Visual Medicine: Part one - Foundations of Medical Imaging

Rendering and Navigation

Klaus Mueller
Computer Science
Stony Brook University
mueller@cs.sunysb.edu

Gordon Kindlmann
Brigham and Women's Hospital, Boston
gk@bwh.harvard.edu

Overview

Rendering Modes: X-ray, MIP, shaded displays

Basic volumetric rendering: using raycasting

Transfer functions: mapping raw data to visuals

Rendering quality: post- vs. pre-shaded rendering

Controlling rendering effort: occlusions, importance

Rendering acceleration: rendering on GPUs

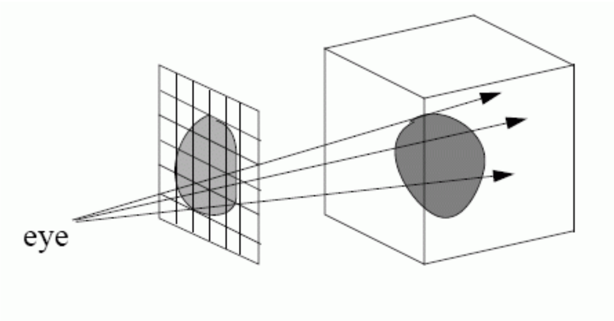
Navigation techniques: virtual colonoscopy

more info in [Kaufman 05]

Raycasting: Concept

Most intuitive rendering technique

- shoot rays into the scene starting from the eye
- the “gold standard” of volume rendering
- use it to derive the fidelity of other paradigms



Volume Rendering Modes: X-Ray

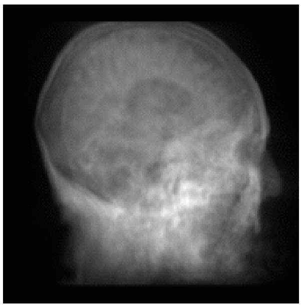
Rays simply sum everything up that falls into their path

- good for first overview, since no data is culled away
- but overdraw and clutter can be a problem

Volume Rendering Modes: X-Ray

Rays simply sum everything up that falls into their path

- good for first overview, since no data is culled away
- but overdraw and clutter can be a problem



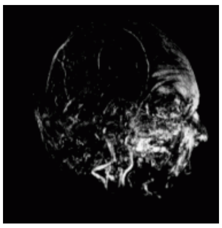

Volume Rendering Modes: MIP (1)

MIP = Maximum Intensity Projection

- a little bit more restrictive than X-ray
- keeps only the maximum value along the ray
- assumes that the maximum value is also the most important
- often used by doctors to get a first look at the data
- great for angiography visualization

legs

head



Volume Rendering Modes: MIP (2)

Various types of MIP [Yen 97] [Shareef 02]

- original MIP (OM)

$$I(p) = \max_{0 \le z \le D}(f(z))$$

- thin slab MIP (TM)

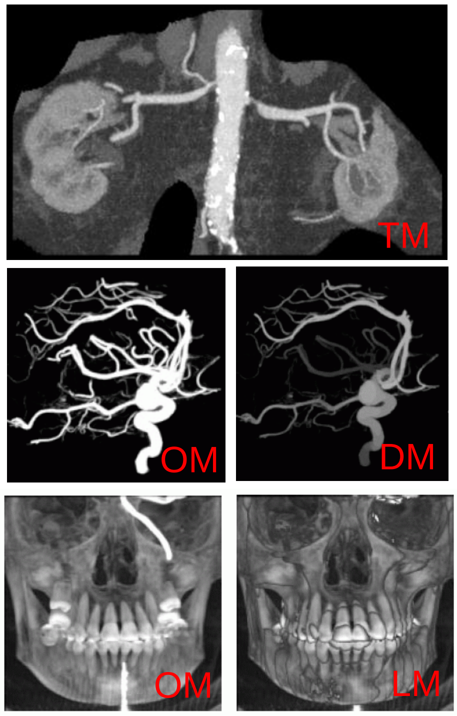
$$I(p) = \max_{d1 \le z \le d2}(f(z))$$

- depth-shaded MIP (DM)

$$I(p) = \max_{0 \le z \le D}(d(z) \cdot f(z))$$

- local maximum MIP (LM)

$$I(p) = \min_{0 \le z \le D}(z| \; f(z) \le t \wedge LMAX(z))$$

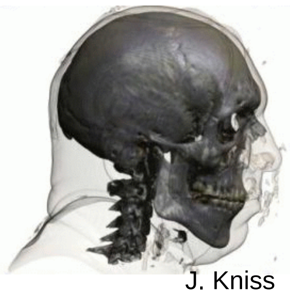


Shaded Volume Rendering

Identify actual surfaces and accentuate them by modeling the reflection of light

- require surface gradients
- need to map densities to color
- this mapping is done via *transfer functions*

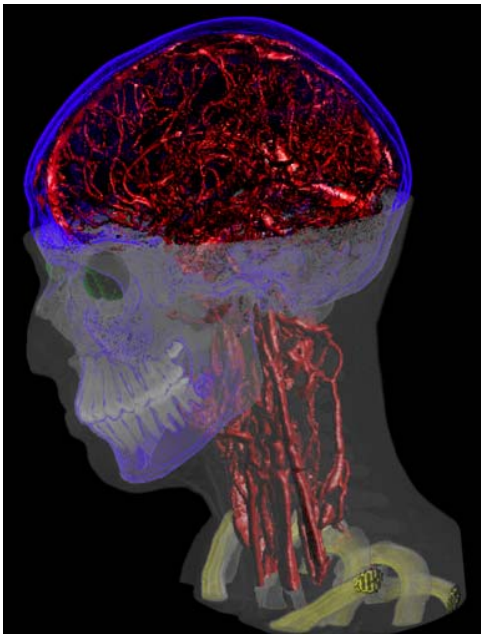
There can be multiple nested surfaces



J. Kniss

- to see all one needs to make front surfaces *semi-transparent* and *composite colors*
- achieve this mapping with a transfer function

All Combined: Two-Level Volume Rendering



Skin and teeth: MIP with different intensity ramps

Blood vessels and eyes: shaded volume rendering

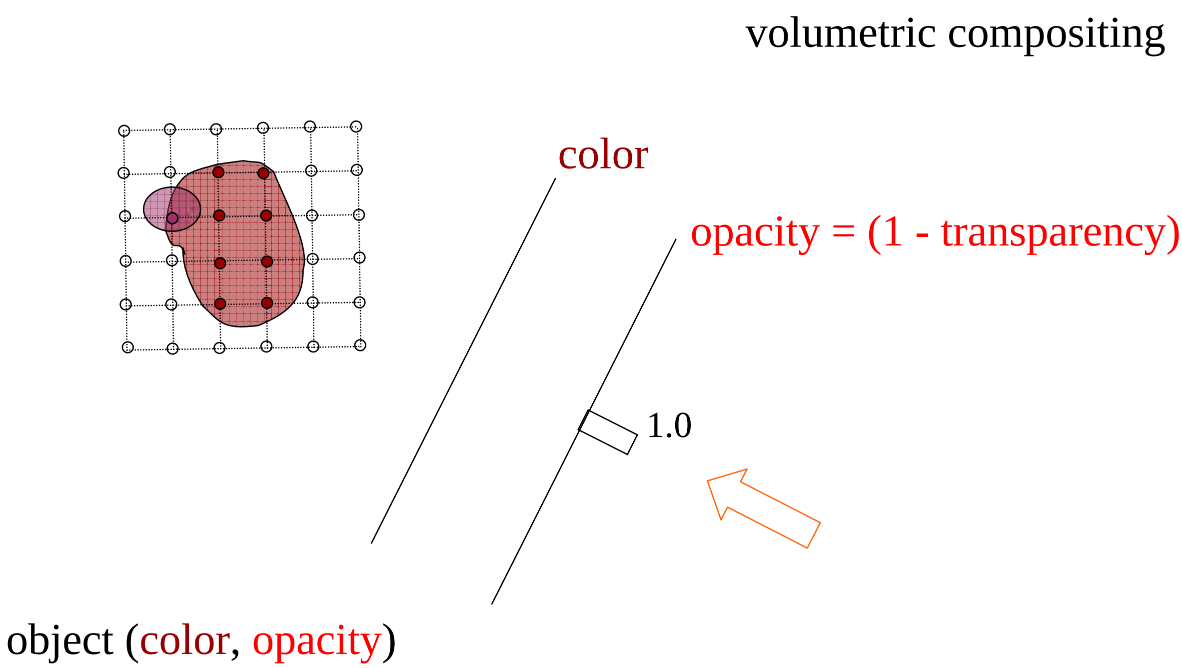
Skull: contour rendering

Vertebrae: gradient magnitude-weighted transfer function with shaded volume rendering

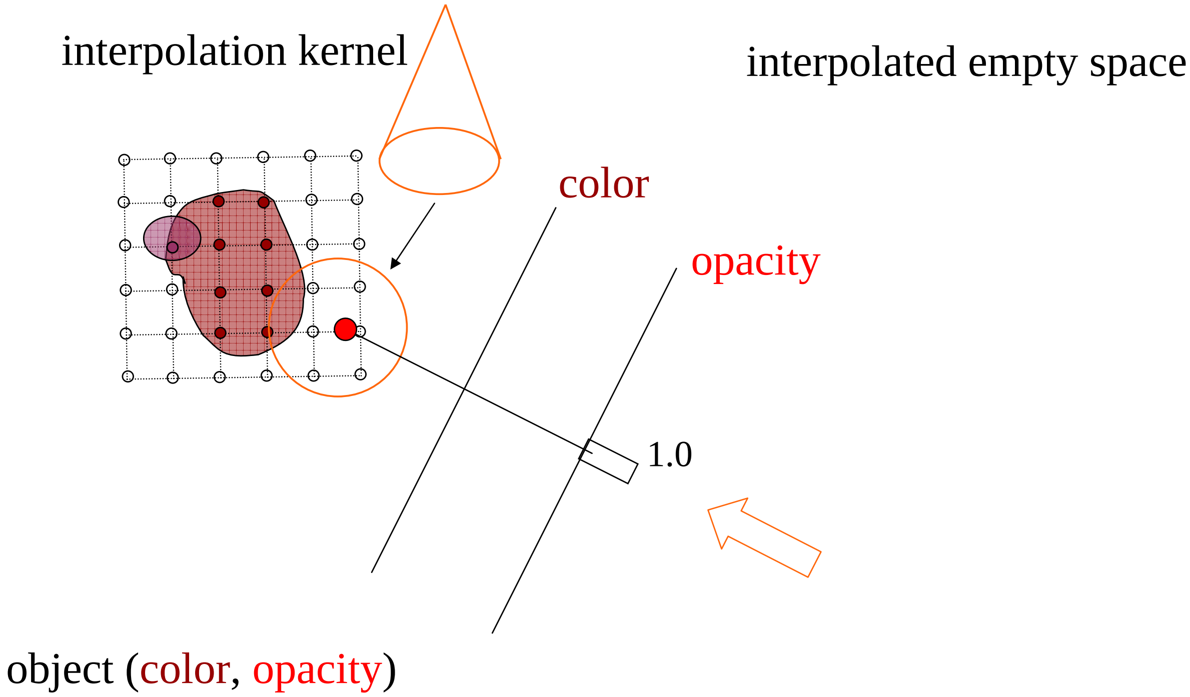
A clipping plane has been applied to the skin object

[Hauser 01]

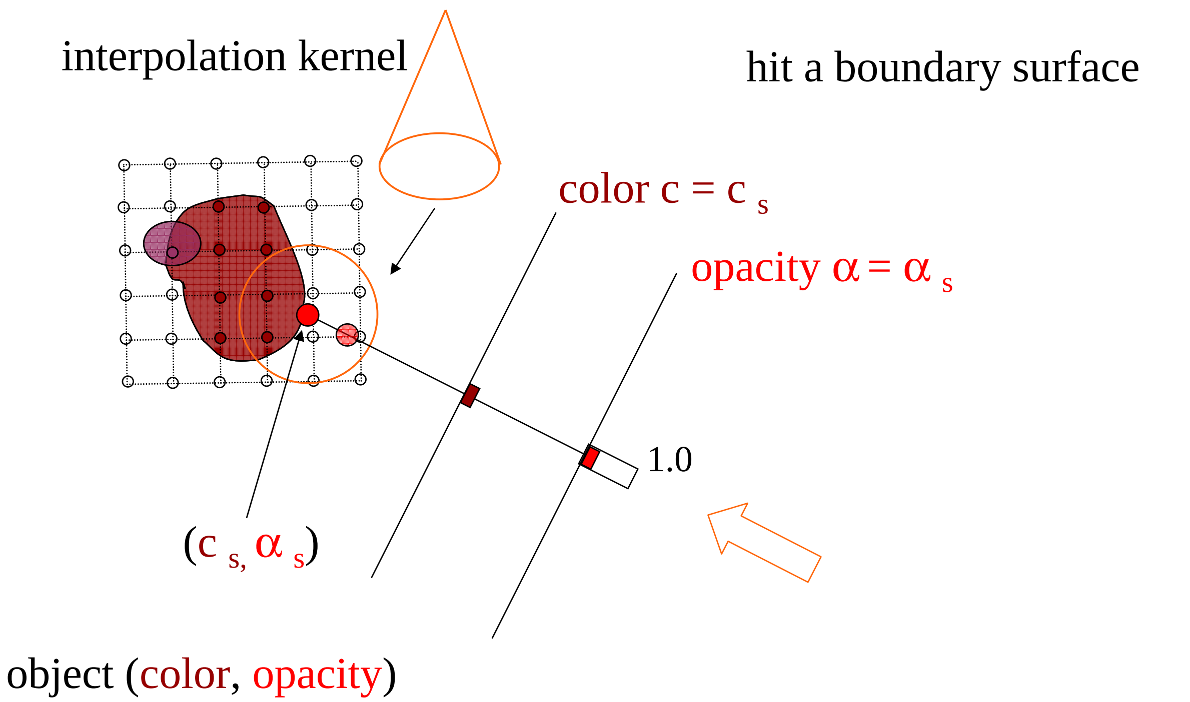
Raycasting Fundamentals



Raycasting Fundamentals



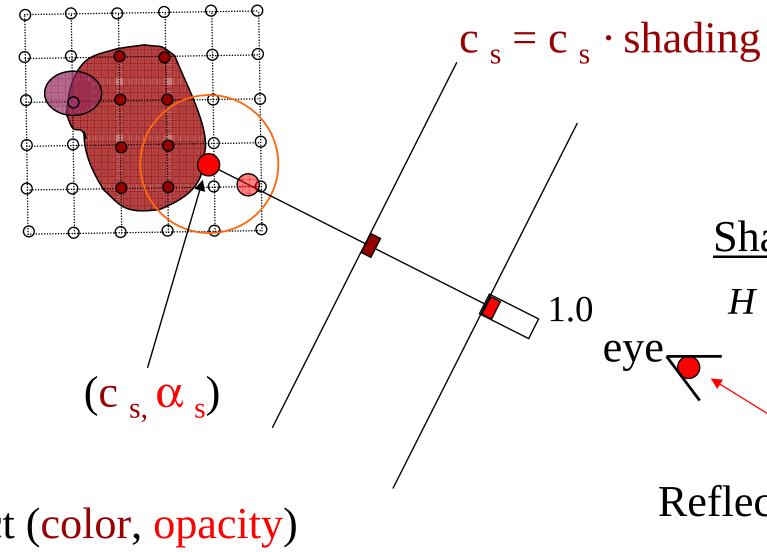
Raycasting Fundamentals

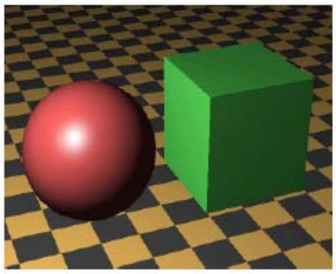


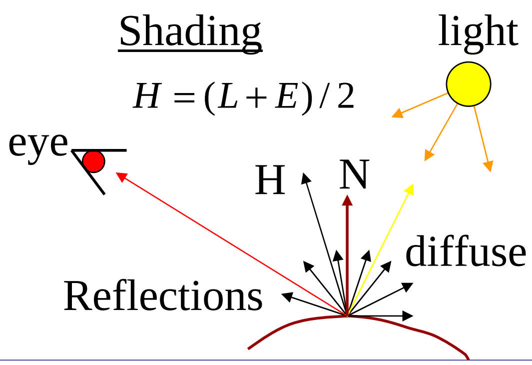
Raycasting Fundamentals

$$shading = c_s(k_a I_a + k_d I_L N \cdot L) + k_s I_L (H \cdot N)^{ns}$$

use unit vectors

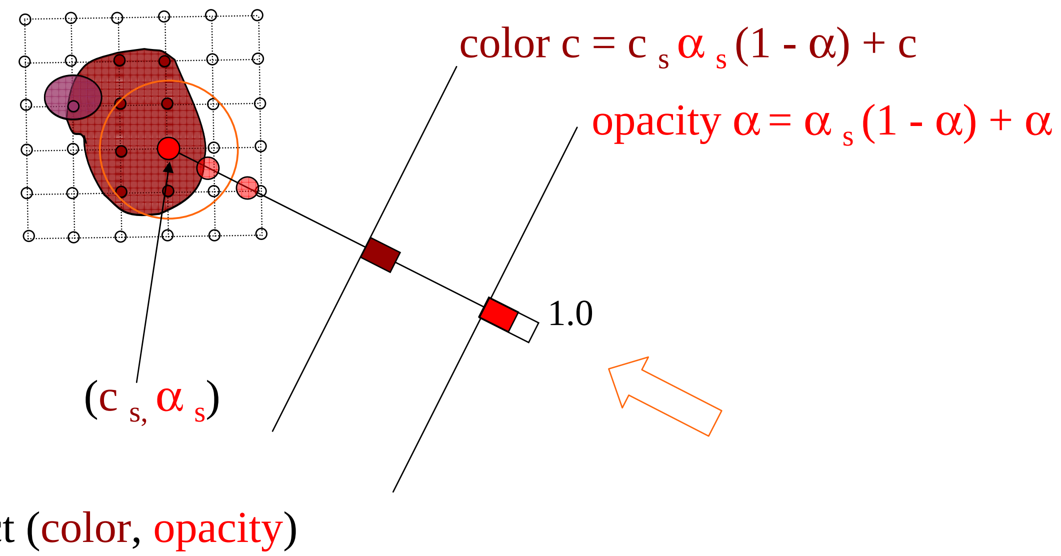






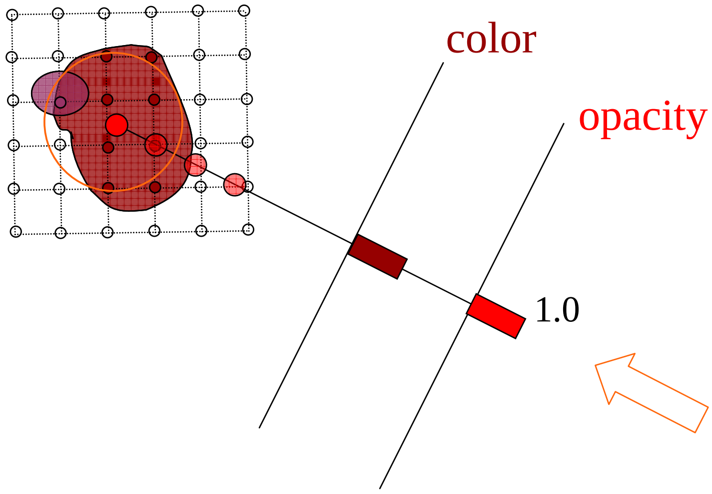
Raycasting Fundamentals

volumetric compositing



Raycasting Fundamentals

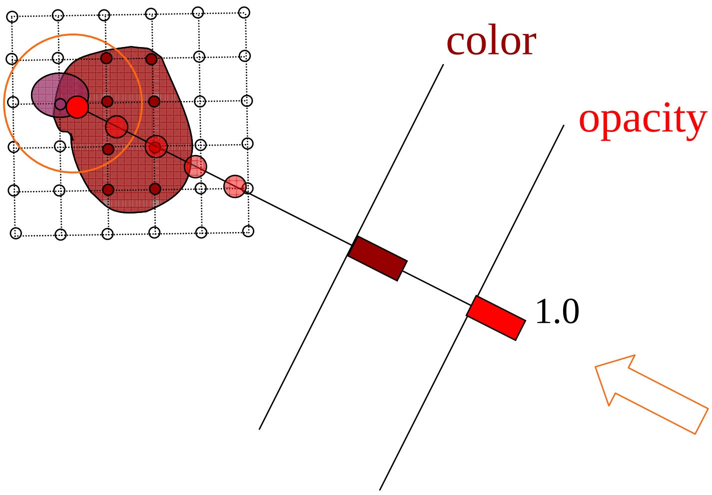
volumetric compositing



object (color, opacity)

Raycasting Fundamentals

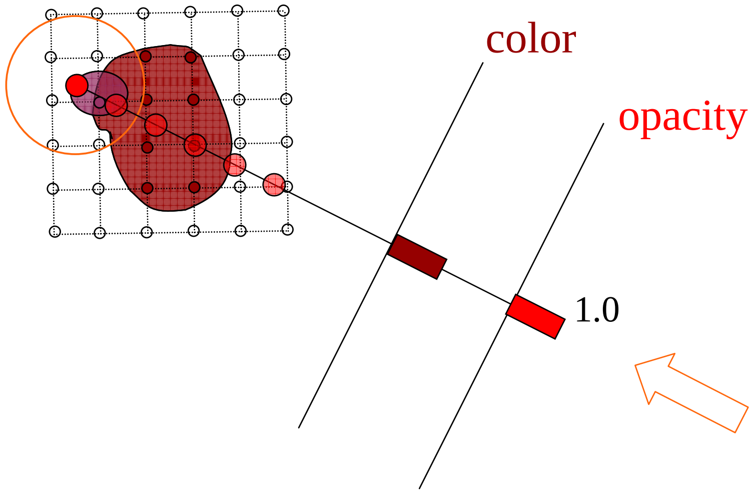
volumetric compositing



object (color, opacity)

Raycasting Fundamentals

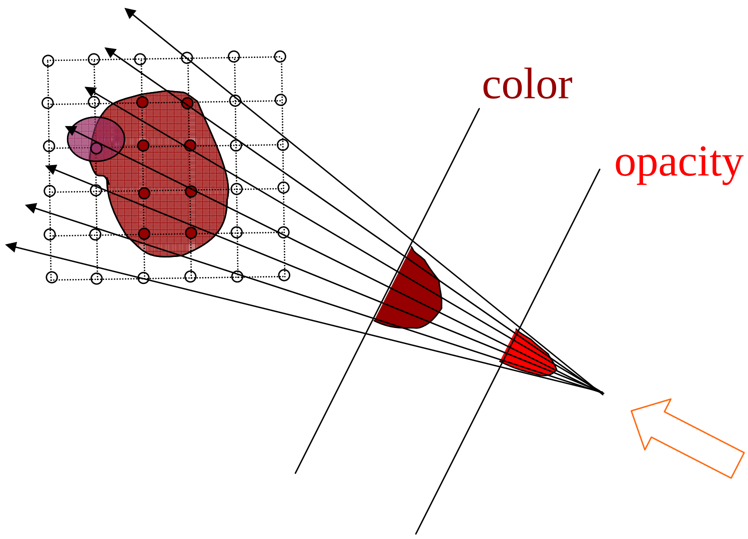
volumetric compositing

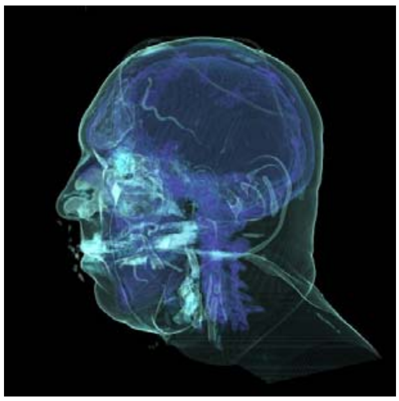


object (color, opacity)

Raycasting Fundamentals

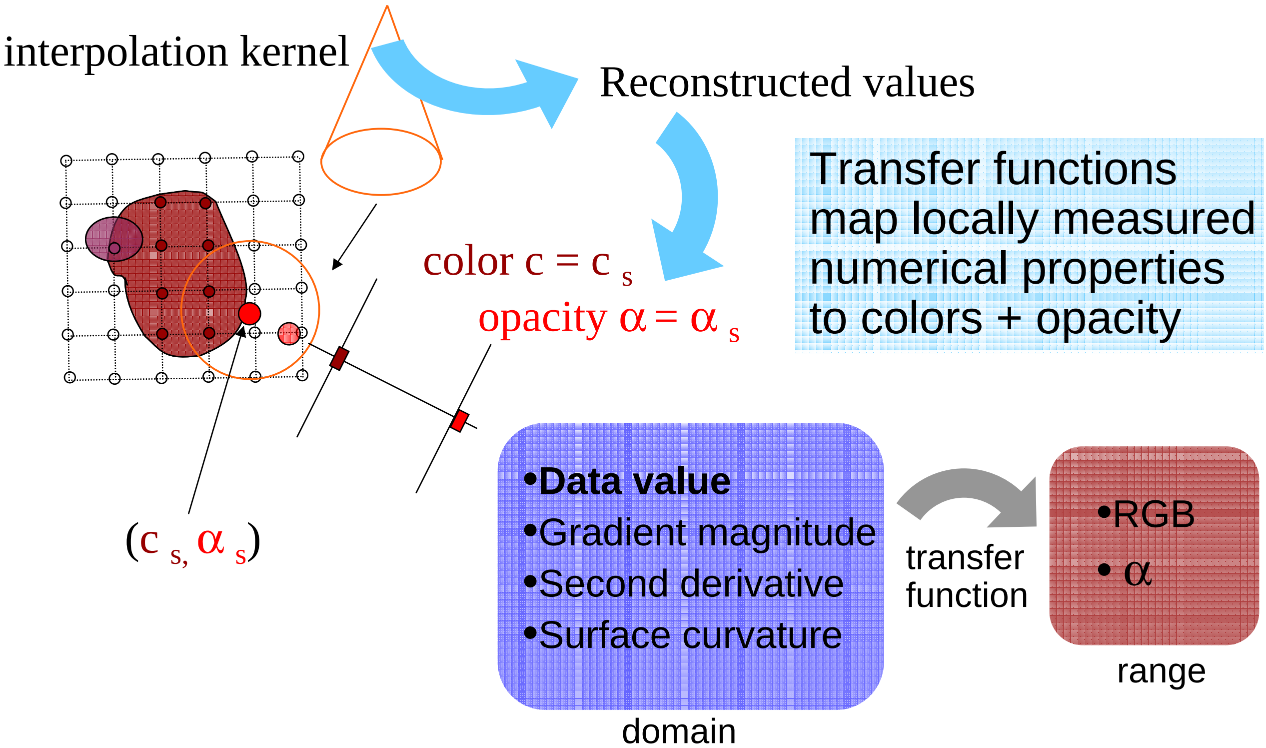
whole image





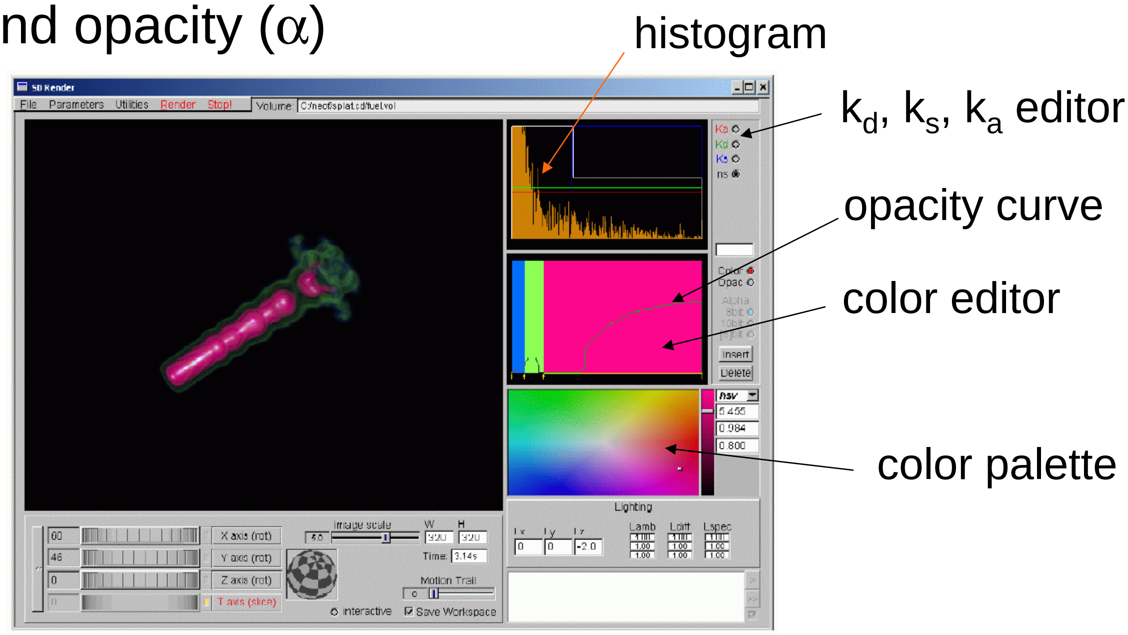
object (color, opacity)

Transfer Functions (1)



Transfer Functions (2)

Map the scalar data (densities) into color (RGB) and opacity (α)

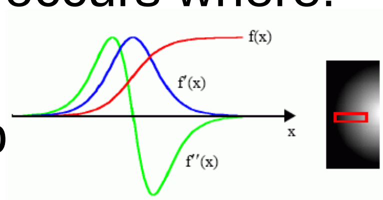


Transfer Functions (3)

Using the gradient information as well can help better surface delineation [Kindlmann 98] [Kniss 01]

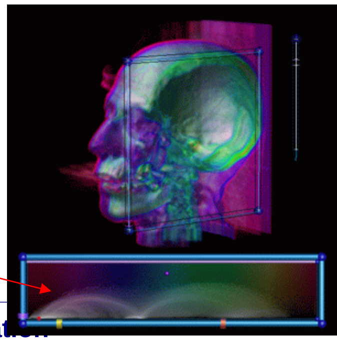
Due to partial volume effect, surface occurs where:

- 1^{st} derivative has a maximum
- 2^{nd} derivative goes through zero

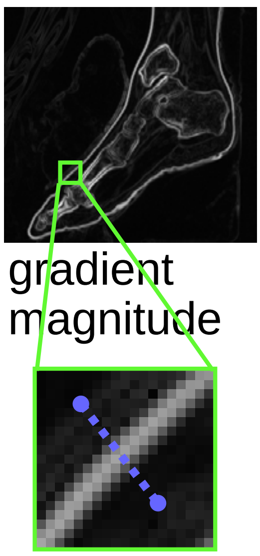


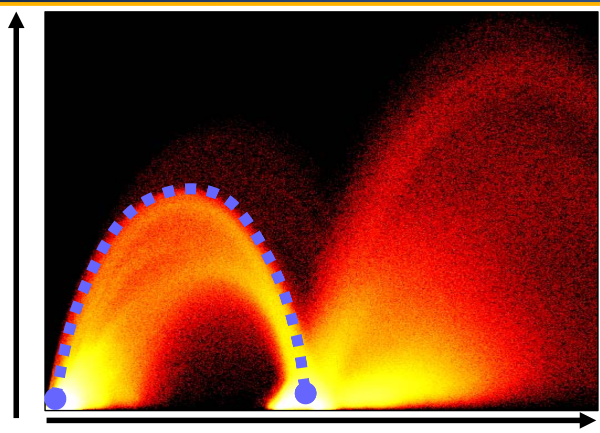
An automatic transfer function generator can assign colors in these areas

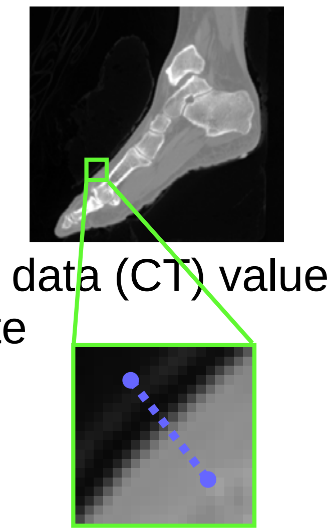
voxel histogram



Interlude: Multi-Dimensional Transfer Functions

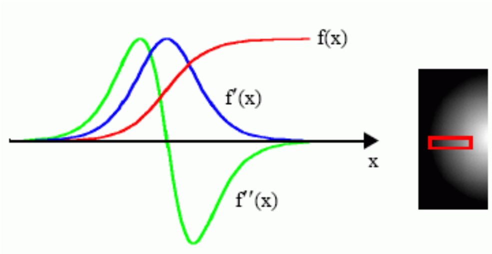


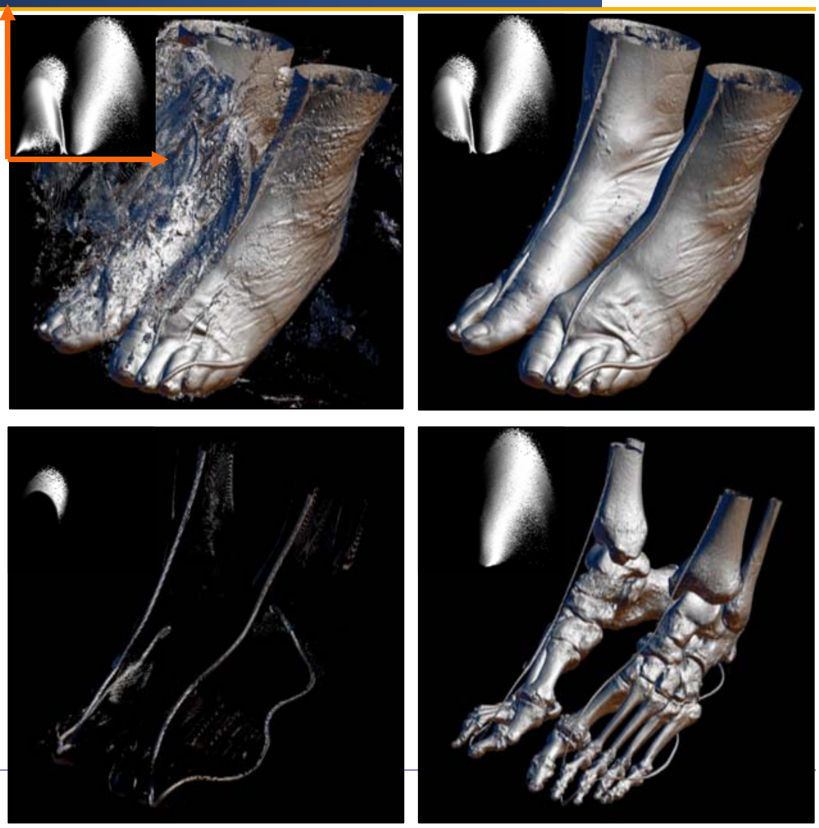




- **Boundaries** in volume create **arches** in (value,gradient) domain [Kindlmann 98]
- Arches guide placement of opacity to emphasize material interfaces [Kniss 01]

Interlude: Multi-Dimensional Transfer Functions





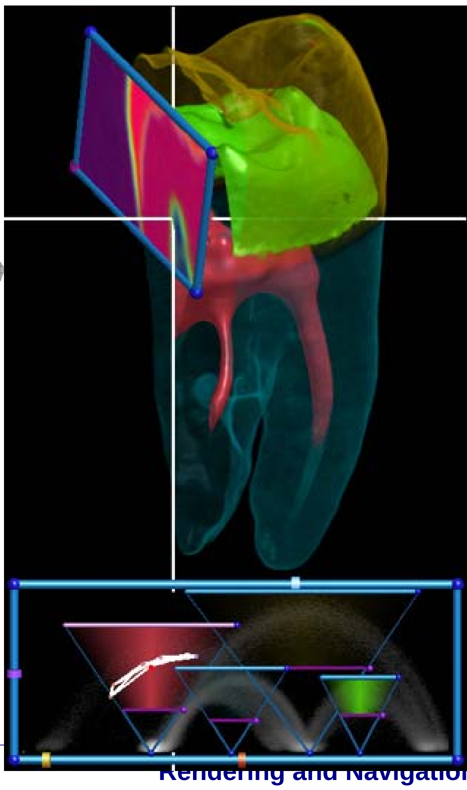
- Boundaries can be described in terms of:
  – maximum in 1st derivative
  – zero-crossing in 2nd derivative
- Semi-automatic classification possible in clean data

Interlude: Multi-Dimensional Transfer Functions

Dual-domain interaction:

[Kniss 01]

Make features opaque by pointing at them



New Rendering

Actions in spatial domain

New transfer function

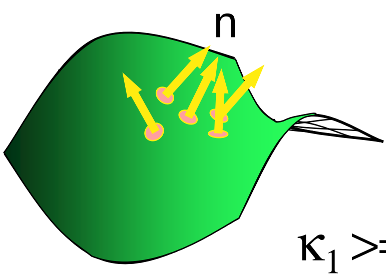
Changes to transfer function

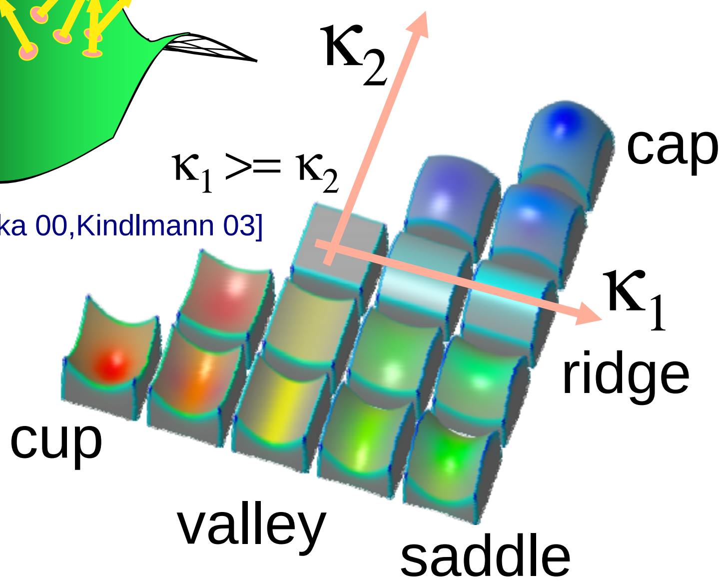
Interlude: Multi-Dimensional Transfer Functions

Curvature: how change in surface position changes surface normal (**n**)

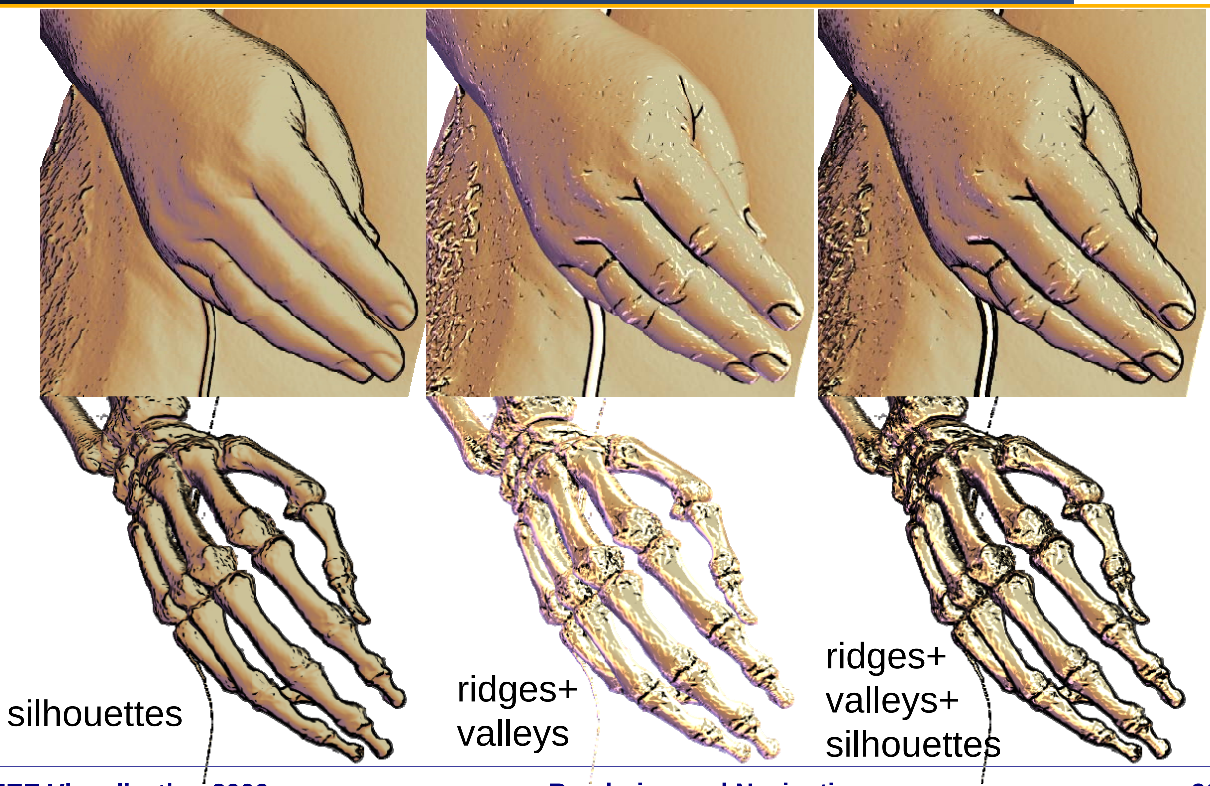
Principal curvatures (κ_1, κ_2) form possible transfer function domain

- Enables surface feature enhancement, better control over silhouettes
- Convolution to measure 1st and 2nd derivatives





Interlude: Multi-Dimensional Transfer Functions



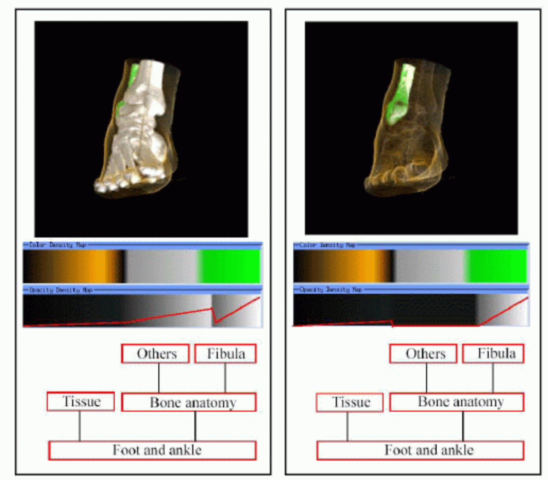
Transfer Functions (4)

For medical visualization it is advisable to keep the interaction with transfer functions at a minimum

Doctors (unlike scientists) do not have the time (nor desire) to play with complex transfer function editors

Better approach: [Mueller 05]

- simplify
- make task-oriented
- automate
- include semantics



Rendering Quality

When to perform shading

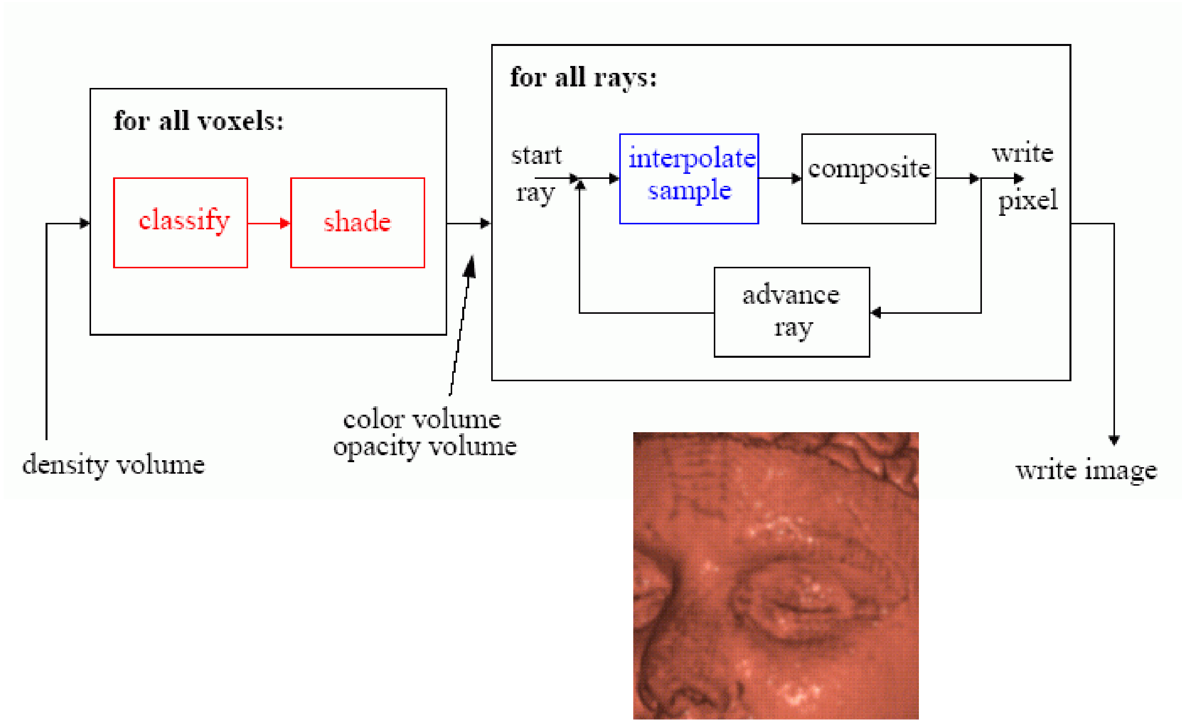
- before interpolation (pre-shaded rendering)
- after interpolation (post-shaded rendering)

Both rendering pipelines have advantages:

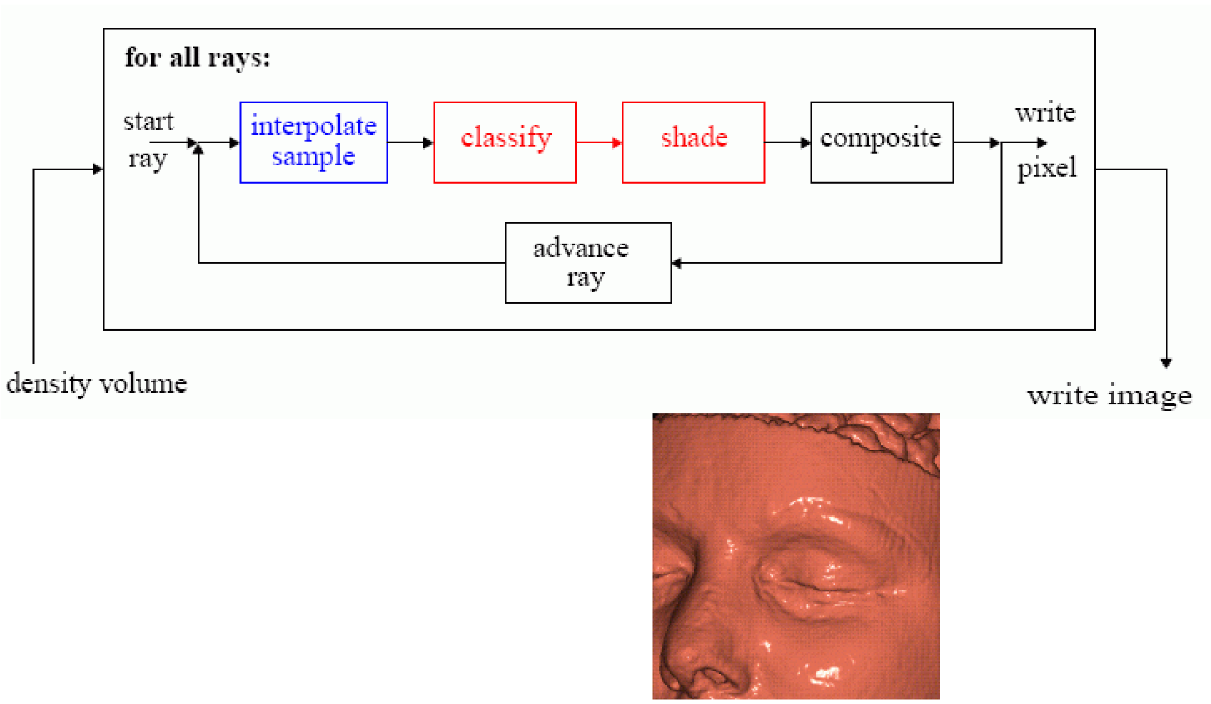
- pre-shaded will not require any further shading during interpolation
- post-shaded will only require shading in visited areas

But there are also qualitative considerations [Mueller 99]

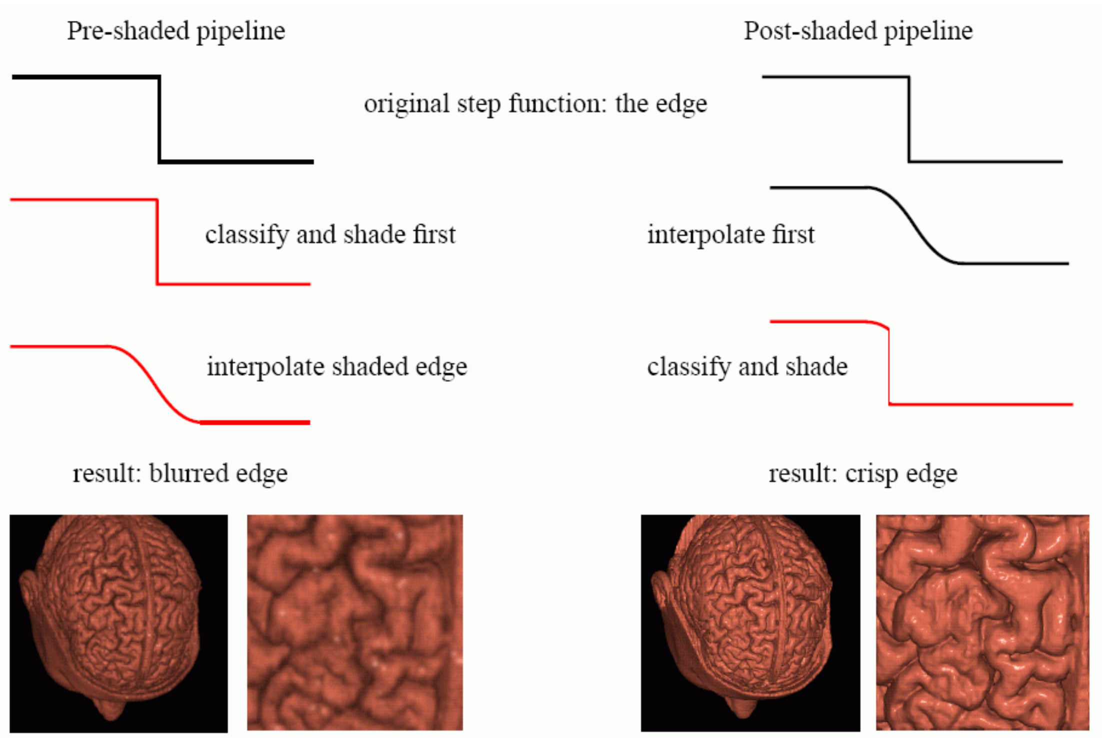
Pre-Shaded Rendering Pipeline



Post-Shaded Rendering Pipeline



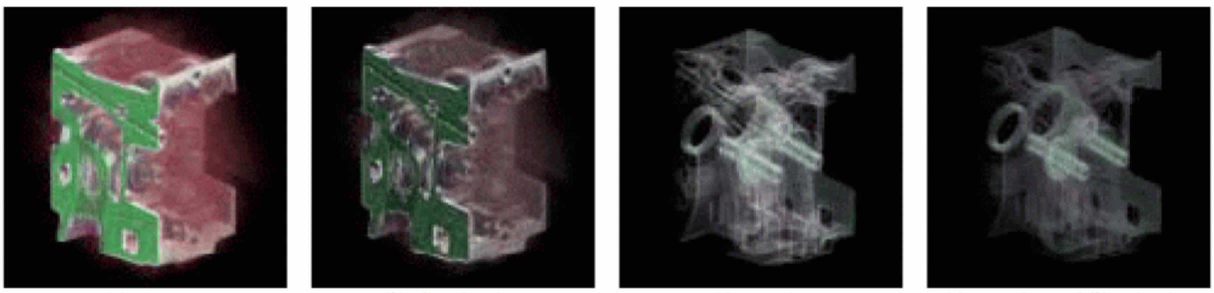
Pre- vs. Post-Shaded Rendering



Rendering Quality: Gradient Modulation

To further accentuate surfaces, multiply either color or opacty or both by the gradient magnitude

- done in the multi-dimensional transfer functions, but discovered much earlier and implemented in the VolumePro board [Pfister 99]



none opacity illumination both

Controlling Rendering Effort (1)

Need to be wise about:

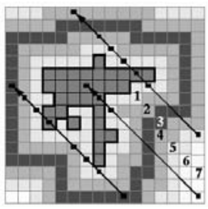
- cache management -- cache faults are expensive
- allocation of rendering effort -- don't spend time on visual effects that are not noticeable
- detail management -- concentrate rendering effort on areas that are in focus and are important

The latter may require semantics gathered in the segmentation and classification

Controlling Rendering Effort (2)

Use early ray termination ($\alpha < 0.9$)

- β-accleration [Danskin 92] speeds rays as they get more opaque



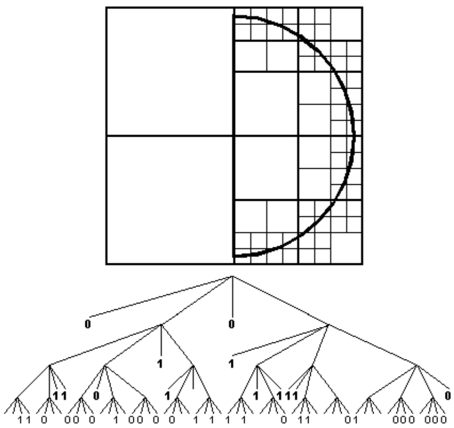
Space leaping

- skip empty space outside object
- schemes may be static for fixed transfer functions (proximity clouds [Cohen 94], distance fields [Hong 97] [Srámek 00])
- or schemes may be dynamic using space-decompositions, such as octrees

Controlling Rendering Effort (3)

Hierarchical space decomposition

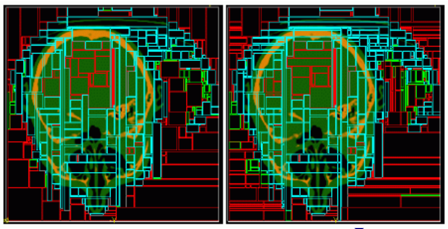
- for each node, store min-max density values
- skip node if transfer function is zero in interval
- else, decend down the tree, recurse [Grimm 04]



Irregular space decompositions

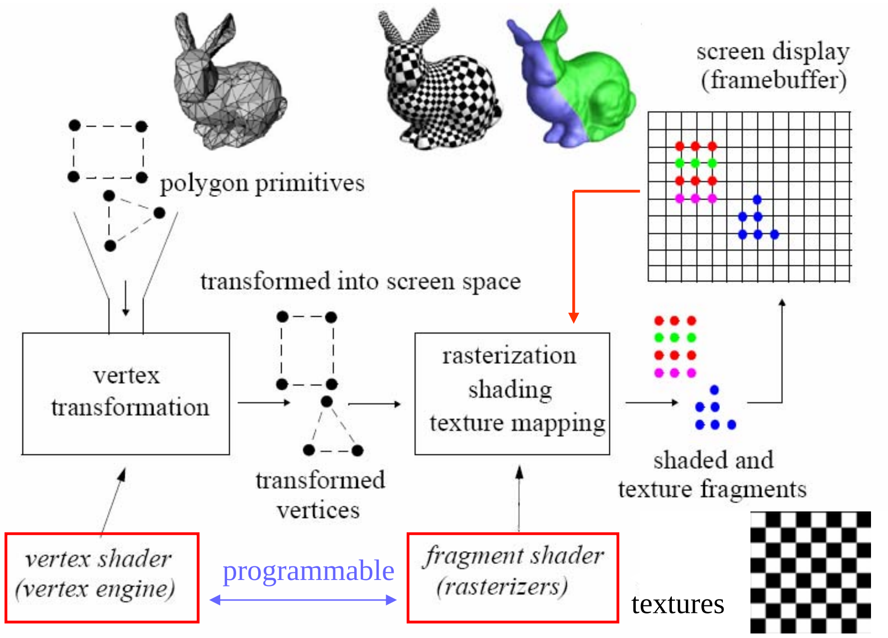
- BSP-trees, kd-trees, etc.

[Li 03]



Accelerated Rendering on GPUs

stream of (X,Y,Z,W)

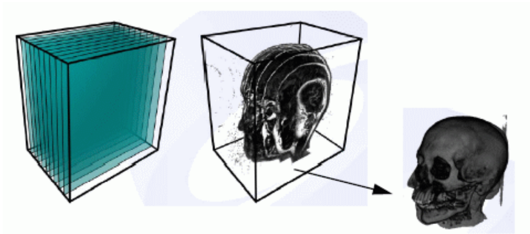


stream of (R,G,B,A)

Accelerated Rendering on GPUs: Concept

Simplest approach [Rezk-Salama 01]

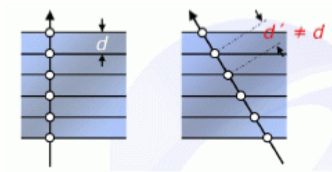
- represent the volume as a stack of axis-aligned “proxy polygons”
- texture-map volume slices onto corresponding proxy polygons
- render polys to screen, properly shifted according to viewing direction, front-to-back
- shade and composite slice by slice



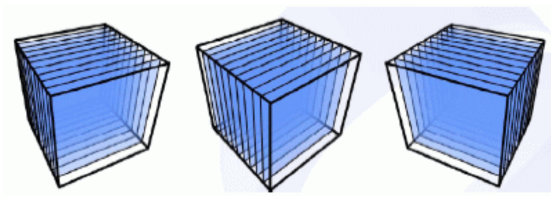
Accelerated Rendering on GPUs: Main Issues

There are two main disadvantages with this approach:

- sampling distance *d* is larger than 1.0 for off-axis viewing directions



- need three stacks of volumes, one for each major viewing direction

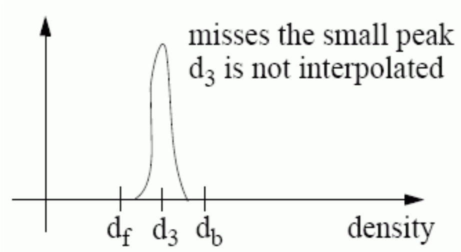


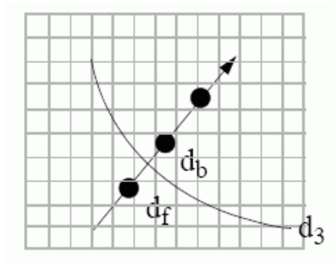
from: Engel, Kraus, Ertl, Graphics Hardware Workshop 2001

Interlude: Pre-Integrated Volume Rendering (1)

Designed to overcome artifacts due to:

- too large sampling intervals (rays, 2D textures)
- transfer functions with high frequency features, which may not be captured by two consecutivly interpolated densities





Interlude: Pre-Integrated Volume Rendering (2)

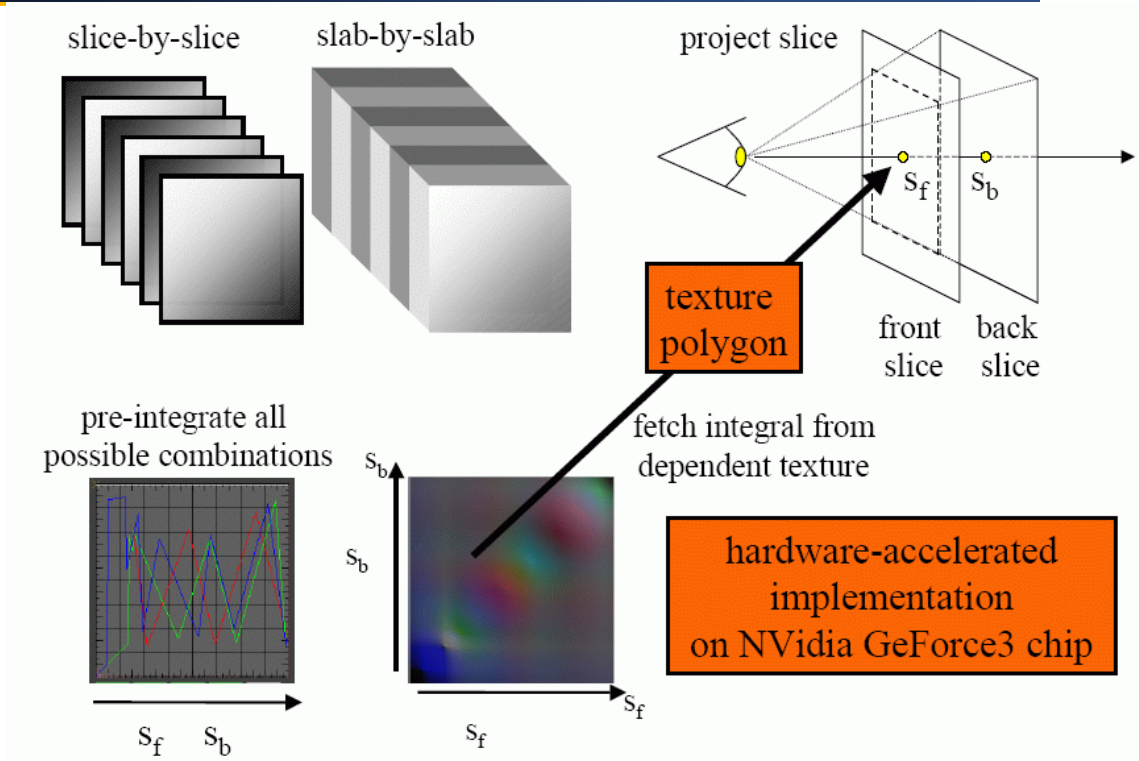
Solution:

- pre-compute color and opacity integrals for all possible front- and back density pairs, d_f and d_b
- gives rise to a 2D table, indexed by interpolated d_f and d_b, assuming piecewise linear densities:

$$c(d_f, d_b) = \int_0^1 \mu((1-s)d_f + sd_b) \cdot c((1-s)d_f + sd_b) e^{-\int_0^s \mu((1-s)d_f + sd_b)dt} ds$$

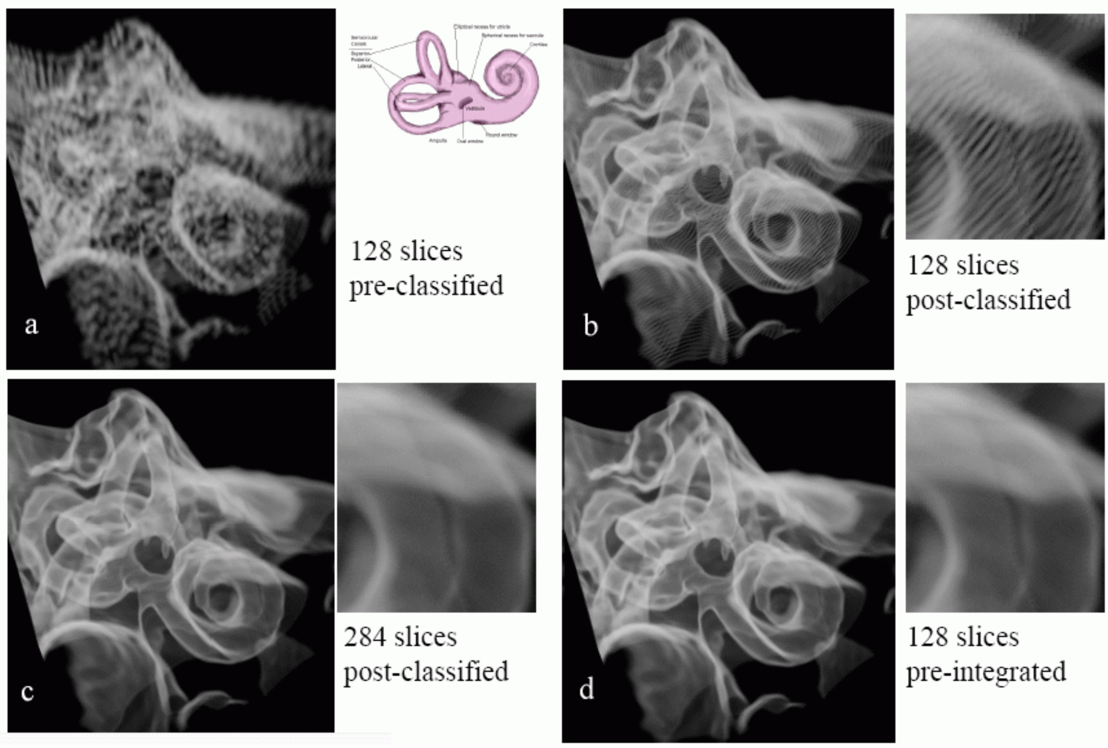
- opacities compute similar

Interlude:
Pre-Integrated Volume Rendering (3)



from: Engel, Kraus, Ertl Graphics Hardware Workshop 2001

Interlude:
Pre-Integrated Volume Rendering (4)

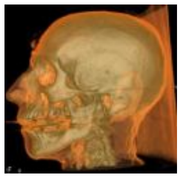


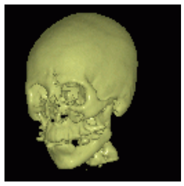
from: Engel, Kraus, Ertl Graphics Hardware Workshop 2001

Accelerated Rendering on GPUs: Conclusions

Using the texture mapping hardware approach allows interactive frame rates with practical-sized datasets

More advanced GPU-based renderer offer:





- raycasting (more natural than textures)
- empty-space skipping [Stegmaier 05] [Leung 06]
- early ray termination, occlusion culling
- advanced rendering effects (shadows, translucencies, advanced lighting, etc.)

These offer advantages in speed, quality, flexibility

Navigation Techniques

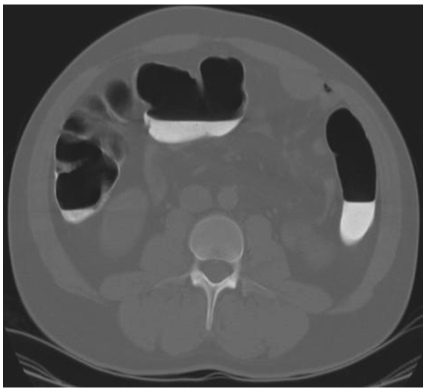
Interactive viewing is key to medical volume exploration

Navigation aids are important as well

- users need to receive some guidance during the exploration of possibly large data
- perceptual studies have shown that humans can only keep a limited amount of information in working memory

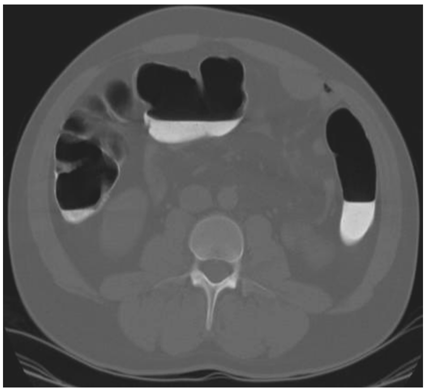
The following case study - Virtual Colonoscopy - unifies real-time exploration with navigation aids

Virtual Colonoscopy: Data Generation and Preparation

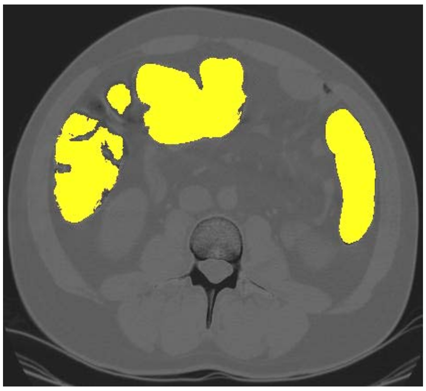


3D Colon - acquired via helical CT

Data Generation and Preparation

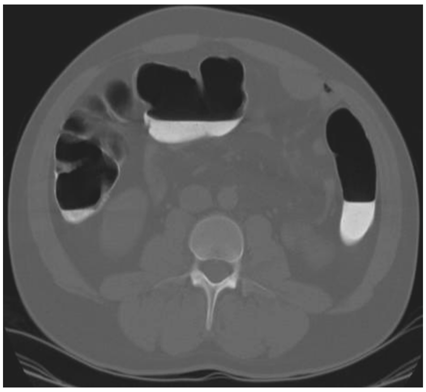


→ Segmentation (3D region growing)

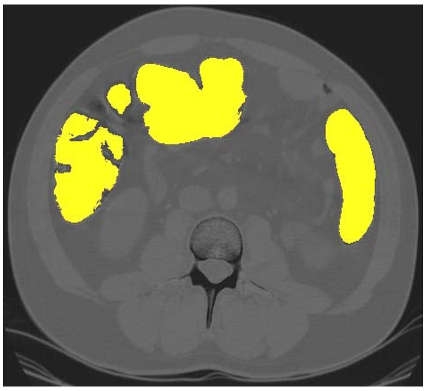


3D Colon - acquired via helical CT

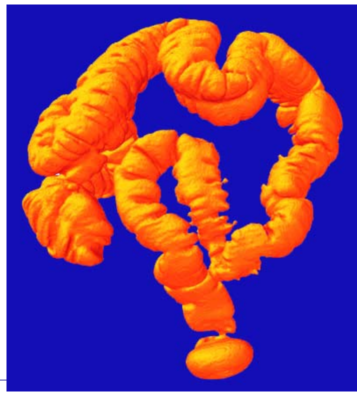
Data Generation and Preparation



→ Segmentation (3D region growing)

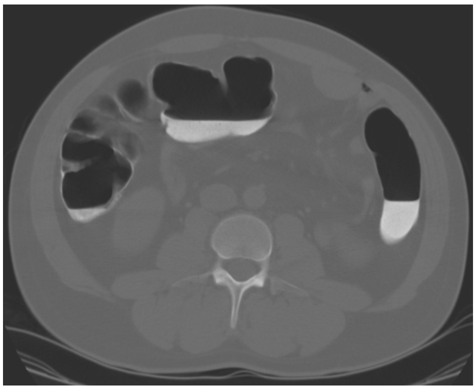


→ Cull remainder

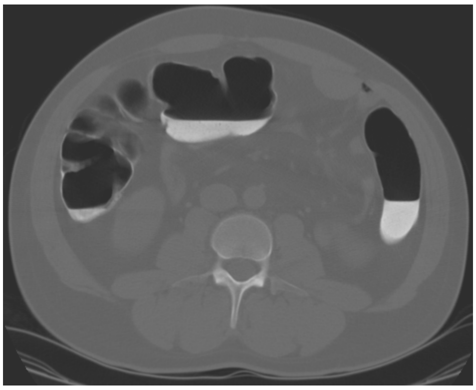


3D Colon - acquired via helical CT

Data Generation and Preparation



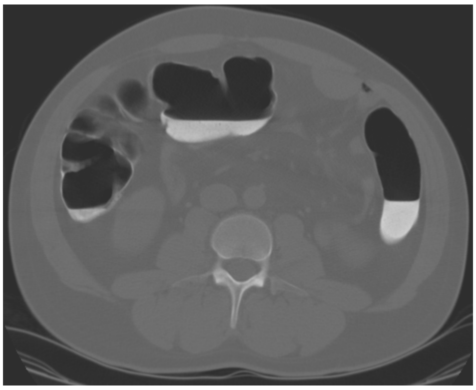
Data Generation and Preparation



→ Remove tagged fluid

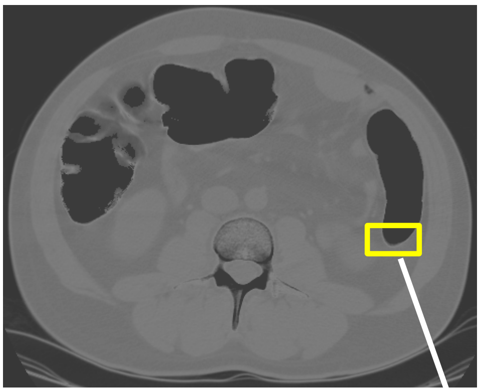
[Lakare 00]

Data Generation and Preparation



→ Remove tagged fluid

[Lakare 00]



Reconstruct smooth surface under fluid

[Lakare 03]

Virtual Colonoscopy: Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)

Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)
- 3D visualization (most appropriate)

Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)
- 3D visualization (most appropriate)

3D visualization paradigms (use colon as example)

- Section colon into straight pieces, slice in the center, and visualize on a virtual tray

Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)
- 3D visualization (most appropriate)

3D visualization paradigms (use colon as example)

- Section colon into straight pieces, slice in the center, and visualize on a virtual tray
- Unroll and flatten colon and view as a 2D sheet

Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)
- 3D visualization (most appropriate)

3D visualization paradigms (use colon as example)

- Section colon into straight pieces, slice in the center, and visualize on a virtual tray
- Unroll and flatten colon and view as a 2D sheet
- Leave as is and perform a virtual fly-through

Visualization Paradigms

Available visualization paradigms

- 2D viewing, slice by slice (non-intuitive)
- 3D visualization (most appropriate)

3D visualization paradigms (use colon as example)

- Section colon into straight pieces, slice in the center, and visualize on a virtual tray
- Unroll and flatten colon and view as a 2D sheet
- Leave as is and perform a virtual fly-through

3D Fly-Through Paradigms

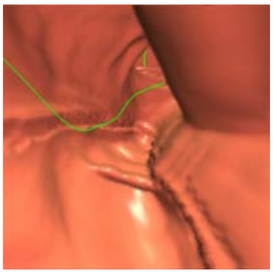
Options range from fully passive to fully interactive

3D Fly-Through Paradigms

Options range from fully passive to fully interactive

Pre-compute a video and just watch

- Cannot stop and explore
- Easy to fall asleep (TV-like)



3D Fly-Through Paradigms

Options range from fully passive to fully interactive

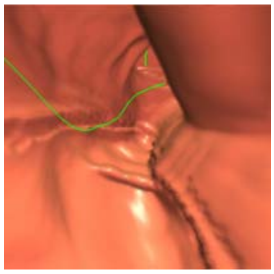
Pre-compute a video and just watch

- Cannot stop and explore
- Easy to fall asleep (TV-like)

Pre-compute a path along which to travel

- Better - more immersive

path



3D Fly-Through Paradigms

Options range from fully passive to fully interactive

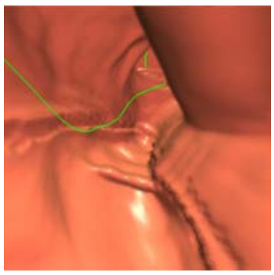
Pre-compute a video and just watch

- Cannot stop and explore
- Easy to fall asleep (TV-like)

Pre-compute a path along which to travel

- Better - more immersive

path



Pre-compute a path, but allow users to “get off”

- Allows users to explore and inspect structures

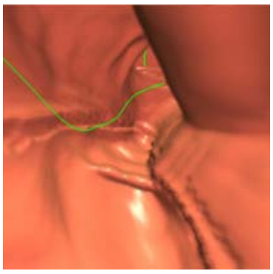
3D Fly-Through Paradigms

Options range from fully passive to fully interactive

Pre-compute a video and just watch

- Cannot stop and explore
- Easy to fall asleep (TV-like)

path



Pre-compute a path along which to travel

- Better - more immersive

Pre-compute a path, but allow users to “get off”

- Allows users to explore and inspect structures

Unguided navigation

- Requires navigation skills (“driver’s license”)

Centerline

A good guiding path is the centerline (medial axis)

[Hong 97]

Centerline

A good guiding path is the centerline (medial axis)

Additional desirable features for navigation:

- Provide a "pull" towards the goal (target)

[Hong 97]

Centerline

A good guiding path is the centerline (medial axis)

Additional desirable features for navigation:

- Provide a "pull" towards the goal (target)
- Provide a "pull" to stay on course, away from walls

[Hong 97]

Centerline

A good guiding path is the centerline (medial axis)

Additional desirable features for navigation:

- Provide a "pull" towards the goal (target)
- Provide a "pull" to stay on course, away from walls

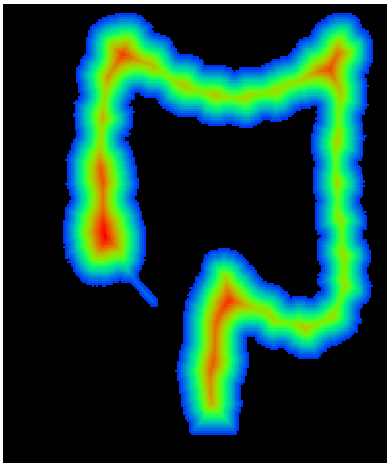
Thus, we need two potential fields

[Hong 97]

Potential Field Computation

Basically a distance transform

Two distance criteria: wall and target



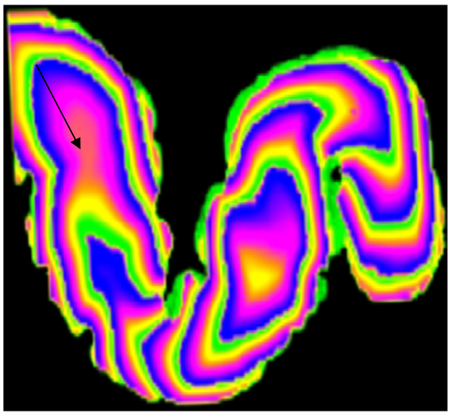
[Hong 97] Distance to boundary Distance to target

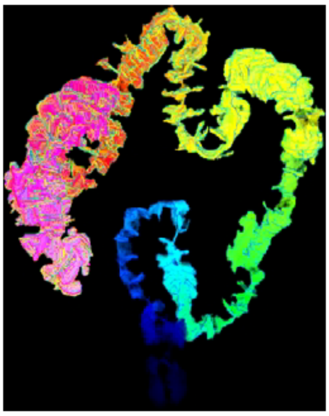
Potential Field Computation

Merge the two potential fields

Each voxel has potential according to both criteria

- Pull user towards target and away from wall





[Hong 97 , Bitter 00] zoomed overall

Rendering: Two Options

Extract polygon mesh from boundary and visualize with graphics hardware

- Discards the original volume data
- Will not allow user to “drill” into the wall to reveal inside-structures

Rendering: Two Options

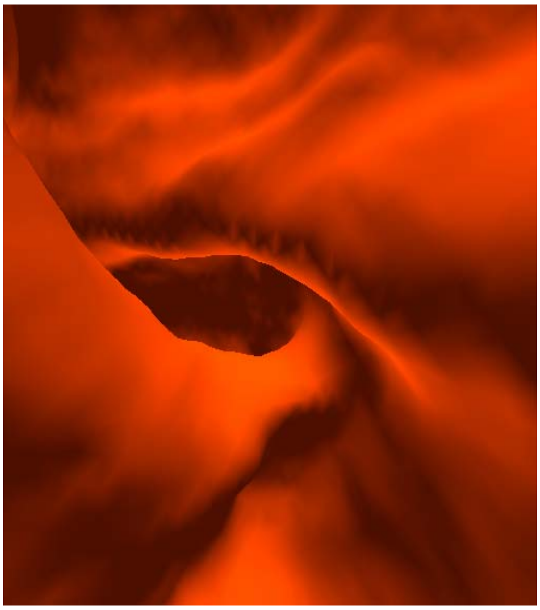
Extract polygon mesh from boundary and visualize with graphics hardware

- Discards the original volume data
- Will not allow user to “drill” into the wall to reveal inside-structures

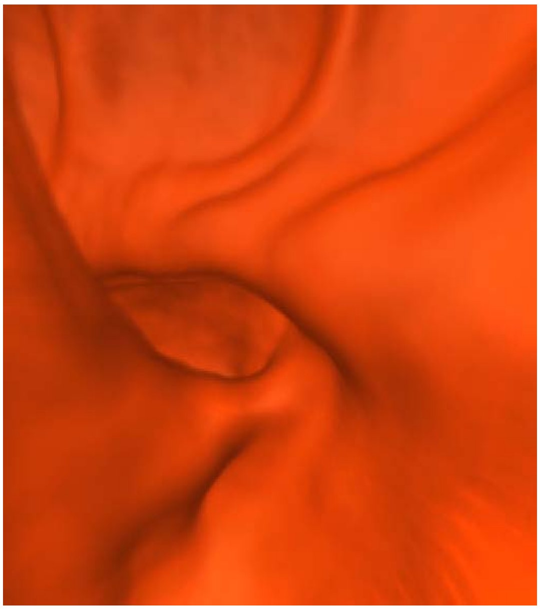
Visualize the volume with direct volume rendering

- Nothing is lost, “drilling” possible
- Will also give more realistic (softer) images since there is no (linear) mesh approximation
- Downside: computationally expensive

Surface vs. Volume Rendering: Human Colon



surface rendered

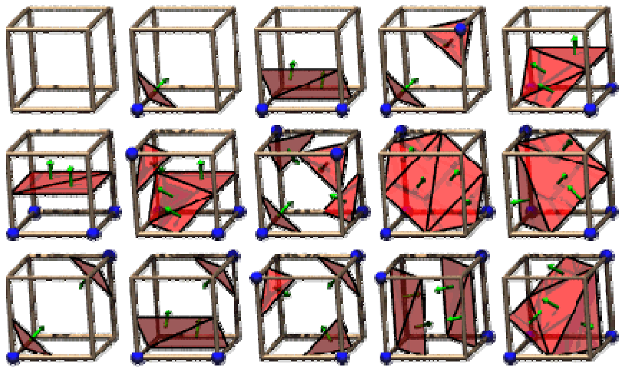


volume rendered

Rendering With Polygons: Mesh Generation

Extraction of polygon mesh with *Marching Cubes*

- Set iso-value *iso* to boundary
- Label all voxels below *iso* as "*out*", else as *"in"*
- Then each voxel 8-cell fits one of 15 base cases:



The 15 Cube Combinations

[Lorensen 87]

Rendering With Polygons: Mesh Generation

Assemble mesh given the extracted polygons

Render with polygon graphics hardware

Rendering With Polygons: Mesh Generation

Assemble mesh given the extracted polygons

Render with polygon graphics hardware

Problem:

- Will likely get very large meshes
- Graphics pipeline will be overwhelmed
- Rendering will not be interactive

Rendering With Polygons: Mesh Generation

Assemble mesh given the extracted polygons

Render with polygon graphics hardware

Problem:

- Will likely get very large meshes
- Graphics pipeline will be overwhelmed
- Rendering will not be interactive

Solution:

- Simplify mesh - undesirable since loss of detail

Rendering With Polygons: Mesh Generation

Assemble mesh given the extracted polygons

Render with polygon graphics hardware

Problem:

- Will likely get very large meshes
- Graphics pipeline will be overwhelmed
- Rendering will not be interactive

Solution:

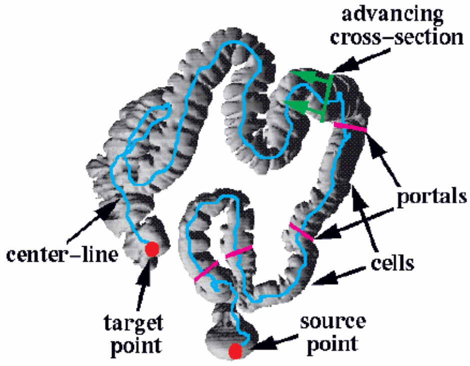
- Simplify mesh - undesirable since loss of detail
- Perform smart occlusion culling during rendering - twisted nature of object helps here

Rendering With Polygons: Occlusion Culling Preparation

Subdivide the colon into cells of about the same number of polygons and/or center path length

At each end of a cell, erect a bounding "portal" polygon perpendicular to the center line

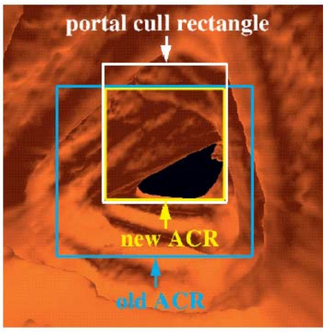
Use the portals to compute visibility during rendering



[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

Locate cell containing camera, render its polygons

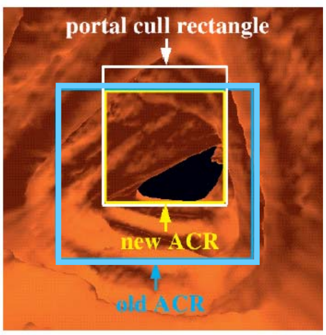


[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen



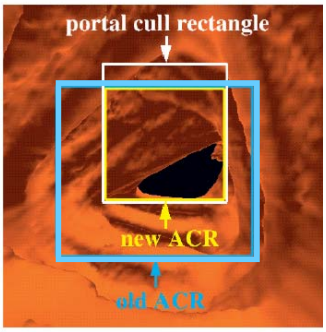
[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen

Render polygons located in the two neighbor cells



[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

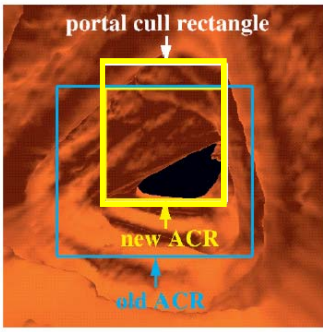
Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen

Render polygons located in the two neighbor cells

Perform two ACR operations for culling: Limit ACR

- by the far portals of these cells



[Hong 97]

Portal limit

Rendering With Polygons: Algorithm With Occlusion Culling

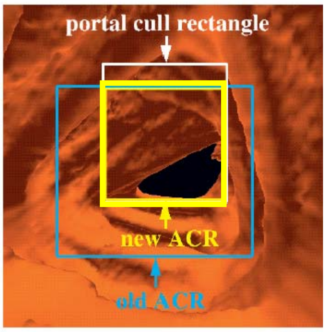
Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen

Render polygons located in the two neighbor cells

Perform two ACR operations for culling: Limit ACR

- by the far portals of these cells



Portal limit

[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

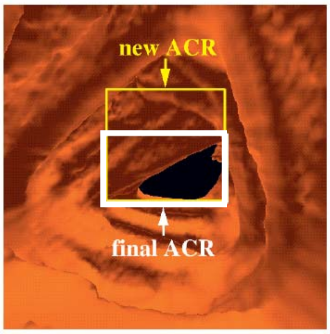
Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen

Render polygons located in the two neighbor cells

Perform two ACR operations for culling: Limit ACR

- by the far portals of these cells
- by z-buffer



Z-buffer limit

[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

Locate cell containing camera, render its polygons

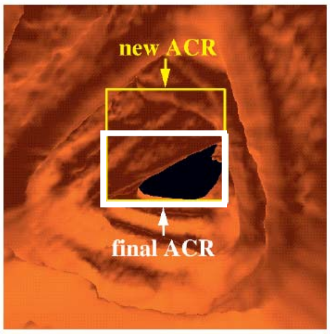
Initialize Aggregate Cull Rectangle (ACR) to screen

Render polygons located in the two neighbor cells

Perform two ACR operations for culling: Limit ACR

- by the far portals of these cells
- by z-buffer

Visit next cells, render polygons, limit ACR



[Hong 97]

Rendering With Polygons: Algorithm With Occlusion Culling

Locate cell containing camera, render its polygons

Initialize Aggregate Cull Rectangle (ACR) to screen

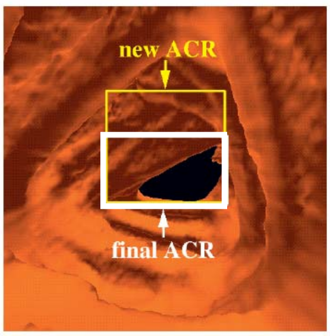
Render polygons located in the two neighbor cells

Perform two ACR operations for culling: Limit ACR

- by the far portals of these cells
- by z-buffer

Visit next cells, render polygons, limit ACR

Stop when ACR degenerates to zero



[Hong 97]

Direct Volume Rendering: Raycasting Fundamentals

Shortcoming:

- Traversing and interpolating empty space until a boundary is hit is time consuming and limits performance

Direct Volume Rendering: Raycasting Fundamentals

Shortcoming:

- Traversing and interpolating empty space until a boundary is hit is time consuming and limits performance

Require some form of accelerated empty-space traversal - “*space leaping*”

Direct Volume Rendering: Raycasting Fundamentals

Shortcoming:

- Traversing and interpolating empty space until a boundary is hit is time consuming and limits performance

Require some form of accelerated empty-space traversal - "*space leaping*"

Solutions

- Render polygons into z-buffer, then use the z-depths to start rays (large polygon overhead)

Direct Volume Rendering: Raycasting Fundamentals

Shortcoming:

- Traversing and interpolating empty space until a boundary is hit is time consuming and limits performance

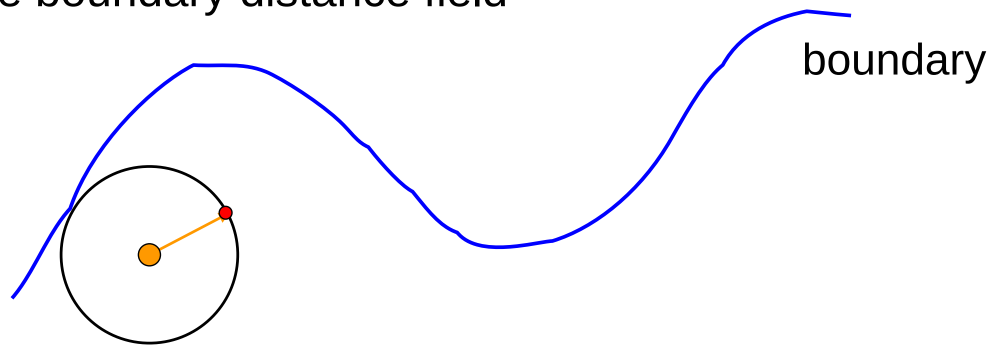
Require some form of accelerated empty-space traversal - "*space leaping*"

Solutions

- Render polygons into z-buffer, then use the z-depths to start rays (large polygon overhead)
- Better: use the potential field to speed up rays

Potential Field Assisted Raycasting

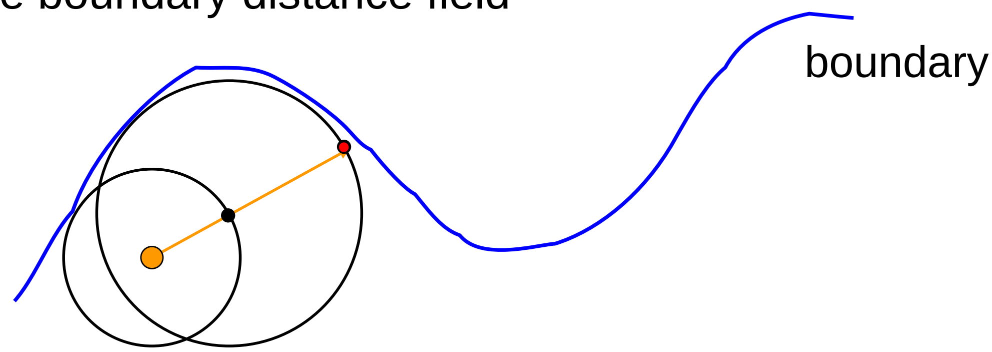
Use the boundary distance field



[Wan 99]

Potential Field Assisted Raycasting

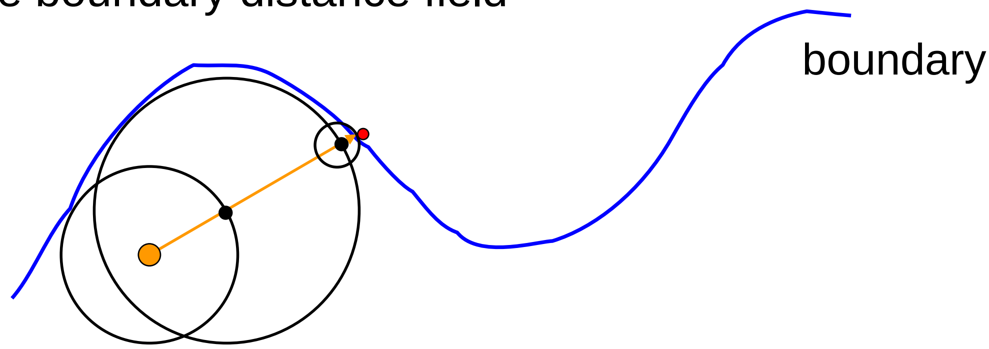
Use the boundary distance field



[Wan 99]

Potential Field Assisted Raycasting

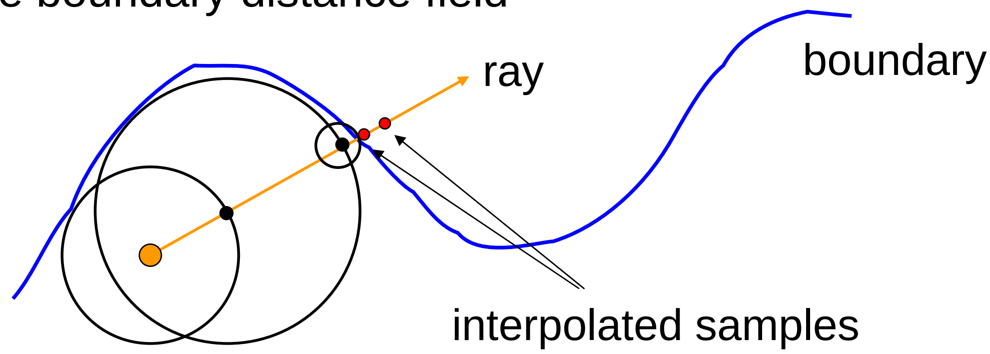
Use the boundary distance field



[Wan 99]

Potential Field Assisted Raycasting

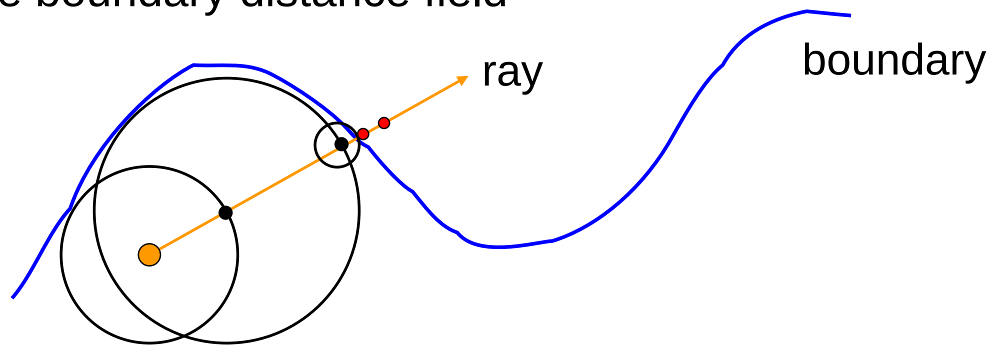
Use the boundary distance field



[Wan 99]

Potential Field Assisted Raycasting

Use the boundary distance field



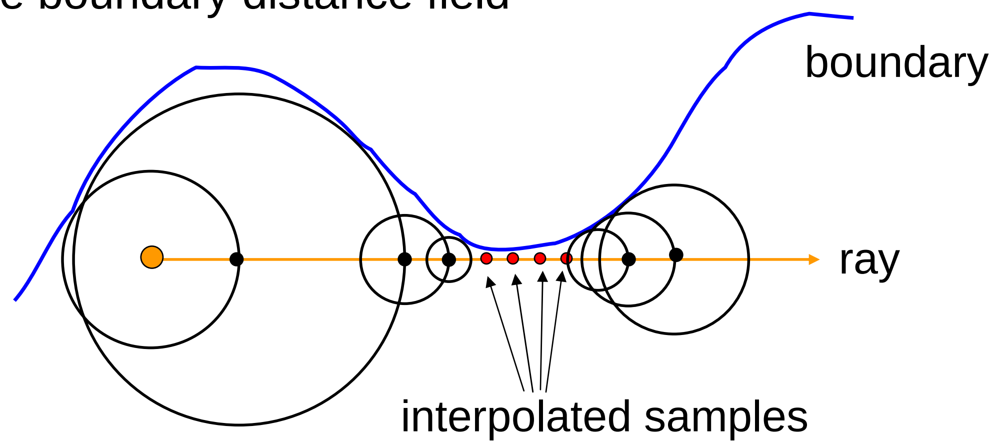
Two hops are required instead of ten samples

This, in fact, corresponds to the average case

These speedups (and further optimizations) allow the required interactive frame rate of 10 fps

Potential Field Assisted Raycasting

Use the boundary distance field

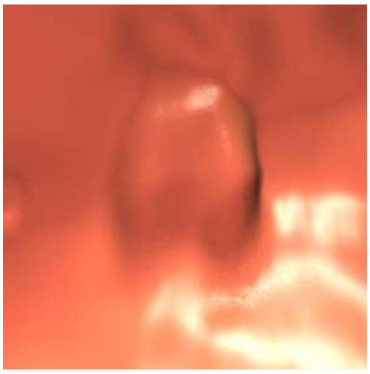


Rays that graze the surface will not accelerate well

- Fortunately these are rare

Opaque vs. Translucent Rendering

Setting a high opacity at the boundary will render the surface opaque

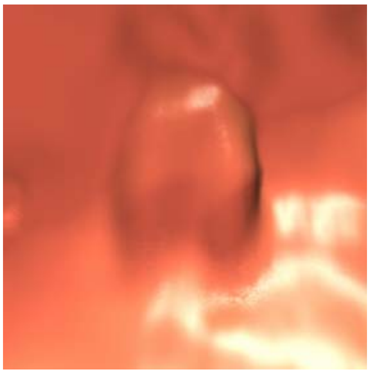
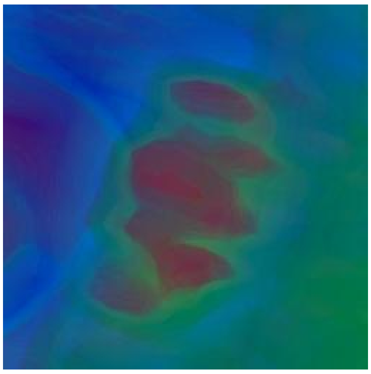
opaque 

Opaque vs. Translucent Rendering

Setting a high opacity at the boundary will render the surface opaque

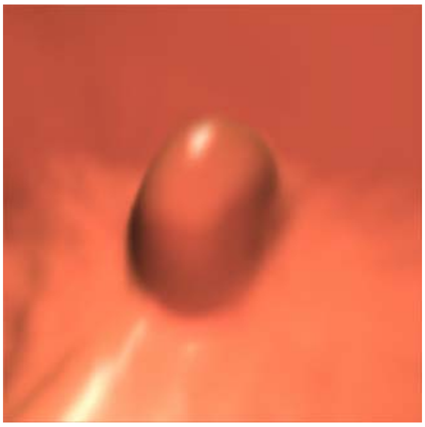
Selecting low opacities allows rays to penetrate further into the boundary tissue

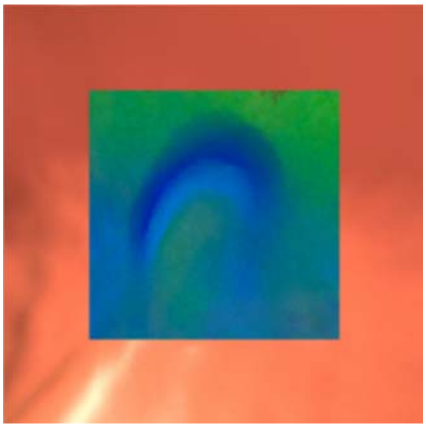
- Will render the boundary tissue translucent

opaque   translucent

Digital Biopsy

Translucent rendering allows one to visualize tissue underneath the surface [Kaufman 05]

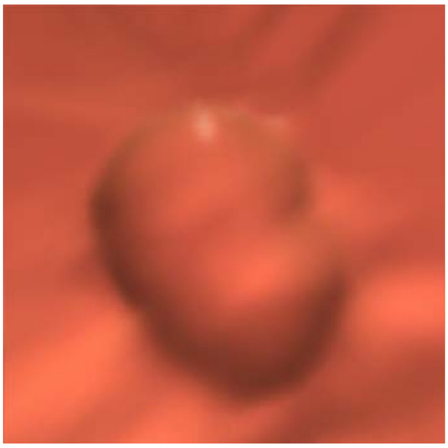


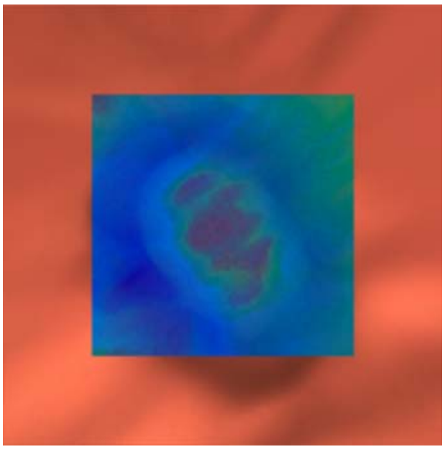


Hyperplastic polyp: surface translucent

Digital Biopsy

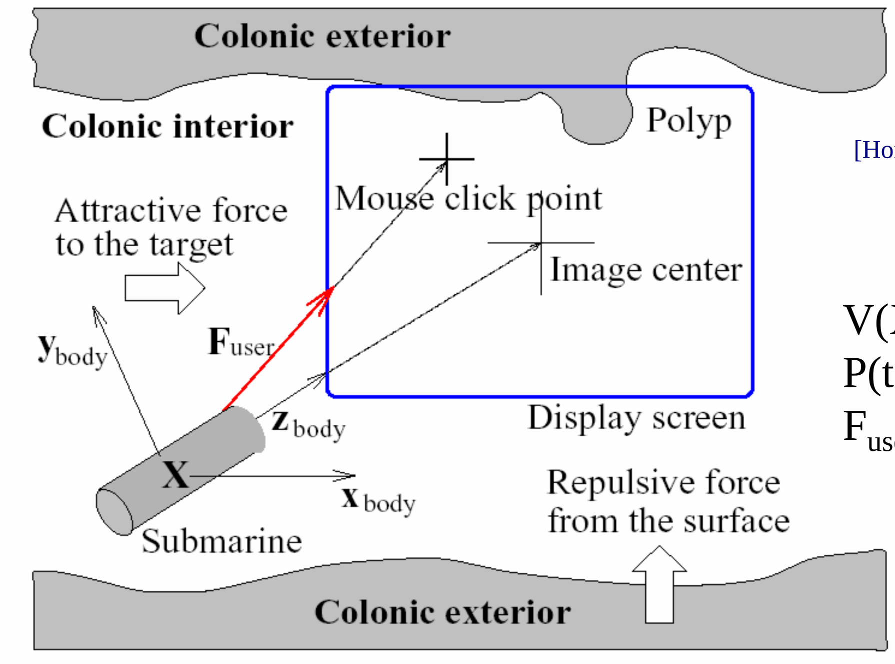
Translucent rendering allows one to visualize tissue underneath the surface [Kaufman 05]





Adenoma: surface translucent

Physically-Based Navigation Control



[Hong 97]

V(X): potential field
P(t): linear momentum
F_{user}: user force

Submarine dynamics: $\dot{P}(t) = -\nabla V(X) - kP(t) + F_{user}(t)$

Physically-Based Navigation Control

The influence of the target D_t and distance D_s potential fields can be adjusted by constants C_t and C_s

$$V(X) = \begin{cases} C_t D_t(X) + C_s(\rho / D_s(X) - 1)^2, & 0 < D_s < \rho \\ C_t D_t(X), & otherwise \end{cases}$$

The user has full control over C_t and C_s

Large C_t accelerates the submarine towards target

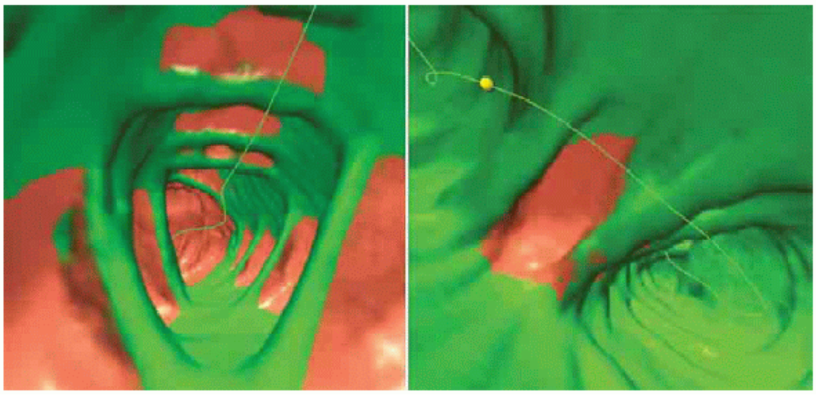
Small C_s allows users to come close to the boundary

F_{user} allows users to control the camera orientation

Exploration Feedback

One way to remind users that they have inspected a certain area is to tag that area in a specific color

- green: seen before
- red: not yet examined

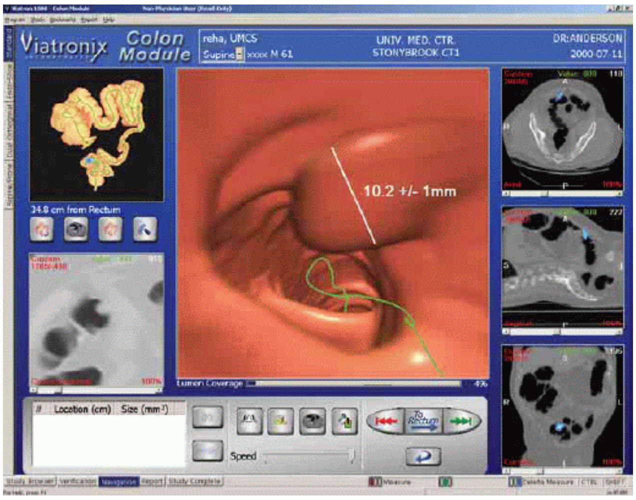


[Kaufman 05]

Further Navigation Aids

Peripheral to main window provided are:

- 2D slicer viewers of raw data
- bird's eye view
- clickable map of unexplored regions
- notebook



References (1)

[Bitter 00] I. Bitter, M. Sato, M. Bender, K. McDonnell, A. Kaufman, and M. Wan, "CEASAR: a smooth, accurate and robust centerline extraction algorithm," IEEE Visualization '00, pp. 45-52, 2000.

[Cohen 94] D. Cohen and Z. Sheffer, "Proximity clouds - an acceleration technique for 3D grid traversal," The Visual Computer, vol. 10, no. 11, pp. 27-38, 1994.

[Danskin 92] J. Danskin and P. Hanrahan, "Fast algorithms for volume ray tracing," Symp. Volume Visualization '92, pp. 91-98, 1992.

[Engel 01] K. Engel, M. Kraus, and T. Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading," Siggraph/Eurographics Workshop on Graphics Hardware 2001.

[Grimm 04] S. Grimm, S. Bruckner, A. Kanitsar, E. Gröller, "Memory Efficient Acceleration Structures and Techniques for CPU-based Volume Raycasting of Large Data," Symposium on Volume Visualization and Graphics, pp. 1-8, 2004.

[Hauser 01] H. Hauser, L. Mroz, G. Bischi, and E. Gröller, "Two-Level Volume Rendering," IEEE Trans. on Visualization Computer Grapics, vol. 7, no. 3, pp. 242-252, 2001.

[Hong 97] L. Hong, S. Muraki, A. E. Kaufman, D. Bartz, T. He, "Virtual voyage: interactive navigation in the human colon, SIGGRAPH '97, pp. 27-34, 1997.

References (2)

[Kaufman 05] A. Kaufman, S. Lakare, K. Kreeger, and I. Bitter, "Virtual colonoscopy," Communications of the ACM, vo. 48, no. 2, pp. pp. 37-41, 2005.

[Kaufman 05] A. Kaufman and K. Mueller, "Overview of volume rendering", The Visualization Handbook, C. Johnson and C. Hansen, editors, Academic Press, 2005

[Kindlmann 98] G. Kindlmann and J. Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," Symp. Volume Visualization '98, pp. 79-86, 1998

[Kniss 02] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional transfer functions for interactive volume rendering," IEEE Trans. Visualization and Computer Graphics, vol. 8, no. 3, pp. 270-285, 2002.

[Lakare 00] S. Lakare, M. Wan, M. Sato, and A. Kaufman, "3D digital cleansing using segmentation rays," IEEE Visualization '00, pp. 37-44, 2000.

[Lakare 03] S. Lakare, A. Kaufman, "Anti-Aliased Volume Extraction," Eurographics - IEEE TCVG Symposium on Visualization, 2003.

[Leung 06] W. Leung, N. Neophytou, and K. Mueller, "SIMD-aware raycasting," Volume Graphics 2006, pp. 59-62, August 2006.

[Li 03] W. Li, K. Mueller, and A. Kaufman, "Empty-space skipping and occlusion clipping for texture-based volume rendering," IEEE Visualization '03, pp. 317-325, October 2003.

References (3)

[Lorensen 87] W. Lorensen and H. Cline, "Marching Cubes: A high resolution 3D surface construction algorithm," Siggraph '87, pp. 163-169, 1987.

Mueller 99] K. Mueller, T. Möller, and R. Crawfis, "Splatting without the blur," Proceedings Visualization '99, pp. 363-371, 1999.

[Mueller 05] K. Mueller, S. Lakare, and A. Kaufman, "Volume exploration made easy using feature maps," Scientific Visualization: Extracting Information and Knowledge from Scientific Data Sets, G. Nielson, G.-P. Bonneau and T. Ertl, eds, Springer-Verlag, Heidelberg, Germany.

[Pfister 99] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The VolumePro real-time ray-casting system," SIGGRAPH 1999, pp. 251-260, 1999.

[Rezk-Salama 00] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl, "Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage-Rasterization," Eurographics/SIGGRAPH Workshop on Graphics Hardware '00, pp. 109-118, 2000.

[Srámek 00] M. Srámek and A. E. Kaufman, "Fast ray-tracing of rectilinear volume data using distance transforms," IEEE Trans. on Visualization and Computer Graphics, vol. 6, no. 3, pp. 236-252, 2000.

[Stegmaier 05] S. Stegmaier, M. Strengert, T. Klein, T. Ertl, "A Simple and Flexible Volume Rendering Framework for Graphics-Hardware-based Raycasting," Volume Graphics 2005, pp. 187-195, August 2005.

References (4)

[Wan 99] M. Wan, A. Kaufman, S. Bryson, "High Performance Presence -Accelerated Ray Casting", IEEE Visualization `99, pp. 379-386, 1999

Acknowledgments

Thanks to Prof. Arie Kaufman and collaborators for various images used in this presentation

Funding provided by: NIH, NSF, ONR