Clipping

Clipping to See Inside

- Obscuring critical information contained in a volume data
  - Contour displays show only exterior visible surfaces
  - Isosurfaces can hide other isosurfaces
  - Other displays can become crowded and complicated

- Clipping: remove a part of a volume to observe the rest of its contents
  - cut away part of a volume to see what is behind it
  - intersection between a slicing plane and a volume dataset
  - also intersection of any surface with a volume dataset

- Different approaches:
  - Planar clipping
  - Interactive clipping
  - Volume clipping

Planar Clipping

- A 3D model is cut by a clip or slicing plane

- It is performed by evaluating each volume element against the boundary of the half-space
  - If inside the half-space, display the element
  - If outside the half-space, discard the element
  - If partially contained in the half-space, check and display the face of the element

- If a face is partially contained in the half-space, test for subpolygons

Clipping with Caping

- Clip the model with a display of the scalar result on the clip surface

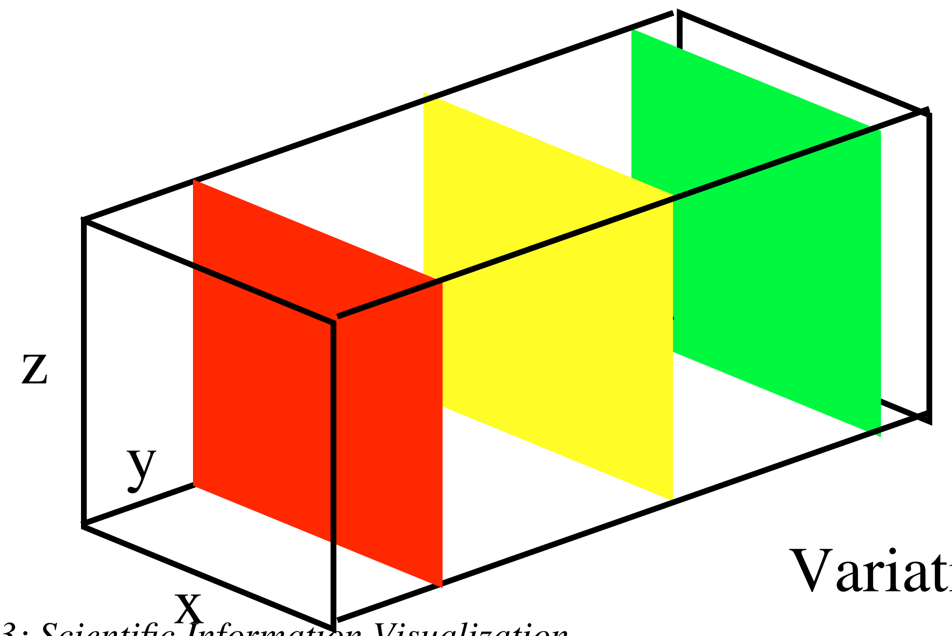
- The caped surfaces are produced by generating additional display polygons for each volume element in the cutting region

- Clipping alone can be performed in graphics hardware
  - Polygon clipping on a polygon-by-polygon basis in hardware

- Capping must be performed in the software
  - Requires knowledge of the entire volume element to interpolate vertex values to form the capping polygon and its scalar values

Sampling Planes

- Combining two or more volume slicing operations with a wire-frame outline of the model
- Allows a simultaneous display of scalar results at multiple locations within the volume



Variation along y-axis

Interactive Clipping

- A planar clipping used in an interactive manner
  - Once the clip plane is defined, it's position and orientation can be changed in real time in order to cut away the given volume at any orientation and any position.

- Clip plane equation = $Ax + By + Cz + D = 0$
  - All points with eye coordinates (x_e, y_e, z_e, w_e) that satisfy $(A\ B\ C\ D)M^{-1}(x_e, y_e, z_e, w_e)^T >= 0$ lie in the half-space defined by the plane, where ***M*** is the current model-view matrix. All points not in this half-space are clipped away.

- Initial plane is parallel to the *z*-axis:

  $A = y_2 - y_1;\ B = -(x_2 - x_1);\ C = 0;\ D = 0$

- Coefficients A, B and C control rotation of the clip plane in a 3D space whereas the coefficient *D* controls translation.

Best-View Mode

- Dynamic manipulation of the clip plane
  - Small effective visible area of the clipped surface
  - Done in low-resolution mode for interactivity

- Bringing to the best-view mode
  - Clipped surface at its maximum exposure
  - Rendering at high resolution

- Automated option for best-view mode
  - Tracking all transformations the clip plane has gone through
  - Simple case of initial *xy* plane setting:

    First rotation about z-axis

    Second rotation about the vector which is defined by the intersection of the clip and *xy* planes.

Rendering

- Texture-based volume rendering
  - A stack of 2D textures (generated form the input data or images)

- Multi-resolution rendering
  - High-resolution (HR) mode

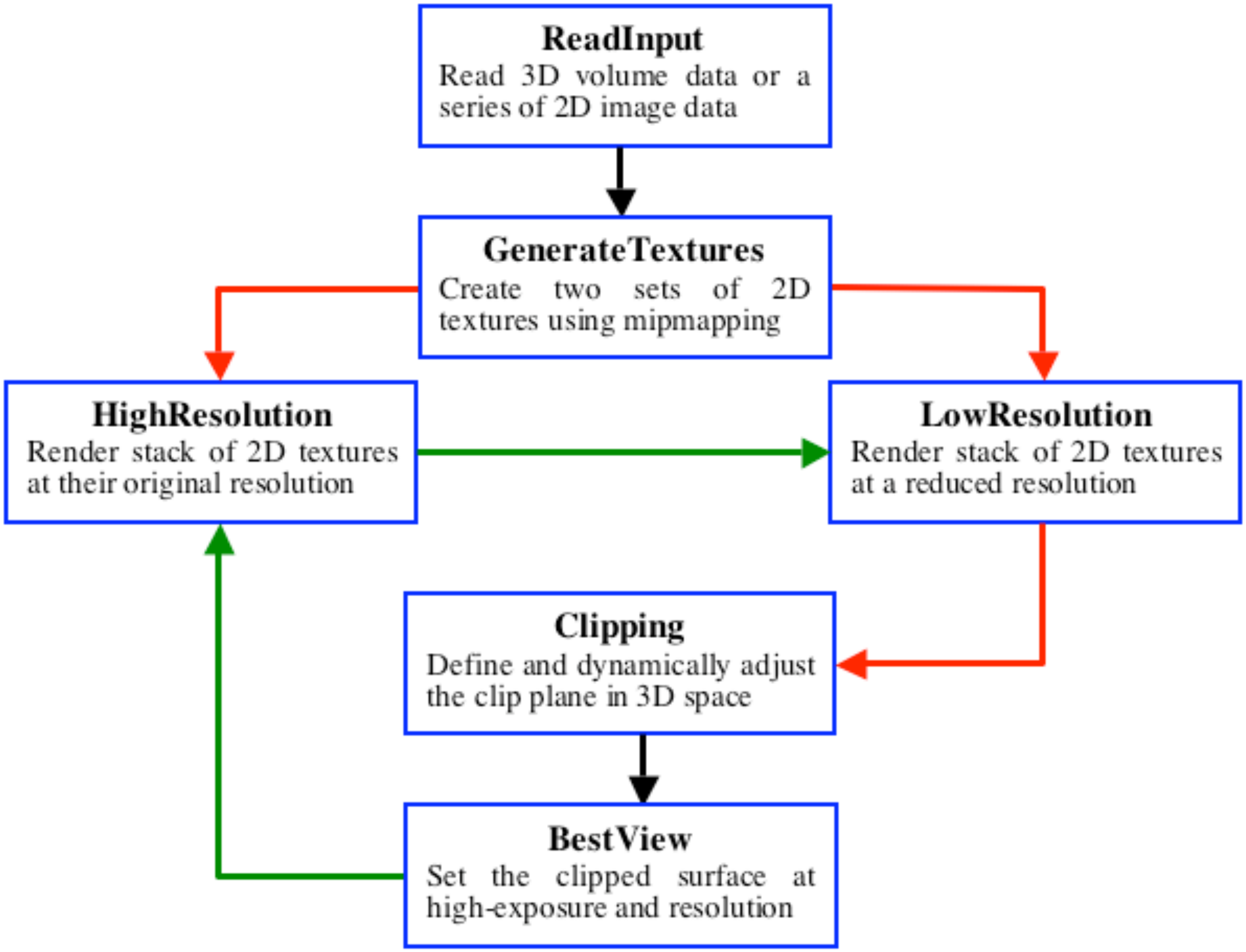
    The original fine grid of the input data or the original resolution of the input images
  - Low-resolution (LR) mode

    The re-sample data at a lower resolution than the original resolution

- Rendering speed
  - LR gives a much better frame rate but HR retains complete information contained in the data.

- Interactivity
  - Navigate through the volume data in LR and switch to HR for quality

Flow Diagram



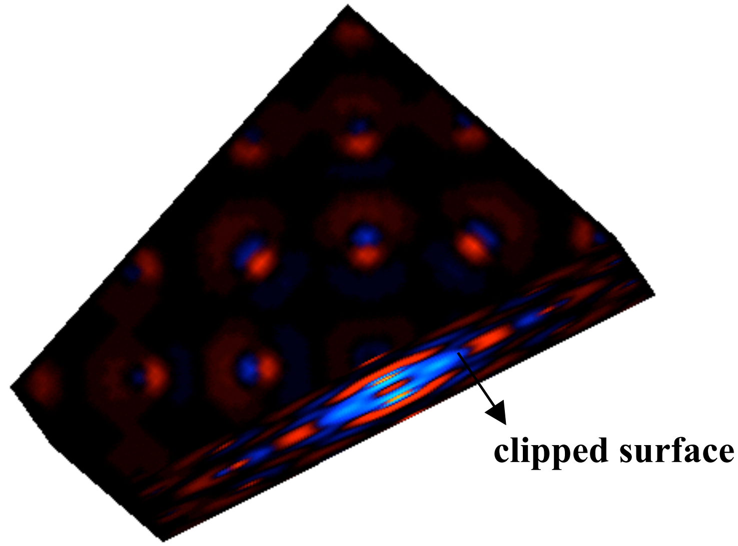
Charge Density

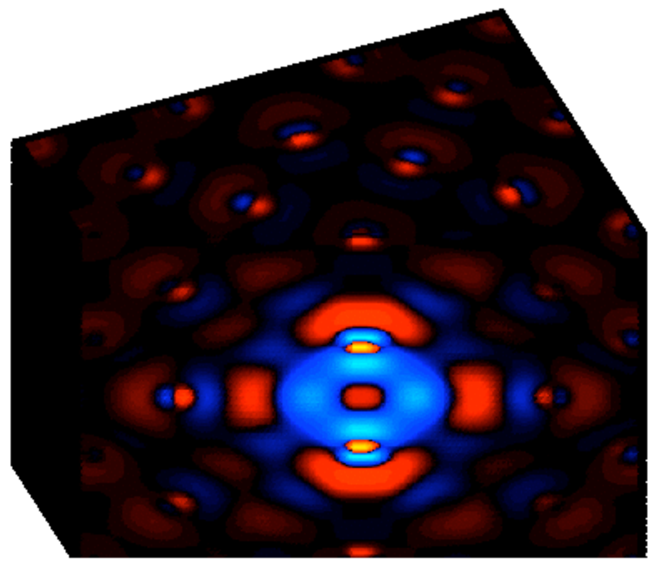
- Quantum mechanical simulations
  - MgO system
  - 64 atoms and 864 electrons
  - Data defined on 512^3 regular grid

- HR rendering (bottom)
  - 512 textures each with 512x512 pixel;
  - 4 frames per seconds

- LR rendering (top)
  - 128 textures at 128x128 resolution
  - 35 frames per second

Electrons are depleted from blue regions whereas electrons are deposited in red regions due to a vacancy defect located at the center of the system.

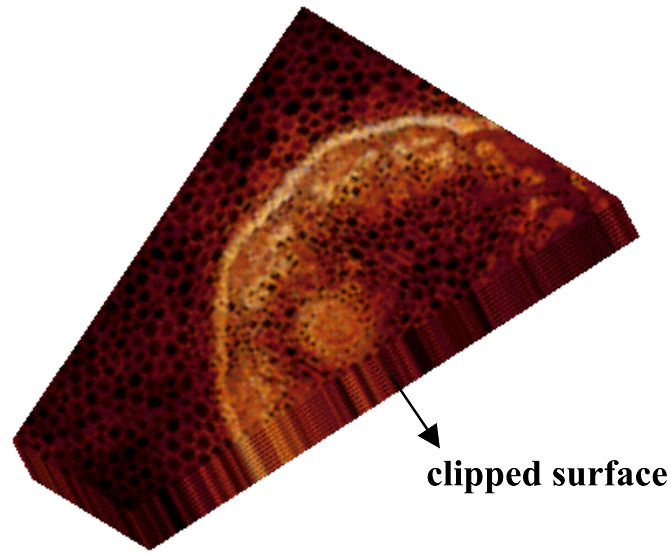


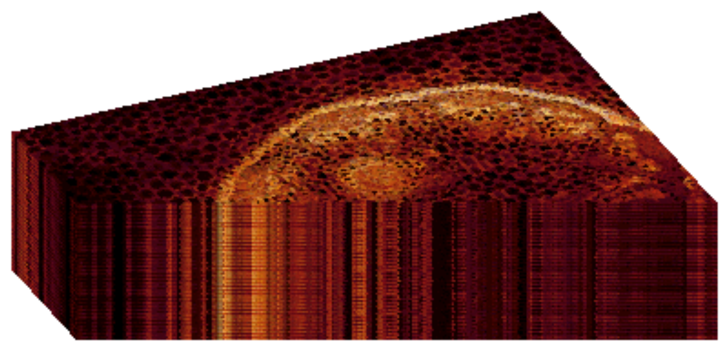


Confocal Data

- Confocal data:
  - Tissues of a plant stem
  - A series of tiff images
- HR rendering (bottom)
  - 200 textures each with 512x512 pixel;
  - 0.6 frames per seconds
- LR rendering (top)
  - 50 textures at 128x128 resolution
  - 28 frames per second

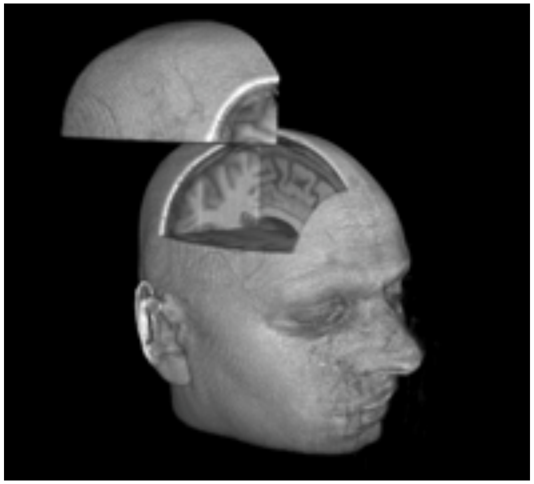
Twenty-five images (each with 512 x 512 pixels) were replicated to generate 200 images used in HR.





Volume Clipping

- Using complex geometries for volume clipping
  - Cutting a cube-shaped opening into a volume
  - Segmentation information used for defining curved clip geometries
  - Isosurface as clip object
  - Surface of a body in fluid dynamics
- Clipping tailored to texture-based rendering on graphics hardware
  - Use per-fragment operations
  - Give interactivity (high frame rates)
- Two clipping techniques
  - Depth-based clipping
  - Voxelized clip object

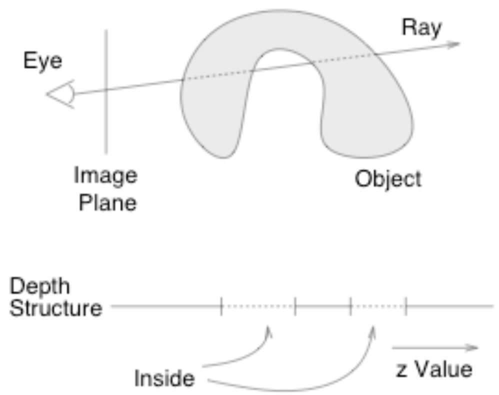


Depth-Based Clipping

- Depth structure of a clip object
  - Clip geometry represented as a tessellated boundary surface
    in the form of triangular meshes
- Graphics hardware allows manipulation of depth values
  - To clip away unwanted parts of the volume
- Produces high-quality images
  - Uses 2D textures with texels having a one-to-one correspondence to pixels
  - Clipping is performed with per-pixel accuracy

Building Depth Structure

- Define depth structure (1 D representation) of a clip object for a single pixel in the frame buffer
  - ➢ Consider a single ray from eye point to the scene
    - Apply operations on those fragments that correspond to the respective position of the pixel in the frame buffer



- Build the depth structure per pixel basis
  - ➢ Stores the depth values for each boundary between object and background space
    - Classify intervals as inside or as outside of the clip object and specify them as visible or invisible
  - ➢ Rendering of each fragment of a volume slice has to check to which class of interval the fragment belongs
    - Based on visibility property, the fragment is blended into the frame buffer or clipped away.

Implementation Issues

- Depth structure stored on a per-pixel basis
  - Exploits the depth buffer to store interval boundaries
  - Only a single boundary can be implemented because only one depth value can be stored per pixel
  - Depth test to decide the visibility property of a fragment

    less - clip away the volume behind the geometry

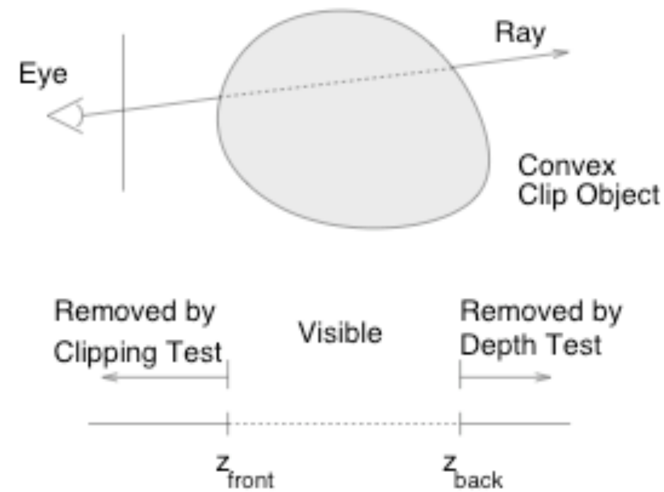
    greater - clip away the volume in front of the clip geometry

- Implementation with OpenGL

- How to handle multiple depth values corresponding to a pixel

Volume Probing

- Depth structure with two boundaries (for convex clip geometry)
  - ➢ Depth tests, depth clipping, and depth computations in the fragment-operations unit
  - ➢ Volume probing: Leaves visible only the volume inside the clip object



- Basic algorithm
  - ➢ Determine z_{front} by rendering the front faces of the clip geometry into the depth buffer
  - ➢ The contents of the depth buffer are stored in a texture and are later used to shift the depth values of all fragments in the following rendering passes by $-z_{front}$
  - ➢ The depth buffer is cleared and the backside of the clip geometry is rendered into the depth buffer (with depth shift enabled) to build the secondary boundary.
  - ➢ Slices through the volume data set are rendered and blended into the frame buffer.

    Depth shift and depth testing are enabled, but the depth buffer is not modified.

Probe Depth Function

- How the depth of a fragment determines its visibility

- Define a boolean function to determine whether a fragment ***f*** (with depth value z_f) passes depth clipping and depth testing:

$$d_f(z_f) = d_{\mathrm{clip}}(z_f) \wedge (z_f \ \mathrm{op}\ z_b)$$

Where $d_{\mathrm{clip}}(z_f) = (z_f \geq 0) \wedge (z_f \leq 1)$ defines depth clipping against the bounds 0 and 1 of the view frustum. z_b is the current entry in the depth buffer.

- With shift of $-z_{\mathrm{front}}$ applied to all subsequent depth values

$$d_f(z_f) = d_{\mathrm{probe}}(z_f) \wedge (z_f \leq 1 + z_{\mathrm{front}})$$

where $d_{\mathrm{probe}}(z_f) = (z_f \geq z_{\mathrm{front}}) \wedge (z_f \leq z_{\mathrm{back}})$ represents the logical operation for displaying the volume only in the interval $[z_{\mathrm{front}}, z_{\mathrm{back}}]$

Shader Program

- Render frontfaces of the clip geometry to get z_{front} depth values
- Transform the contents of the depth buffer to main memory as 32 bit unsigned integers per depth value z_{front}
- Define HILO texture object (a 2D texture consisting of two 16 bit unsigned integers per pixel)
- Enable a texture shader program to replace z value of a fragment by z-z_{front}
  - **DotProductDepthReplace** fragment operation
    - Perform a texture look up in HILO texture
    - Compute dot products, Z and W, between respective texture coordinates and previously fetched values from HILO
    - Replace the current fragment's depth by Z/W

Volume Cutting

- Only the volume outside the clip object remains visible

- Invert the role of visibility property
  - Logical operation for volume cutting:

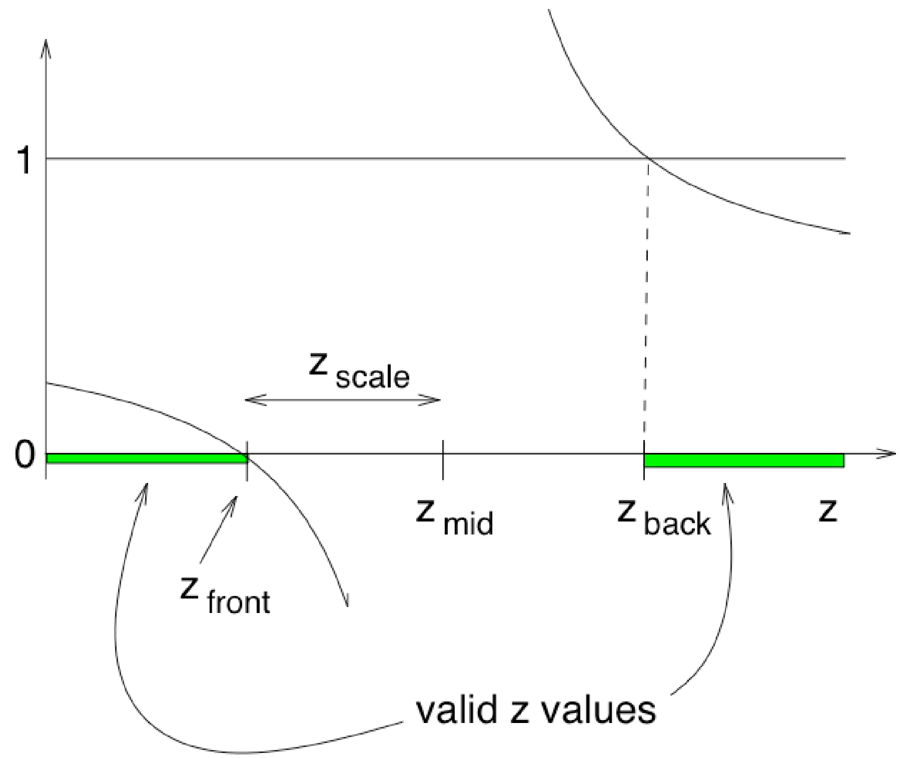
    $d_{cutting}(z_f) = (z_f \leq z_{front}) \vee (z_f \geq z_{back})$

  - Inverse depth function:

$$g(z) = \frac{1}{2}\frac{z_{scale}}{z - z_{mid}} + \frac{1}{2}$$

$$z_{scale} = \frac{z_{back} - z_{front}}{2}$$

$$z_{mid} = \frac{z_{back} + z_{front}}{2}$$



Clipping Based on Volumetric Textures

- Clip object is voxelized and represented by an additional volume data
  - Uses a second volumetric texture to specify clipping voxels of the real volume
    - any arbitrary objects (convex or concave)
  - Store voxelized clip geometry as a binary volume
    - Voxel inside (**1**) or outside (**0**) the clip geometry

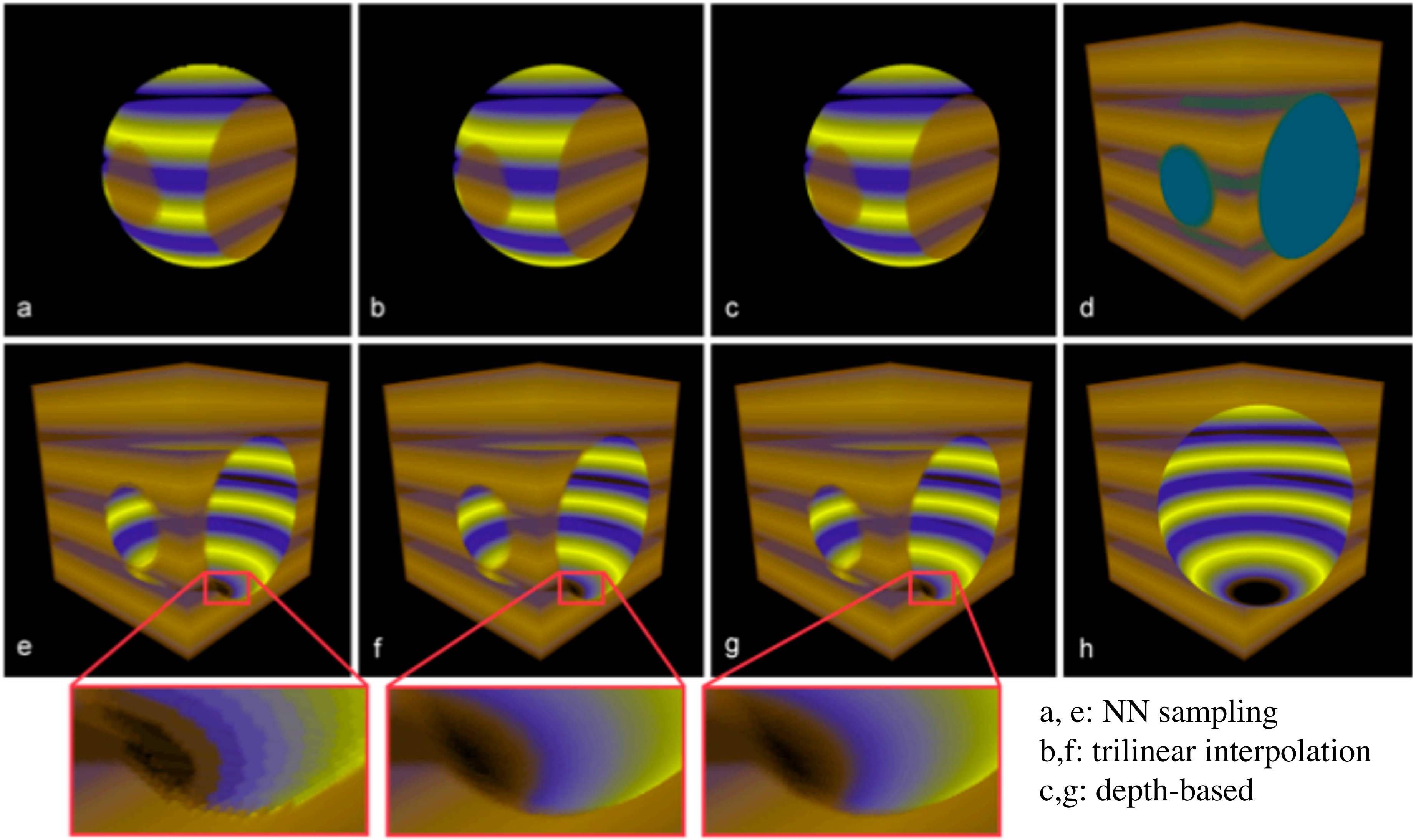
- Rendering maps a texture slice of the data set and a slice of the clip texture onto the same slice polygon
  - Two textures are combined using a per-component multiplication
    - All voxels to be clipped are multiplied by zero and completely discarded

- The clip object can be rotated or translated by applying any affine transformation to texture coordinates.

- A change in volume probing and volume cutting by applying a per-fragment invert mapping to the clipping textures.

Clipped Volume Images (size = 128^3)

Frame rates: 24.1 (no clipping), 15.8 (voxelized clip) and 8.2 (depth-based clip) foe window display size of 512^2.



a, e: NN sampling
b,f: trilinear interpolation
c,g: depth-based

References

- G. Khanduja and B.B. Karki, Visualization of 3D scientific datasets based on interactive clipping, *The 13th Int'l. Conf. on Central Europe in Computer Graphics, Visualization and Computer Vision (WSCG'05),* ISBN 80-903100-9-5, pp. 34 – 37, 2005.

- M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and T.Erl. Level-of-detail volume rendering via 3D textures. *In Proceedings of IEEE Volume Visualization 2000*, pp. 7-13, 2000.

- D. Weiskopf, K. Engel and T. Ertl. Volume clipping via per-fragment operations in texture-based volume visualization, *IEEE Visualization 2002 Proceedings,* pp. 93-100, ACM Press, 2002

- D. Weiskopf, K. Engel, and T. Ertl Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Trans. on Visualization and Computer Graphics*, 9, 298-312 2003.