Texture-Based Visualization

Texture Mapping Techniques

- Exploiting graphics hardware to perform volume rendering
  - Graphics Processing Unit (GPU): increasing processing power and flexibility

    Cabral et al., "Accelerated volume rendering and tomographics reconstruction using texture mapping hardware", *Symposium on Volume Visualization*, 91, 1994

- Two major approaches:
  - 2D texture mapping

    Uses stacks of textured slices to represent the volume

  - 3D texture mapping

    Use a single three-dimensional texture to represent the volume.

Basic Steps

- Upload the whole volume to the graphics hardware as textures
  - A stack of 2D textures
  - A single 3D texture

- Make the textured slices are either object-aligned or view-aligned
  - Store three stacks of slices, one stack for each major viewing axis, and choose one most parallel to the current viewing direction
  - Map 3D texture onto polygons parallel to the viewing plane

- Render a number of partially transparent slices in front-to-back or back-to-front order using alpha bending
  - More efficient than raycasting

    Entire 2D slice of the voxels are "cast" at one time rather than each image pixel built up ray by ray.

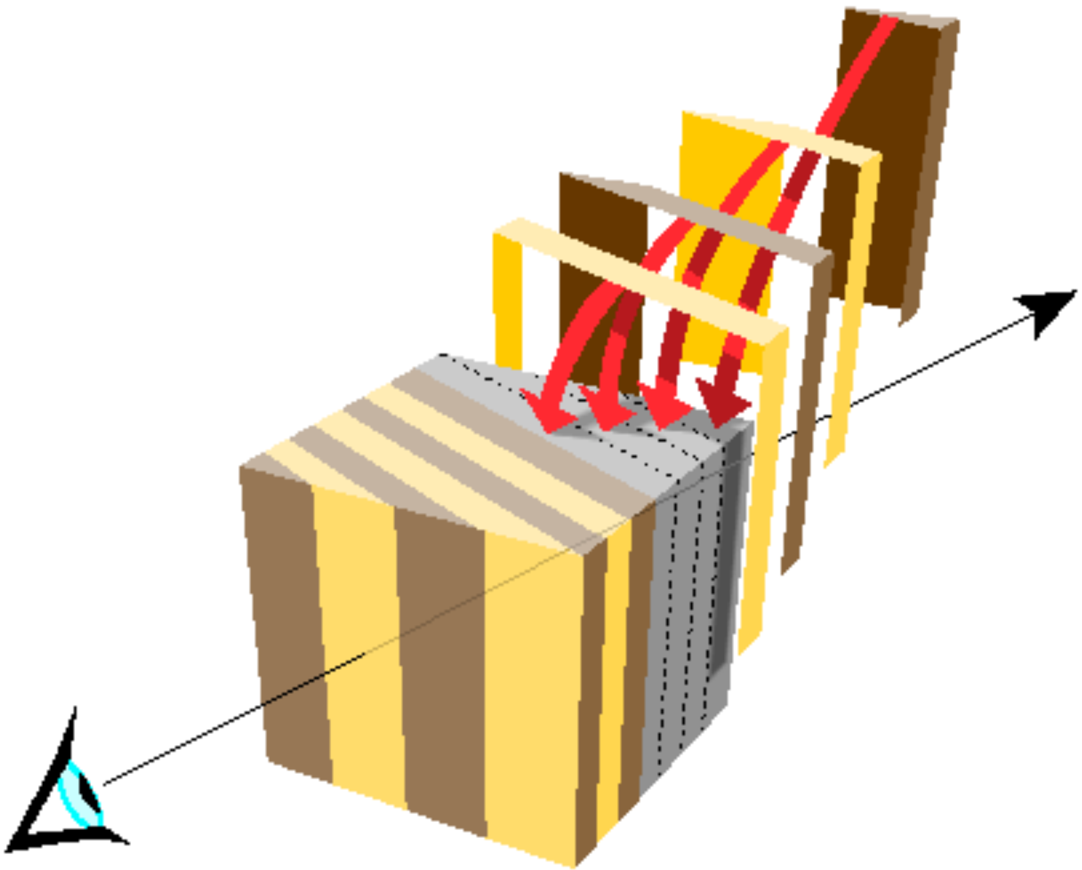
Object-Aligned Slices

- Used in combination with 2D textures
- Every slice represents one volume voxel slice
- Image quality is best when the slices are parallel to the view plane
  - Store 3 stacks of textures representing the main view directions and display only the slice stack that is most parallel to the current viewing direction
  - Hardware does bilinear interpolation in a 2D texture resulting in fast rendering
  - Supersampling is not possible:

    opacity changes with rotation
- Multitexture slices
  - Produce intermediate slices by blending together the 2 neighboring textures with position inside the 2 slices as blending factor
  - Make supersampling possible.

View-Aligned Slices

- Used with one 3D texture representing the volume
- Slices can be stacked along an arbitrary direction
  - ➢ The slices stay parallel to the view plane while the volume texture is rotated
- Image quality is independent of the viewing direction
  - ➢ Trilinear interpolation can be used
  - ➢ Opacity stays constant
  - ➢ Supersampling is possible.



Blending: Over and Under

- Over
  - Most common way: back-to-front rendering
  - Approximates the flow of light through a colored, translucent material
  - Texels with higher alpha values tend to obscure texels behind them

  glblendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)

- Under
  - Volume slices are rendered front-to-back order
  - Gives the same result as the over operator blending slices from the back-to-front order

Blending: Attenuate and MIP

- Attenuate
  - Simulates an X-ray of material
  - Texel's alpha attenuates light shinning through the material along the view direction towards the viewer
  - Final brightness at each pixel is attenuated by the total texel density along the direction of view

    glBlendFunc(GL_CONSTANT_ALPHA_EXT, GL_ONE)

    glBlendColorEXT(1.0, 1.0, 1.0, 1.0/number_of_slices)

- MIP
  - MIP: Maximum Intensity Projection
  - Finds the brightest texel alpha from all the texture slices at each pixel location
  - Acts like a contrast enhancing operator

    glBlendFunc(GL_ONE, GL_ONE)

    glBlendEquationEXT(GL_MAX_EXT)

Sampling Frequency

- A number of factors to consider in deciding the number of slices (data polygons) for rendering the volume

- 2D versus 3D textures
  - 2D textures are constrained in a plane while 3D textures can sample data along any direction

- Performance
  - Interactive and details modes for viewing the volume

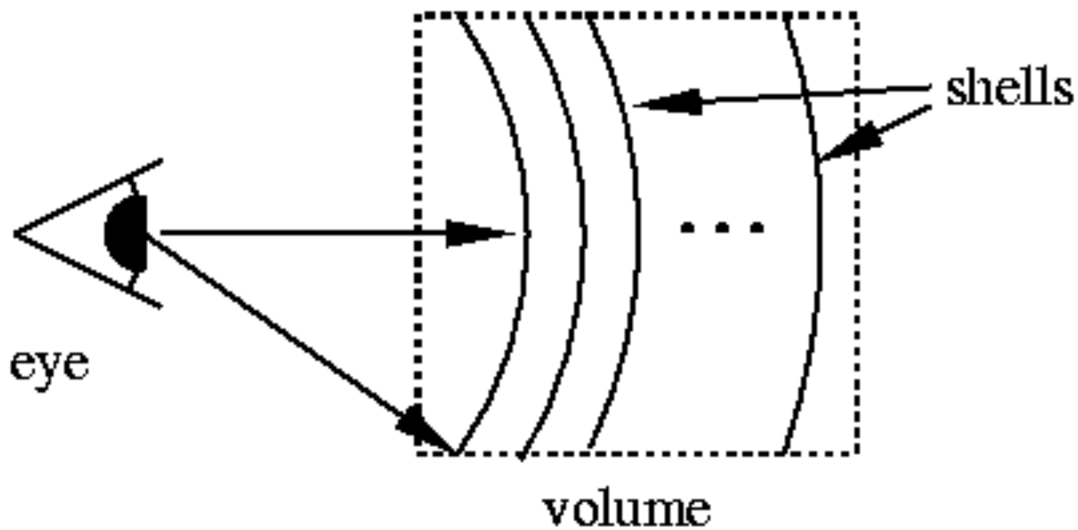
- Cubical voxels
  - Uniform sampling (texture sampling rate from slice to slice = texture sampling rate within each slice)

- Non-linear blending
  - Rescale the alpha values of data if the number of slices used to render the volume changes

Sampling Frequency (Contd.)

- Perspective
  - Increase the density of slices with distance from the viewer

- Flat versus Spherical Slices
  - Use spherical slices to get good close-ups of the data
  - Spheres are centered at the eye-point
  - Tessellate the spheres finely enough to avoid concentric shells from touching each other



Shrinking the Image

- Best visual quality
  - ➢ Render the volume image so that the size of a texel is about the same size of a pixel
- Reducing the volume size will cause the texel data to be sampled into a smaller area
- The smaller image can have density artifacts that are not in the original volume data
  - ➢ Better first render the image full size in the desired orientation and then shrink the resulting 2D image.

Virtualizing Texture Memory

- Volume data does not have to be limited to the maximum size of 3D texture memory

- Divide the data volume into a set of smaller "blocks"
  - Each brick is loaded into the texture memory, then data slices are textured and blended from the brick
  - Bricks can be processed from back-to-front order
  - The process is repeated until the entire volume is loaded.

Transfer Functions

- To highlight particular classes of volume data
- Uses OpenGL's lookup table to texel values during texturing
- If lookup tables are not available, the processing can be done to the volume data by the application before loading the volume.

Shading Textured Volume

- Data can be lit and shaded
  - Lighting needs to be computed per volume texel

- Two approaches:
  - Shading done on the host before you load the data as texture
  - Shading done while the texture being loaded

- Shading with texture:
  - Transform the texel data using the color matrix extension
  - Save the components of the gradient vectors as color components in the texture
  - Lighting can be done while the data is being visualized using three matrices

    Light direction matrix, color matrix, and texture color matrix

    Take dot product between the light and normal

    Color matrix is a part of the pixel path so the calculations are done in the pixel pipeline

- Interactive computation of data's gradient vectors can be done.

Transforming Textured Volume

- Clipping
  - Additional surfaces can be created on the volume with user defined clipping planes
  - A clipping plane can be used to cut through the volume, exposing a new surface

- Warping
  - Data volume can be warped by non-linear shifting the texture coordinates of the data slices
  - Tessellate the vertices to provide more vertex locations to perturb the texture coordinate values

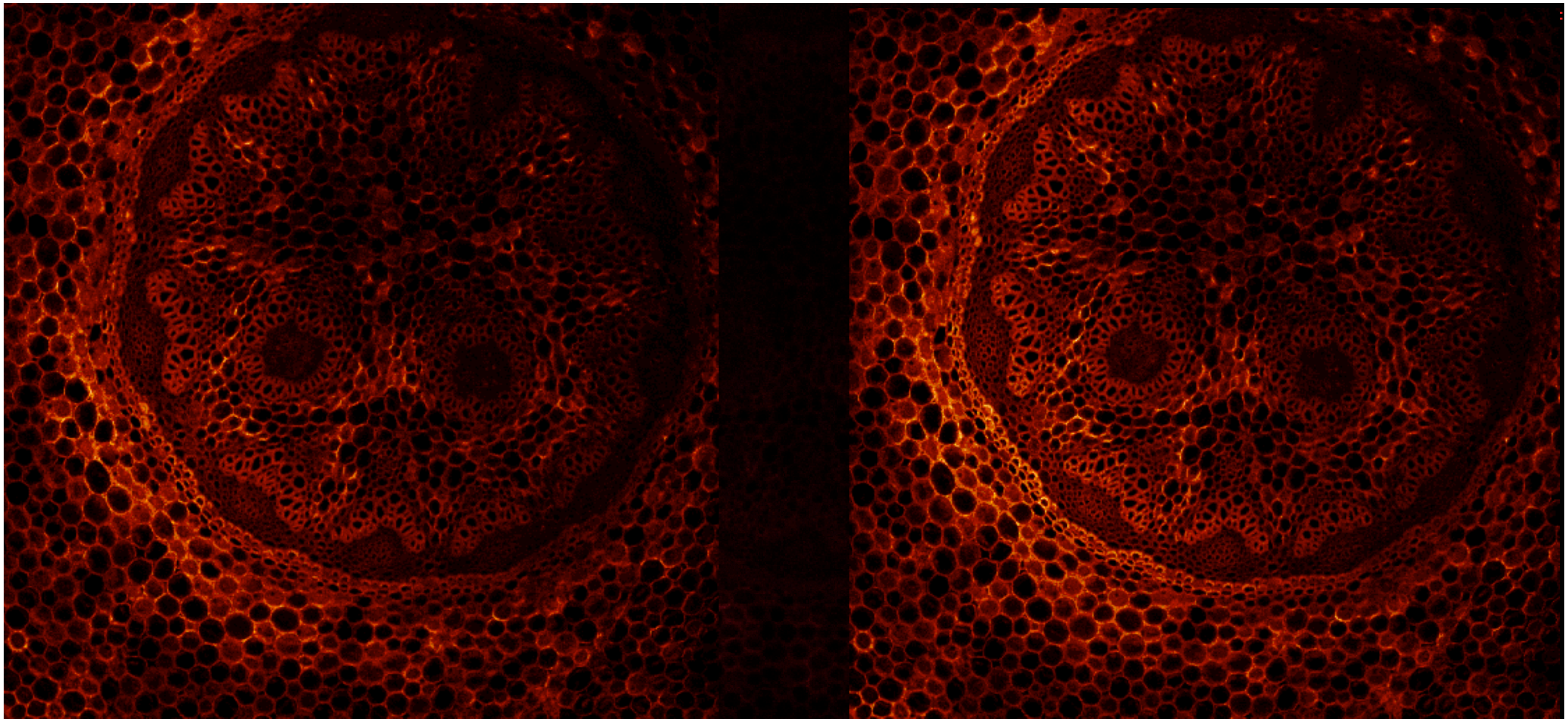
- Including geometric objects
  - Opaque objects are rendered along with the volumetric data slice using depth buffering for both.
    - Pixels on the data planes behind the object are not rendered
    - Planes in front of the object blend the object in; this blending gradually obscure the object, making it appear embedded in the volume data
  - Transparent objects must be rendered a slice at a time
    - Chopping the object into slabs using user defined clipping planes
    - Slab thickness corresponds to the spacing between the volume data slices.

Skipping Invisible Voxels

- Two types of invisble voxels
  - Empty voxel (with fully transparency)
  - Occluded voxel (lying behind other voxels)
  - How to skip such voxels in texture-based rendering
- Partition a volume dataset into sub-volumes based on similarity of voxels
  - Consider volume domain (position) and transfer function domain (densities and gradient magnitudes)
  - Subdivide the texture space into an octree
  - Use texture hulls (of any shape) of all connected non-empty regions
  - Growing boxes (can be represented as orthogonal BSP tree)
- Skip subvolumes containing transparent voxels for rendering
- Cull or clip occluded voxels with orthogonal opacity map updated from the contents of the frame buffer during rendering

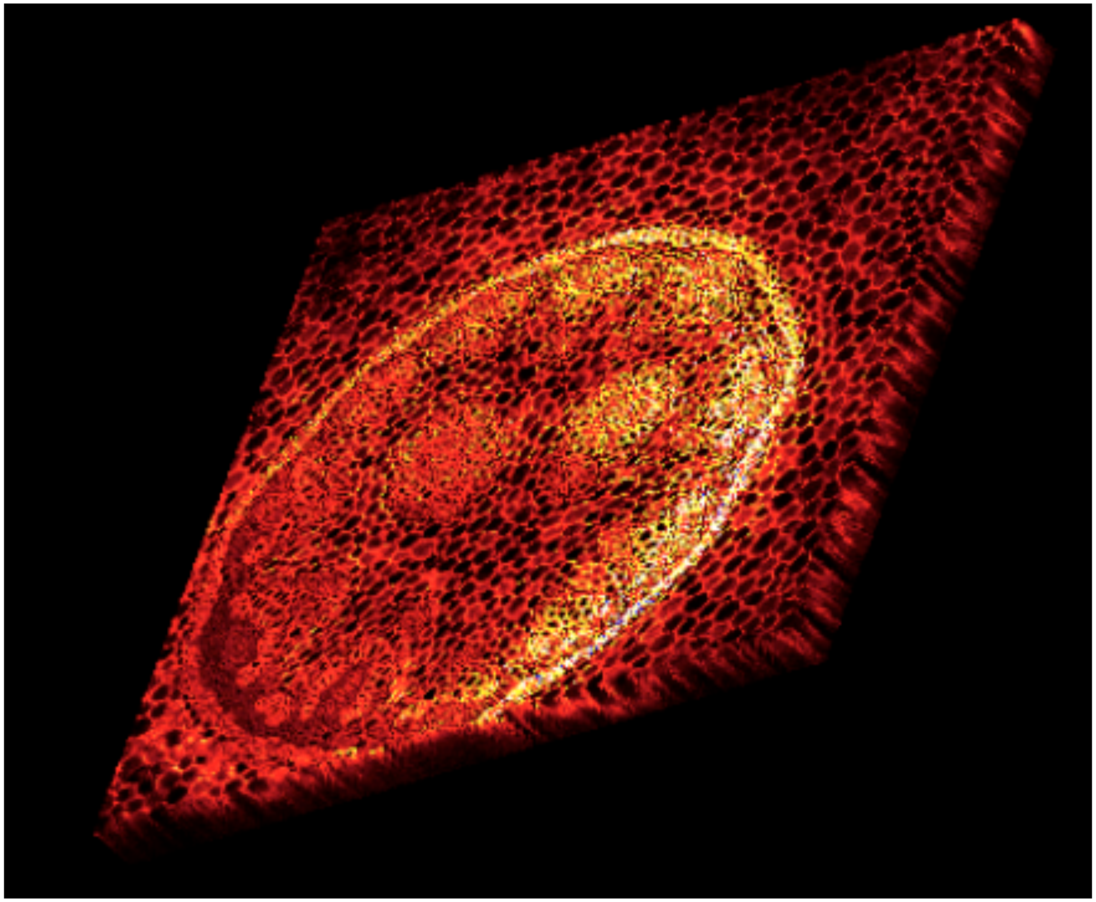
Confocal 2D Images

- Confocal 2D images
  - Plant stem (J. Lynn at LSU Biology)



3D Construction

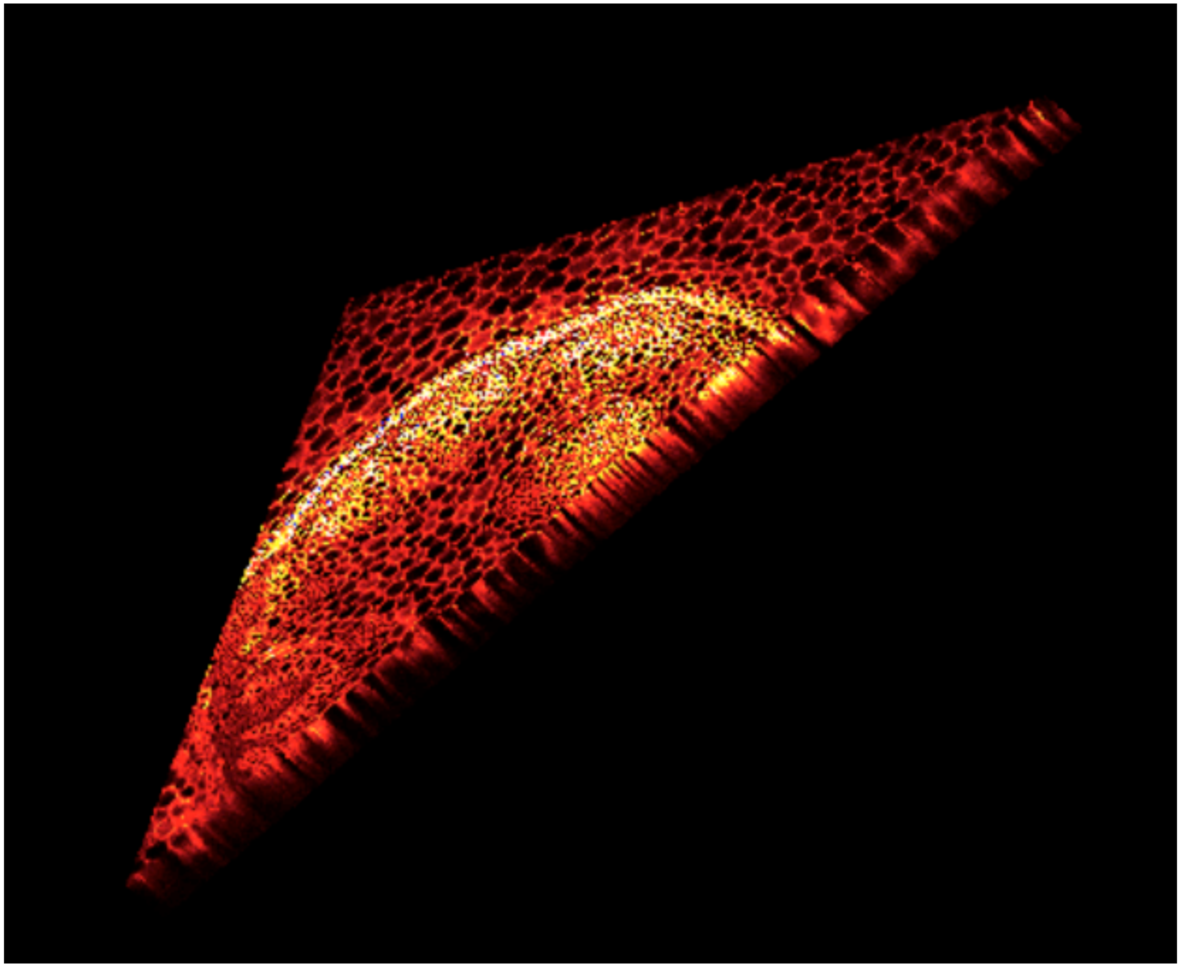
- Load individual 2D images and stack them one above the other
  - A third dimension is thus added along the direction of stacking to get 3D data
- Two ways for 3D reconstruction:
  - Texture-based
    - Each slice an individual texture object
    - All slices together as a 3D texture object
  - Pixel-by-pixel
    - Each slice is treated as an array of pixels.



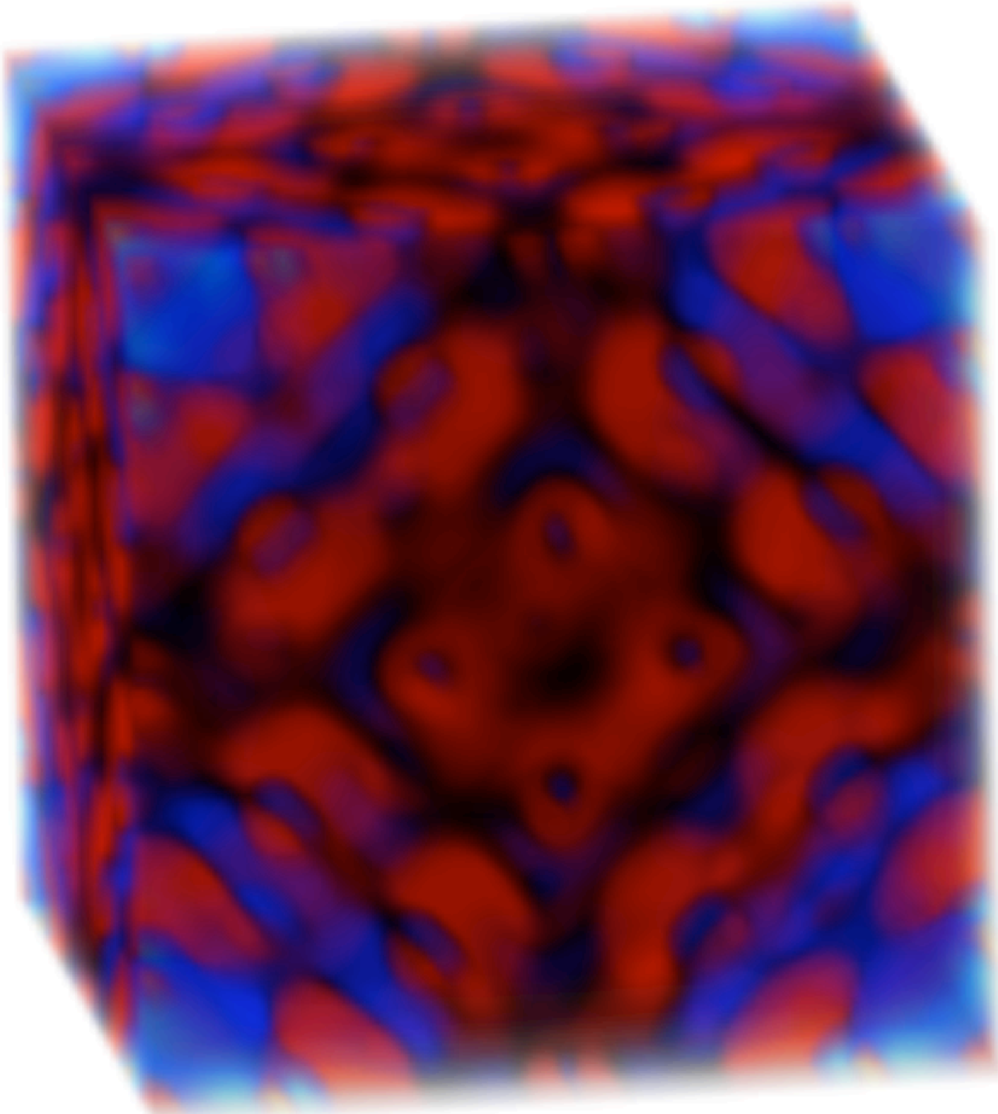
Rotated view of stacked tiff images

Using Clipping

- Clipping the 3D data image along arbitrary plane
- Set and adjust clipping plane
- Select the best view mode.



Volume Rendering with Texture



Clipping with Texture