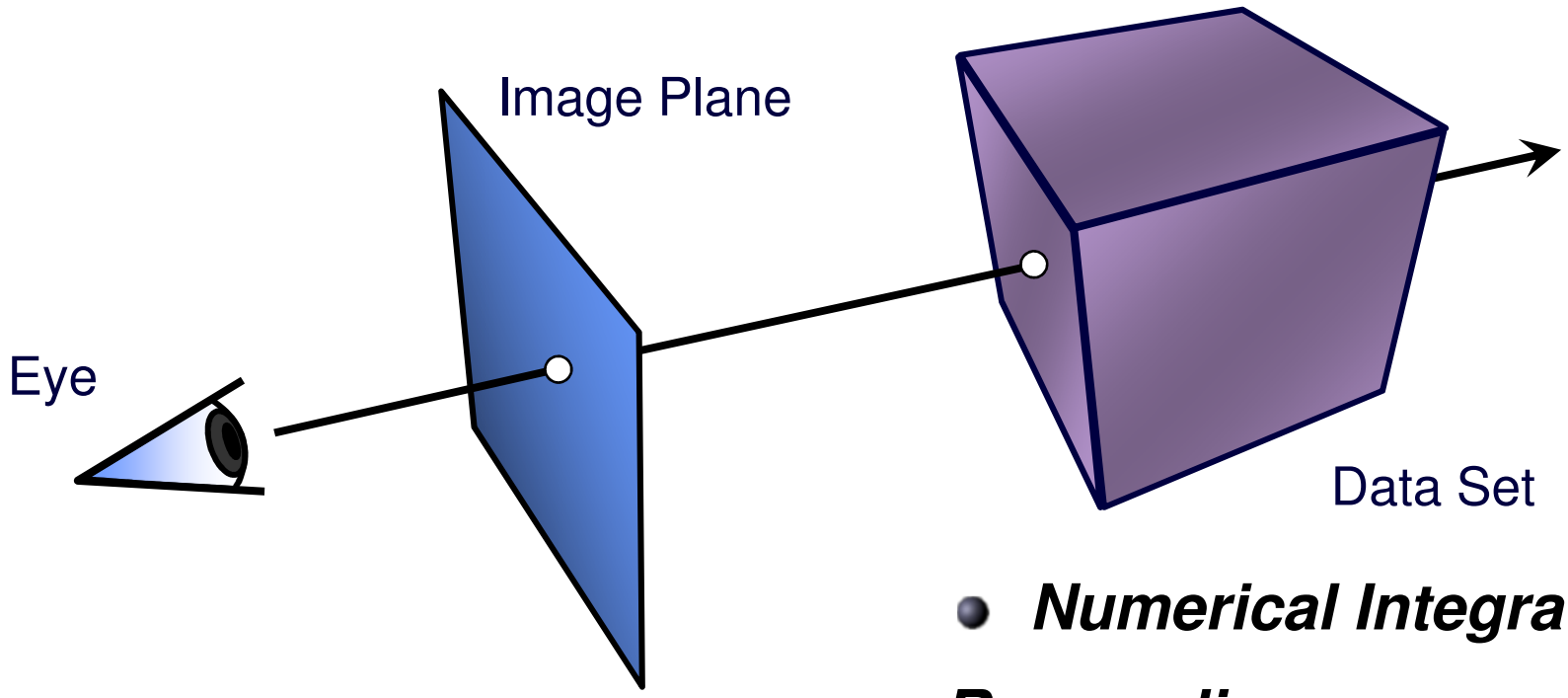
Volume Rendering Outline



Rendering done in real-time on commodity graphics hardware

Ray Casting

- Software Solution

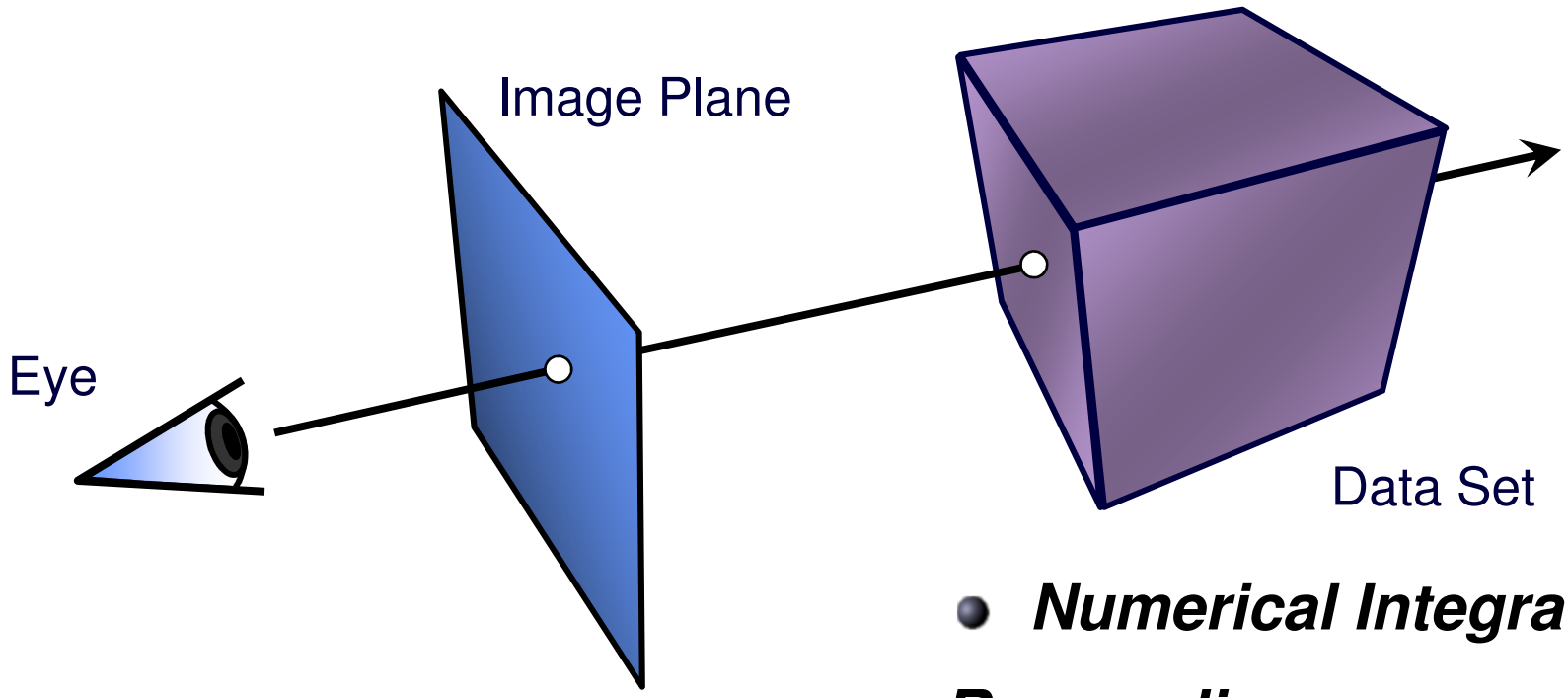


- ***Numerical Integration***
- ***Resampling***

⇨ ***High Computational Load***

Ray Casting

- Software Solution

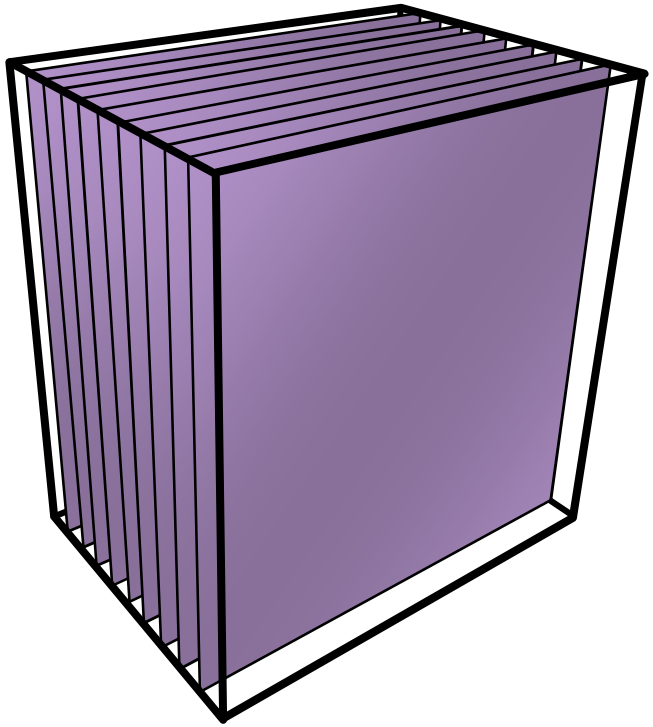


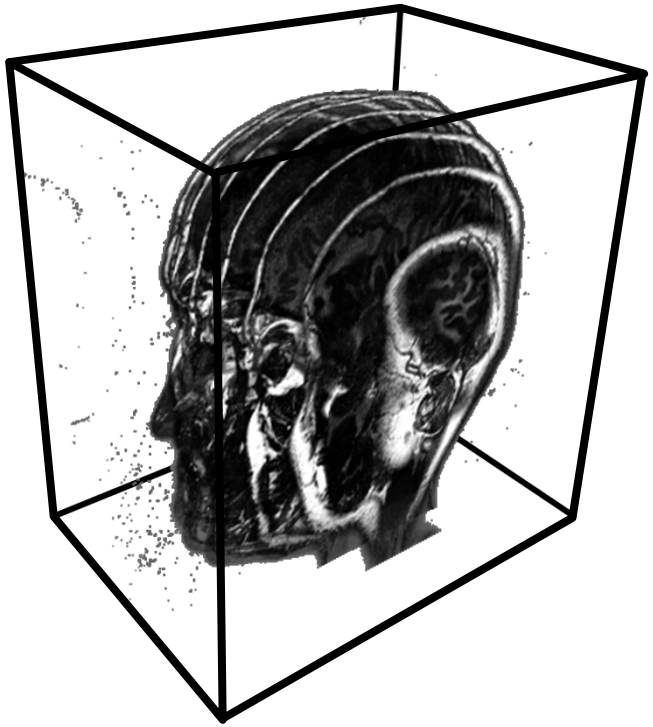
- ***Numerical Integration***
- ***Resampling***

⇨ ***High Computational Load***

Plane Compositing

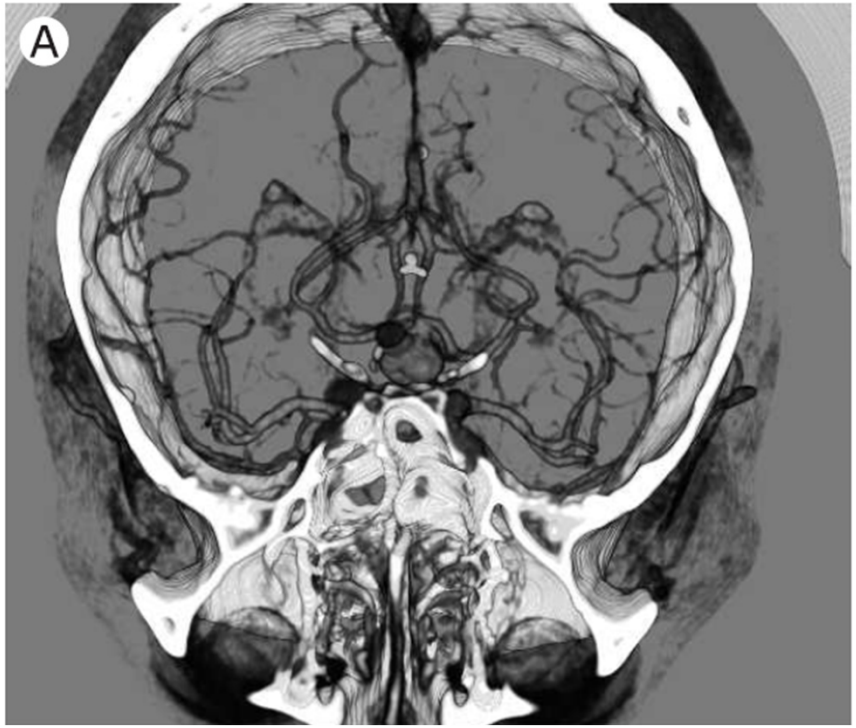
- Proxy geometry (Polygonal Slices)



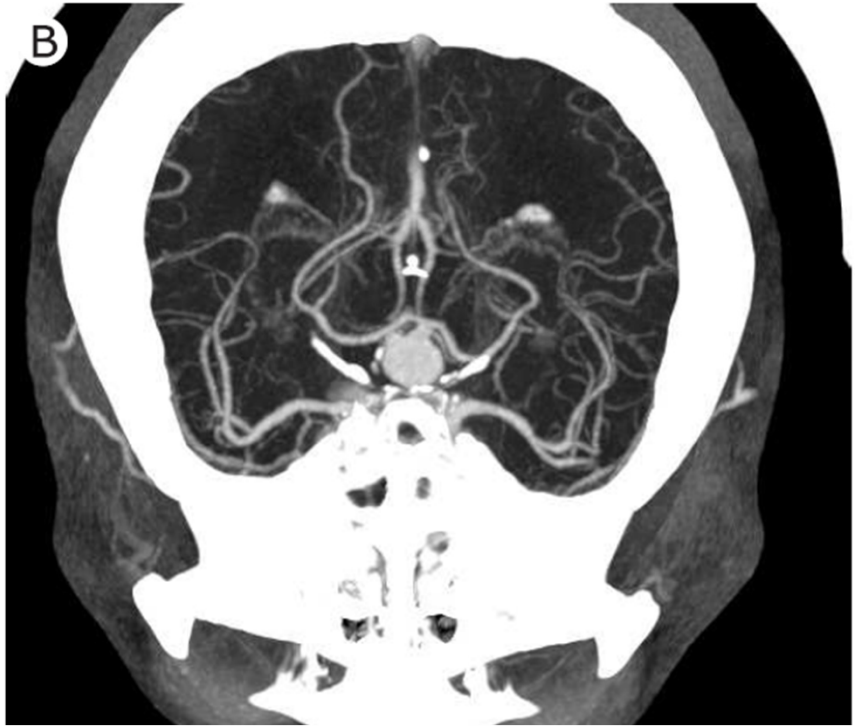


Compositing

- ***Maximum Intensity Projection***
  No emission/absorption
  Simply compute maximum value along a ray



Emission/Absorption

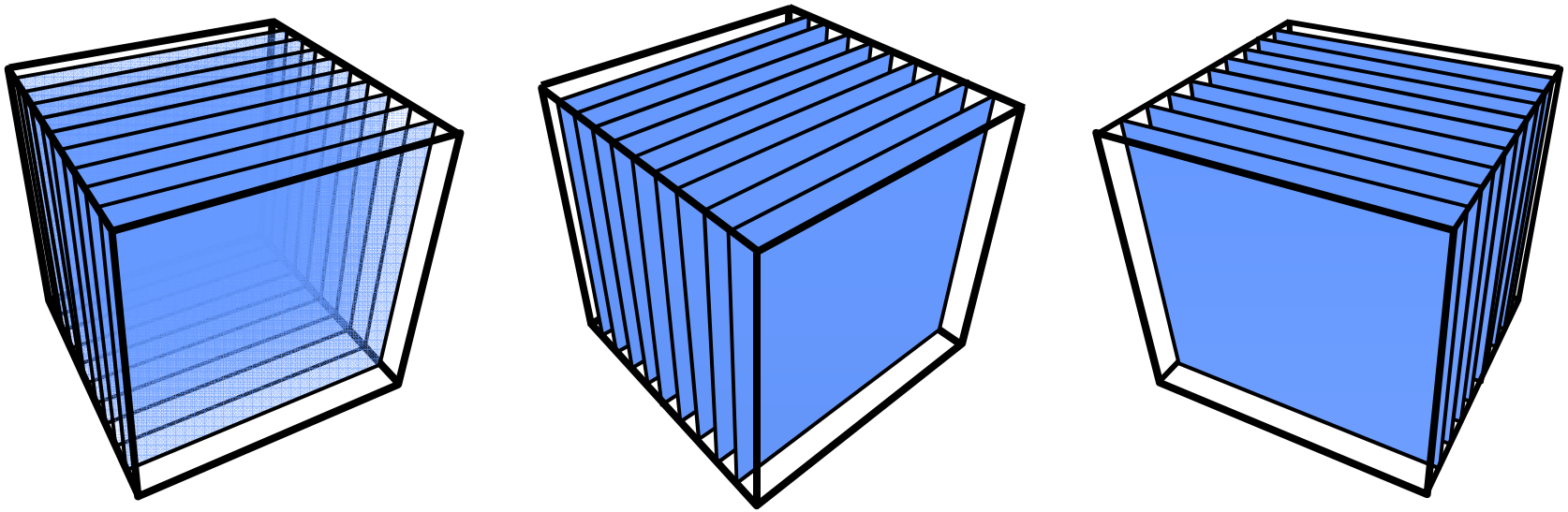


Maximum Intensity Projection

2D Textures

- Draw the volume as a stack of 2D textures
  Bilinear Interpolation in Hardware

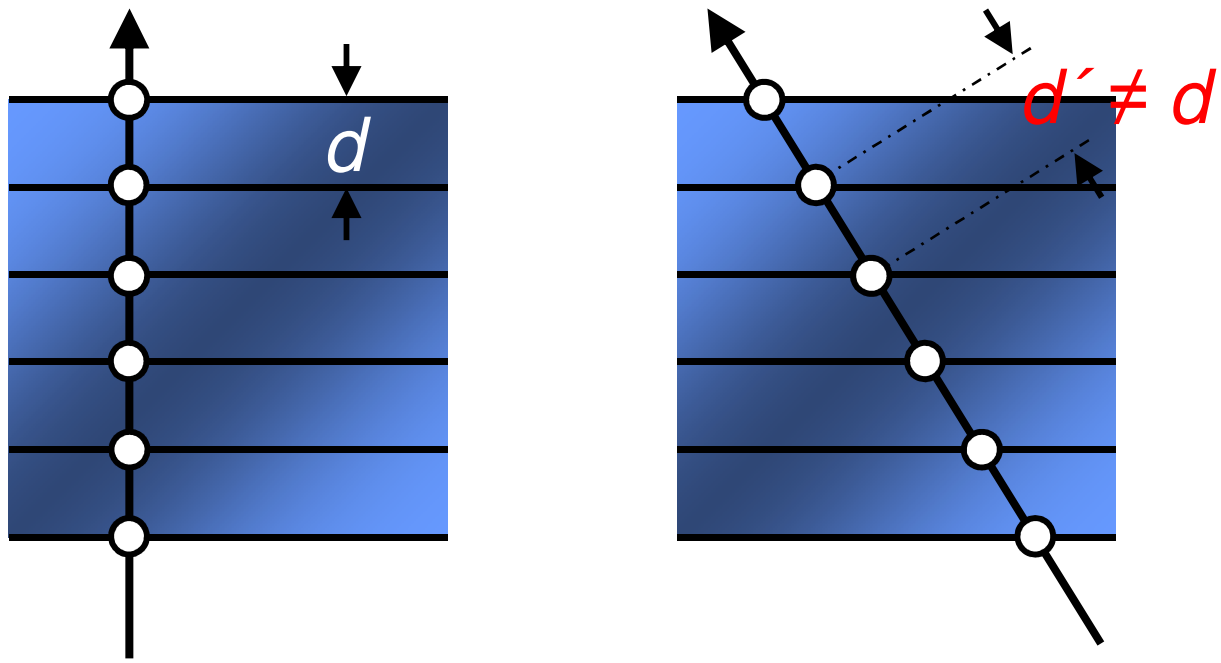
⇨ Decompostition into axis-aligned slices



- 3 copies of the data set in memory

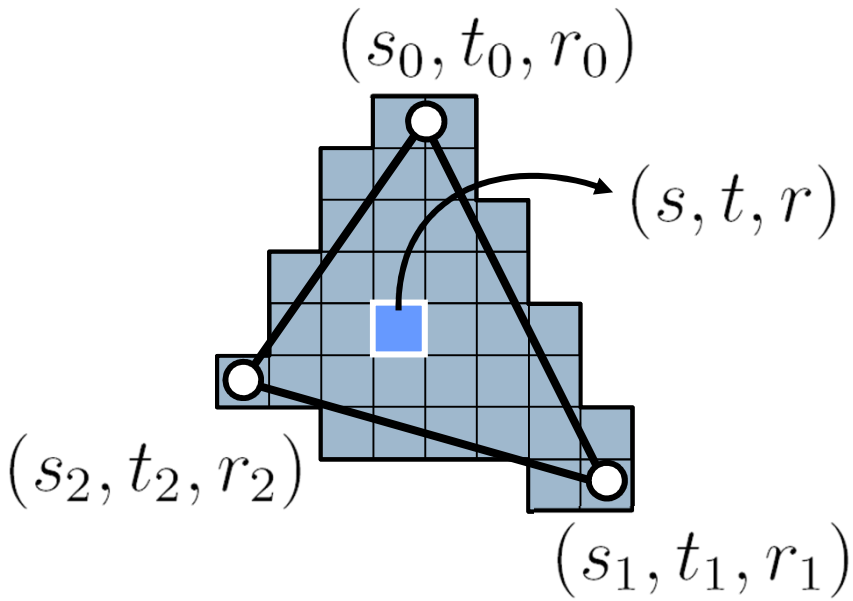
2D Textures: Drawbacks

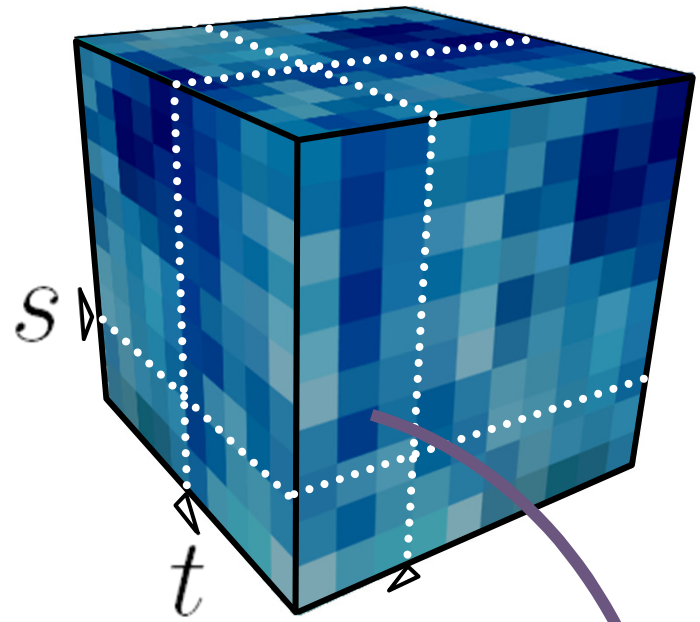
- Sampling rate is inconsistent



- Emission/absorption slightly incorrect
- ***Super-sampling on-the-fly impossible***

3D Textures





For each fragment:
interpolate the
texture coordinates
(barycentric)

Texture-Lookup:
interpolate the
texture color
(trilinear)

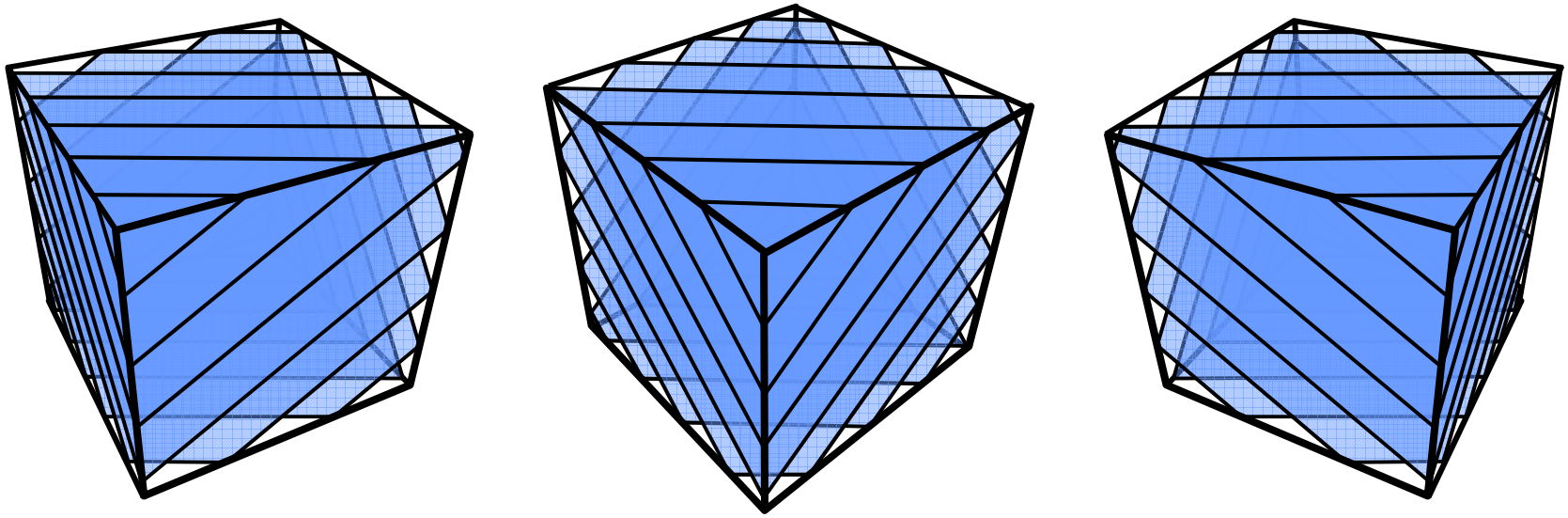
R G B A

3D Textures

3D Texture: Volumetric Texture Object

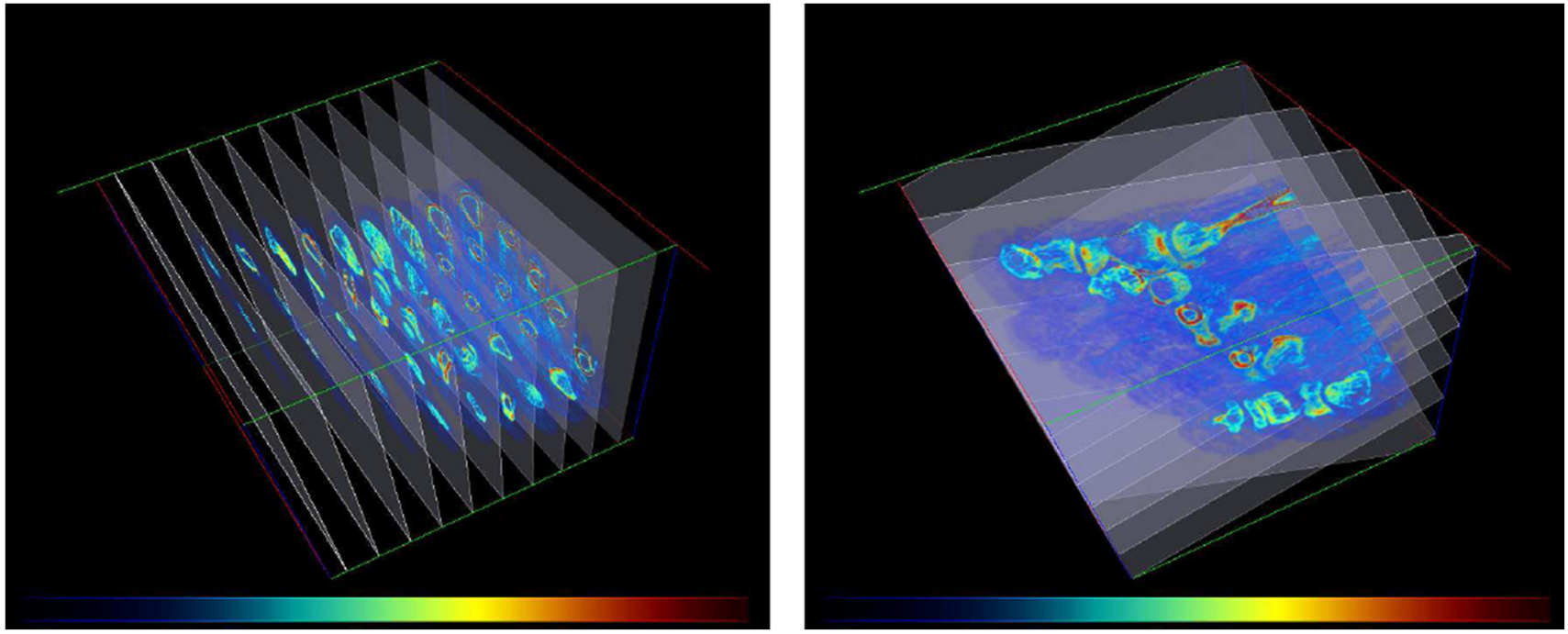
- Trilinear Interpolation in Hardware

⇨ Slices parallel to the image plane



- One large texture block in memory

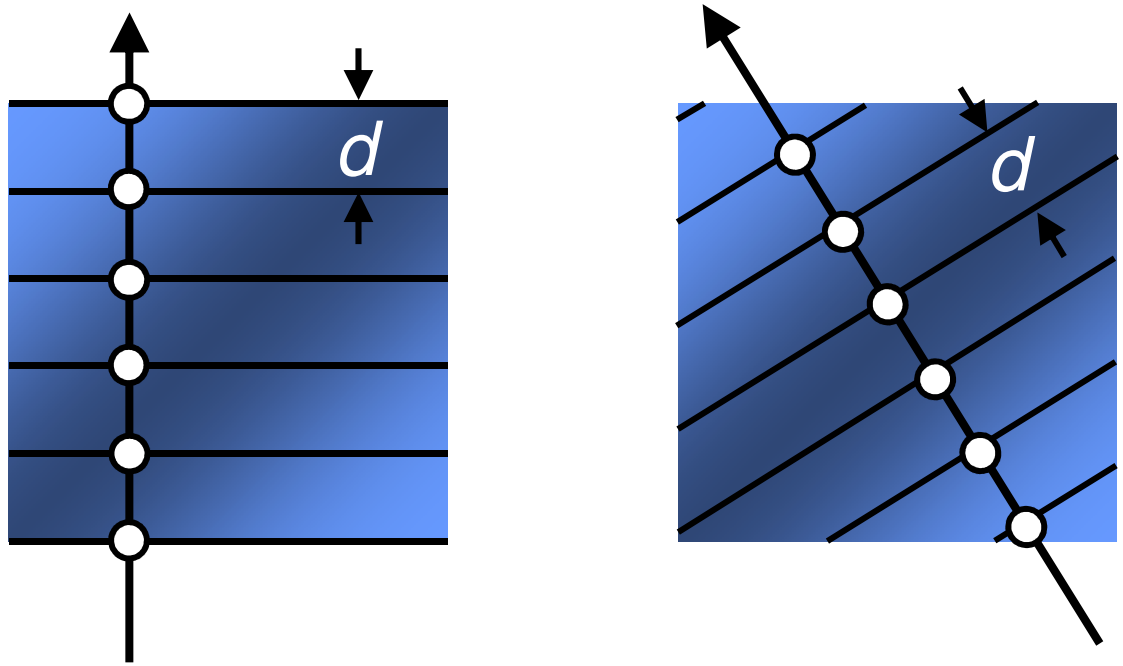
Comparison of 2D with 3D Texturing



Left: 2D textures, right: 3D textures
[Lewiner2006]

Resampling via 3D Textures

- ***Sampling rate is constant***



- Supersampling by increasing the number of slices

Shadows

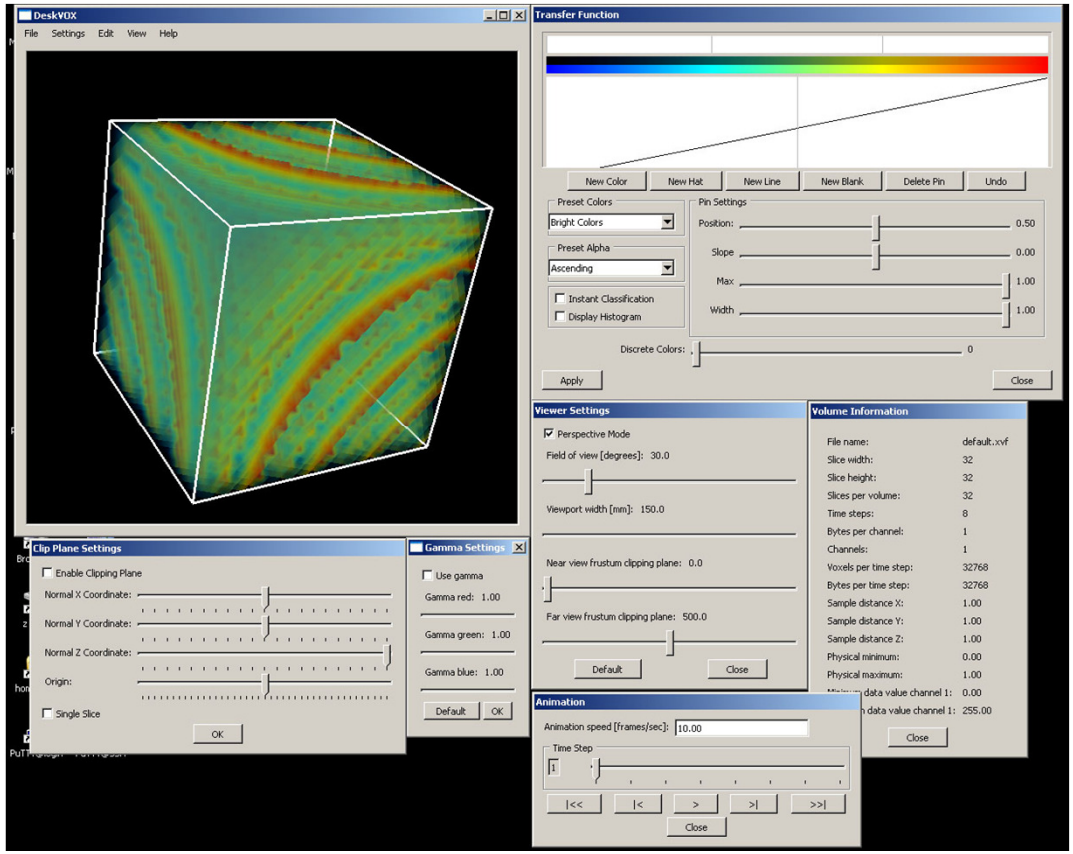
Volume rendering with shadows
(from GPU Gems)

Implementation: Loading a 3D Texture

```
// init the 3D texture
glEnable(GL_TEXTURE_3D_EXT);
glGenTextures(1, &tex_glid);
glBindTexture(GL_TEXTURE_3D_EXT, tex_glid);
// texture environment setup
glTexParameteri( GL_TEXTURE_3D_EXT, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
glTexParameteri( GL_TEXTURE_3D_EXT, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
glTexParameteri( GL_TEXTURE_3D_EXT, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE );
glTexParameteri( GL_TEXTURE_3D_EXT, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE );
glTexParameteri( GL_TEXTURE_3D_EXT, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE );
// load the texture image
glTexImage3DEXT(GL_TEXTURE_3D_EXT, // target
0, // level
GL_RGBA, // color storage
(int) tex_ni(), // width
(int) tex_nj(), // height
(int) tex_nk(), // depth
0, // border
GL_COLOR_INDEX, // format
GL_FLOAT, // type
_texture ); // allocated texture buffer
glPixelTransferi(GL_MAP_COLOR, GL_FALSE);
```

Demo: DeskVox

- DeskVox was created at IVL/Calit2
  - http://ivl.calit2.net/wiki/index.php/VOX_and_Virvo



Videos

- Human head, rendered with 3D texture:
  - http://www.youtube.com/watch?v=94_Zs_6AmQw
- GigaVoxels:
  - http://www.youtube.com/watch?v=HScYuRhgEJw

References

- Volume rendering tutorial with source code
  - http://graphicsrunner.blogspot.com/2009_01_01_archive.html
- Simian volume rendering software
  - http://www.cs.utah.edu/~jmk/simian/