Directly Visualizing Volume Data

Mike Bailey

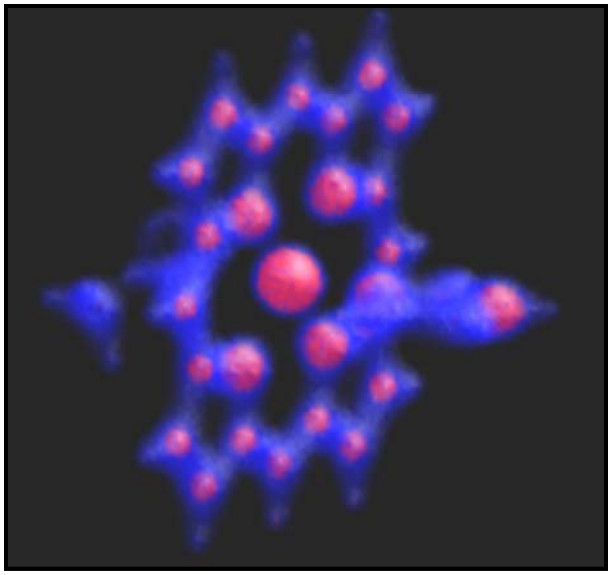
mjb@cs.oregonstate.edu

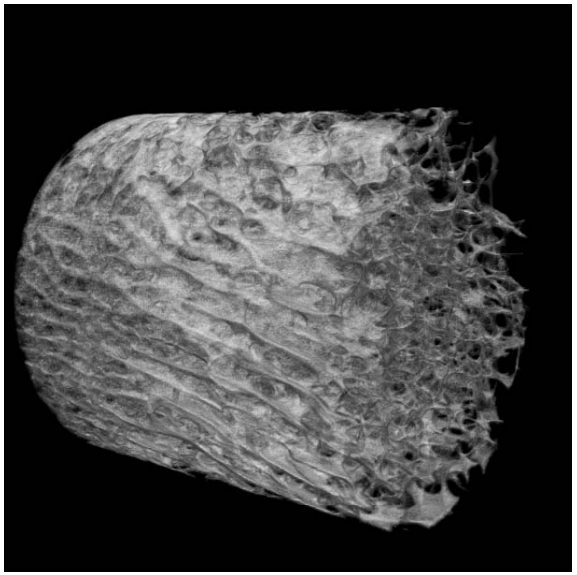
Oregon State University

Volume Data: A Visual Definition

Why Do We Care About Volume Visualization?

- **Medical: CAT, MRI, 3D ultrasound**
- **Science and engineering: CFD, stress, thermal, molecular**
- **Volumes are normally very difficult to comprehend**



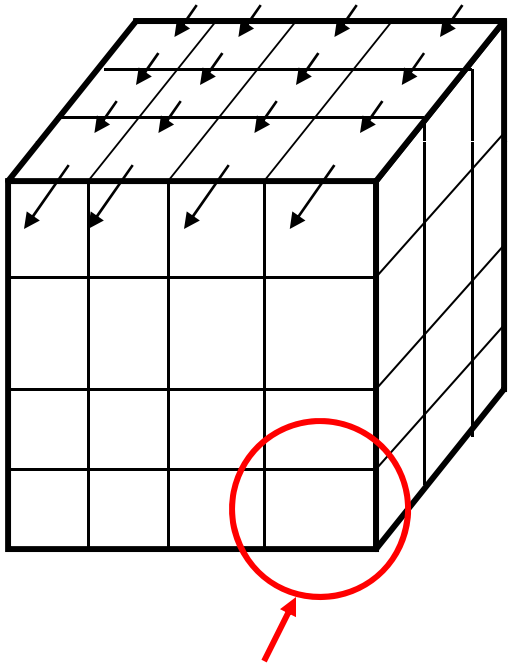


Understanding Volume Data Usually Involves a Compromise

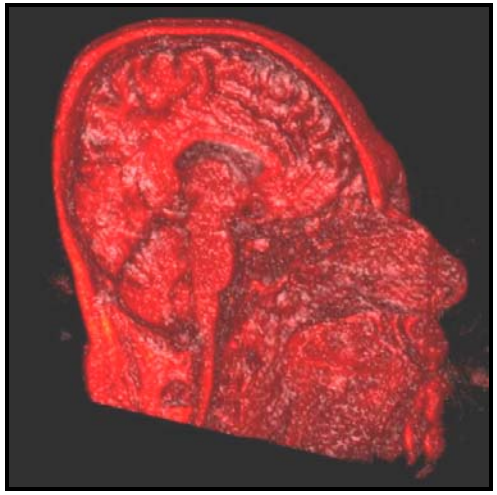
Point Clouds	→	All values everywhere, hard to see much, artifacts
Interpolated-colors cutting planes	→	All values in a plane
Contours cutting plane	→	Discrete values in a plane
Isosurfaces	→	One value everywhere

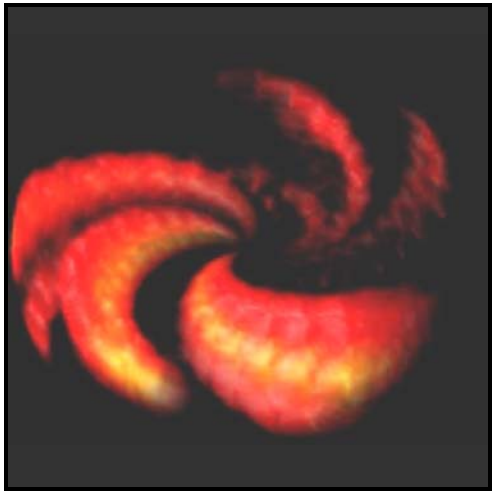
Direct Volume Rendering

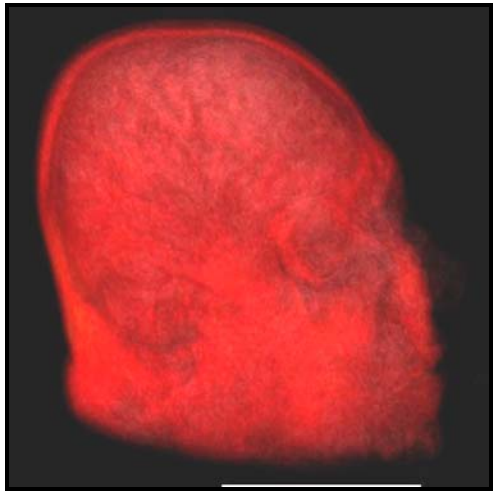
Composite the colors and alphas of the voxels



A *Volume Element*, or *voxel*



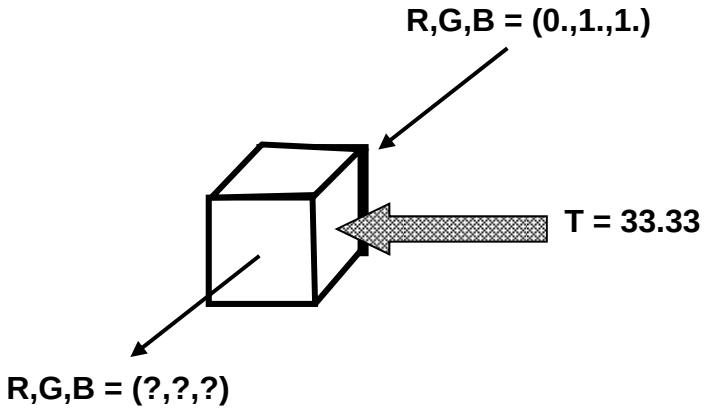




TMIN = 0.
TMAX = 100.

The color transfer function is a **Black-Red-Yellow-White heated object scale**, mapping a scalar value of 0. to Black, and 100. to White.

The opacity transfer function is a linear ramp so that the opacity is 1. (opaque) when T = 100. and 0. (transparent) when T = 0.



You are compositing back-to-front through the volume. At this moment, the running values of RGB are **(0., 1., 1.) .** The next voxel you encounter has a **T value of 33.33**

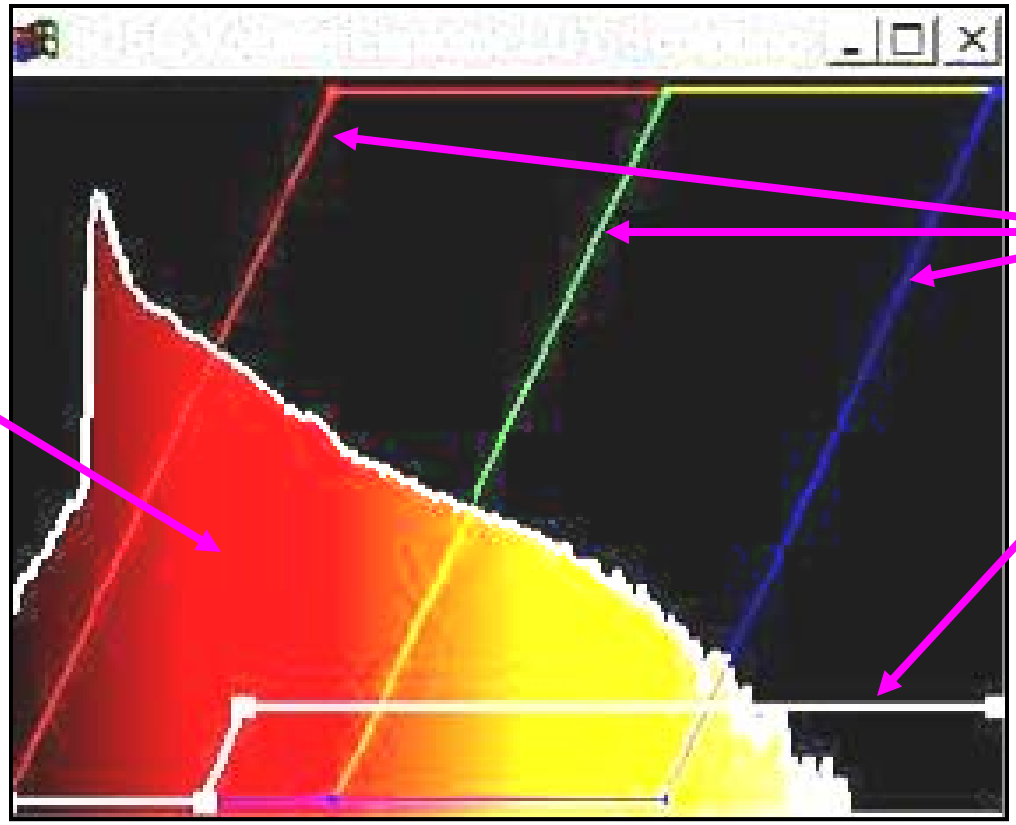
1. **What is the color of *just this voxel*?**
2. **What is the opacity of *just this voxel*?**
3. **What will the new running RGB values be when you are done compositing this voxel with the old running RGB values?**

What is the color of *just this voxel*?

What is the opacity of *just this voxel*?

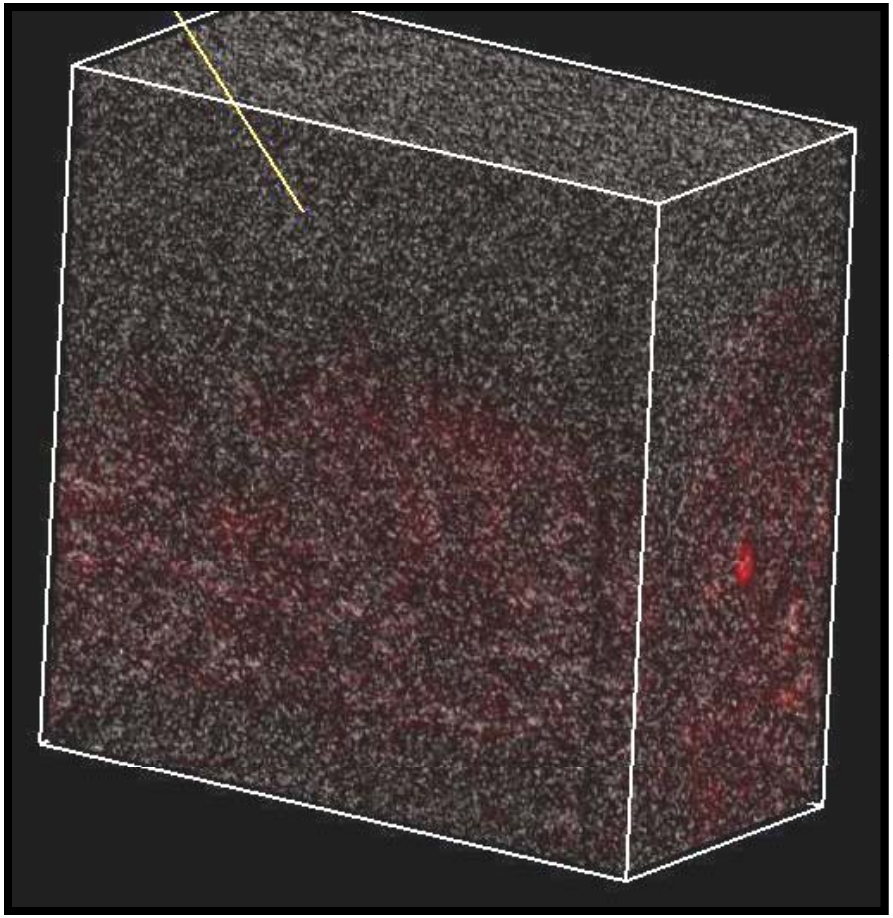
What will the new running RGB values be when you are done compositing this voxel with the old running RGB values?

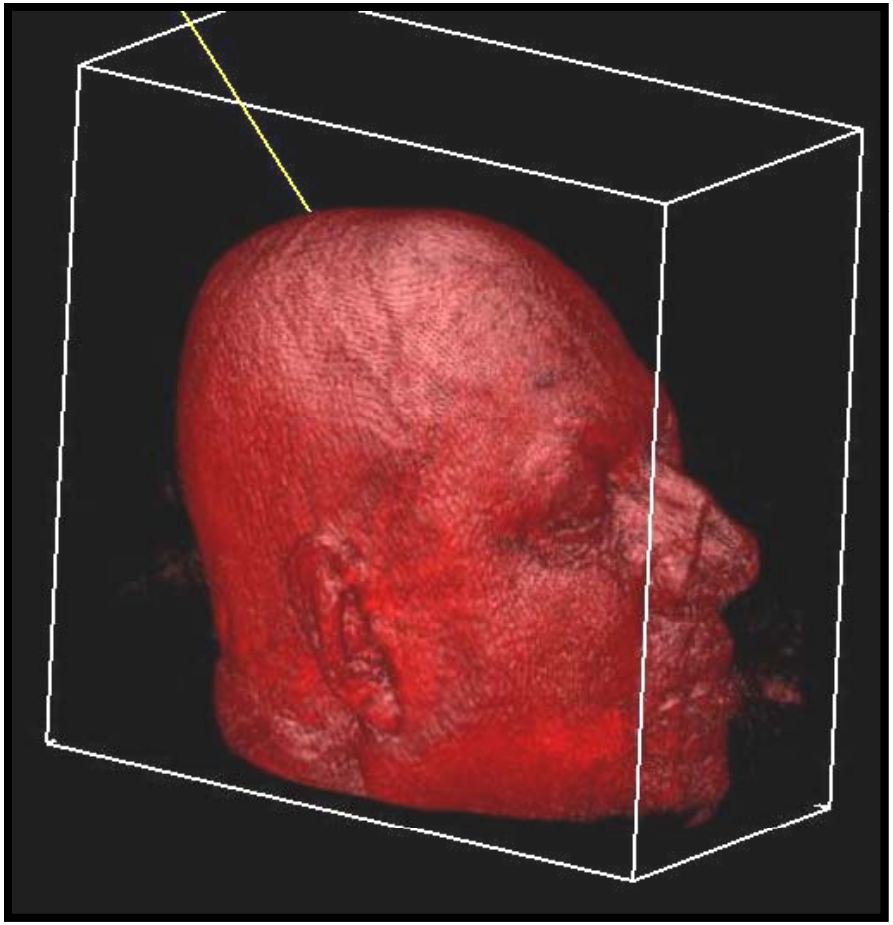
Transfer Function



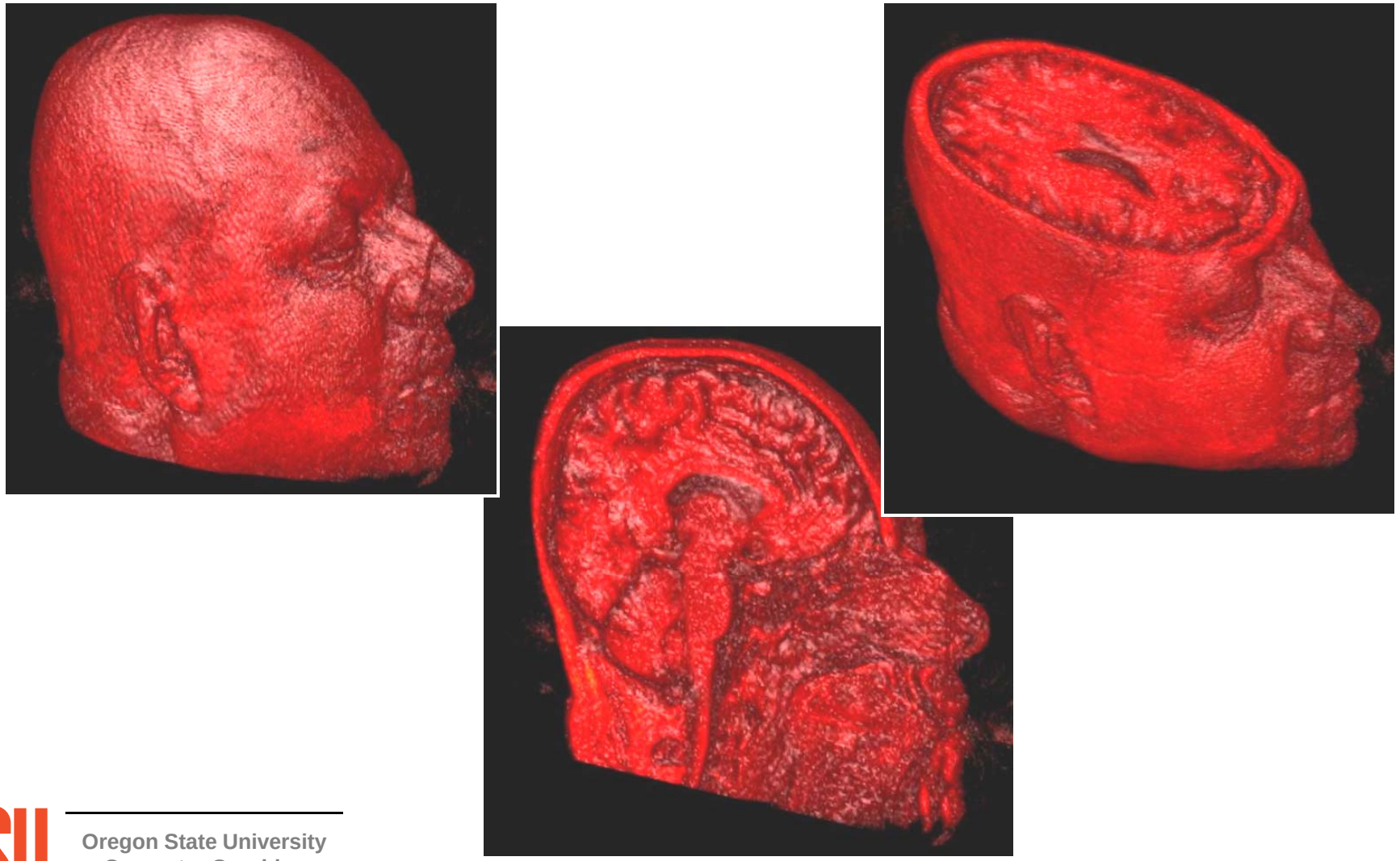
OSU *vx* Transfer Function Sculpting Window

Scalar-Data Cropping the Volume

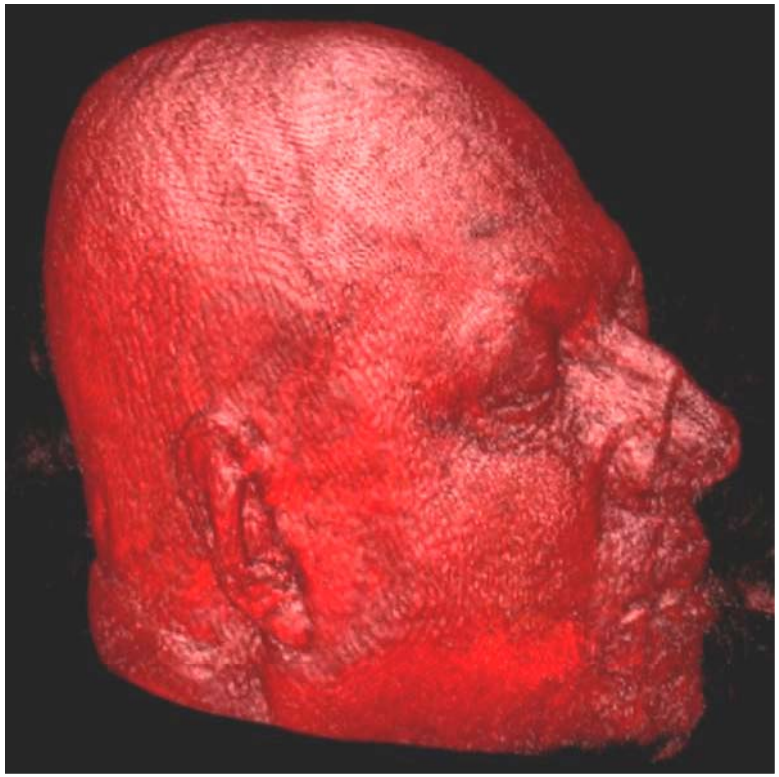


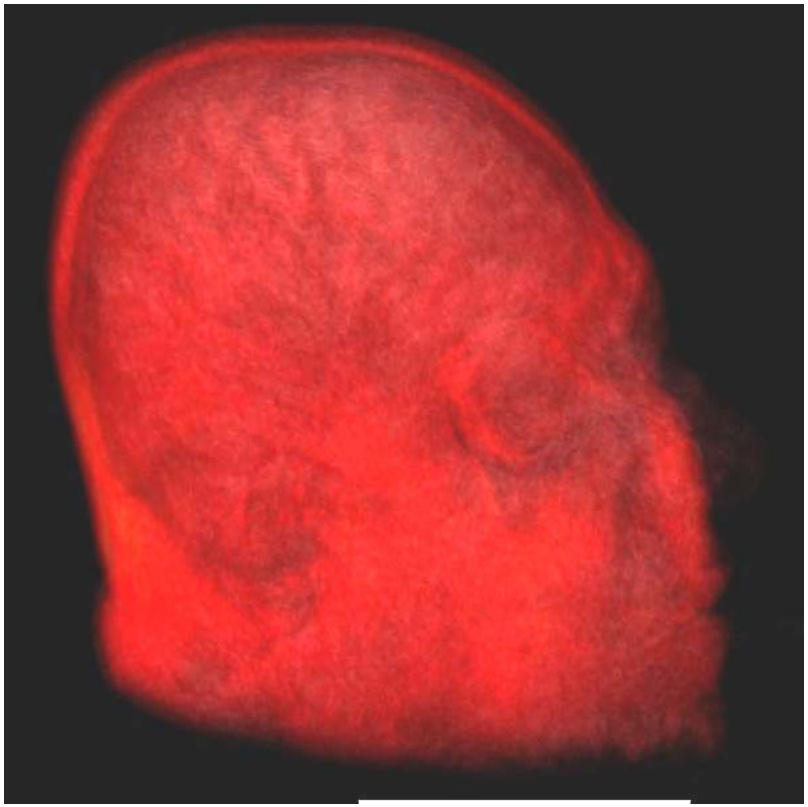


Spatially Cropping the Volume



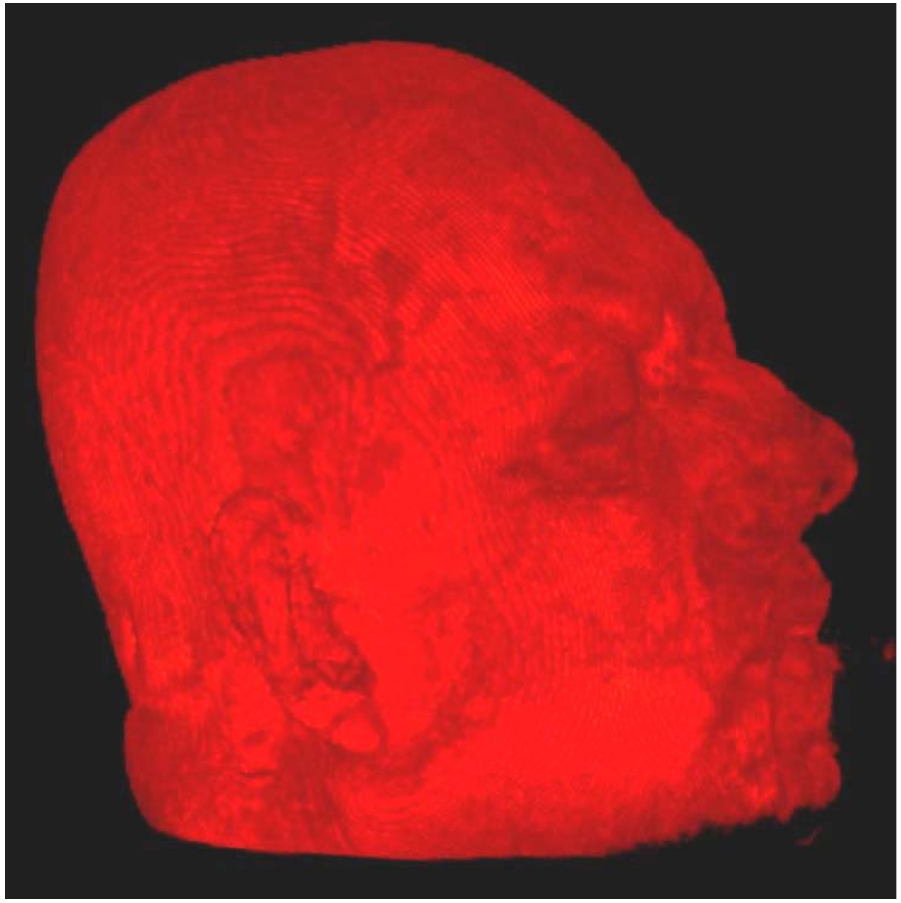
Change in Maximum Opacity

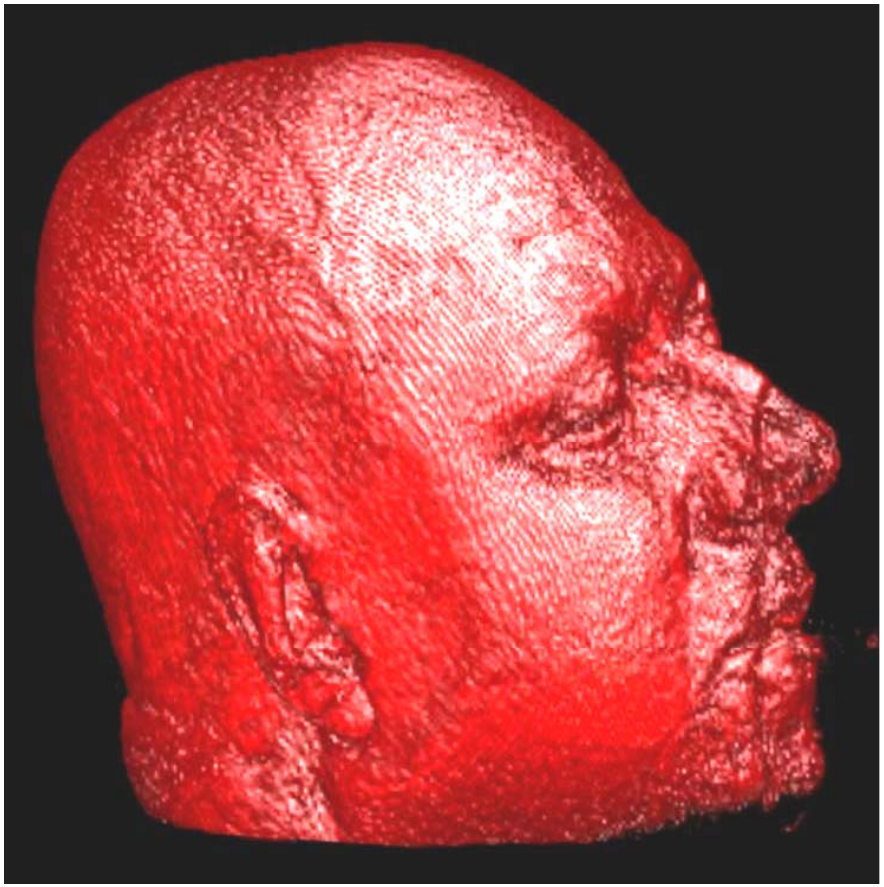




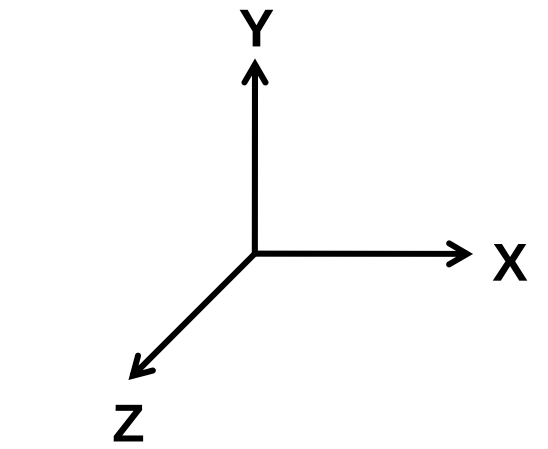
“Magic Lens” to selectively look inside

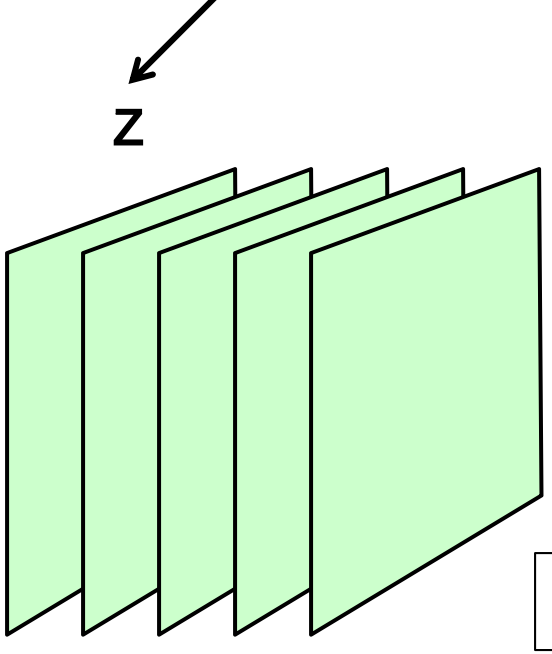
Lighting

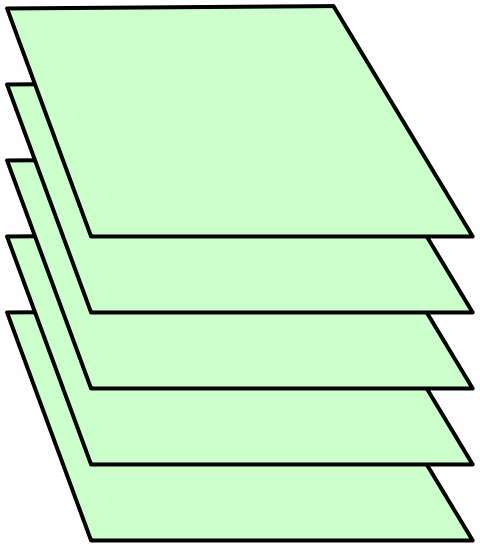




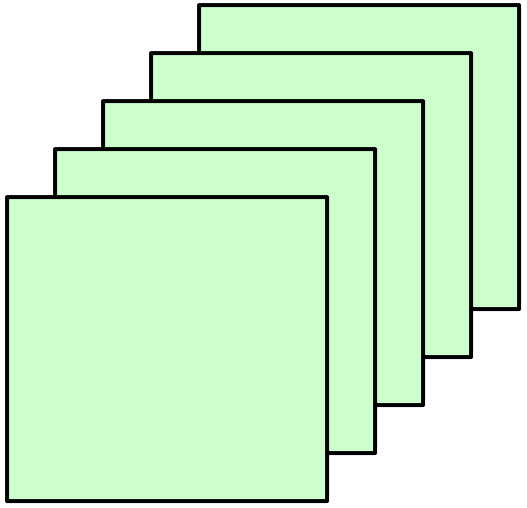
Volume Rendering with Parallel Texture Planes







unsigned char TextureXZ[NY][NX][NZ][4];
"NY slices of an NX by NZ texture"



unsigned char TextureYZ[NX][NY][NZ][4];
"NX slices of an NY by NZ texture"

unsigned char TextureXY[NZ][NX][NY][4];
"NZ slices of an NX by NY texture"

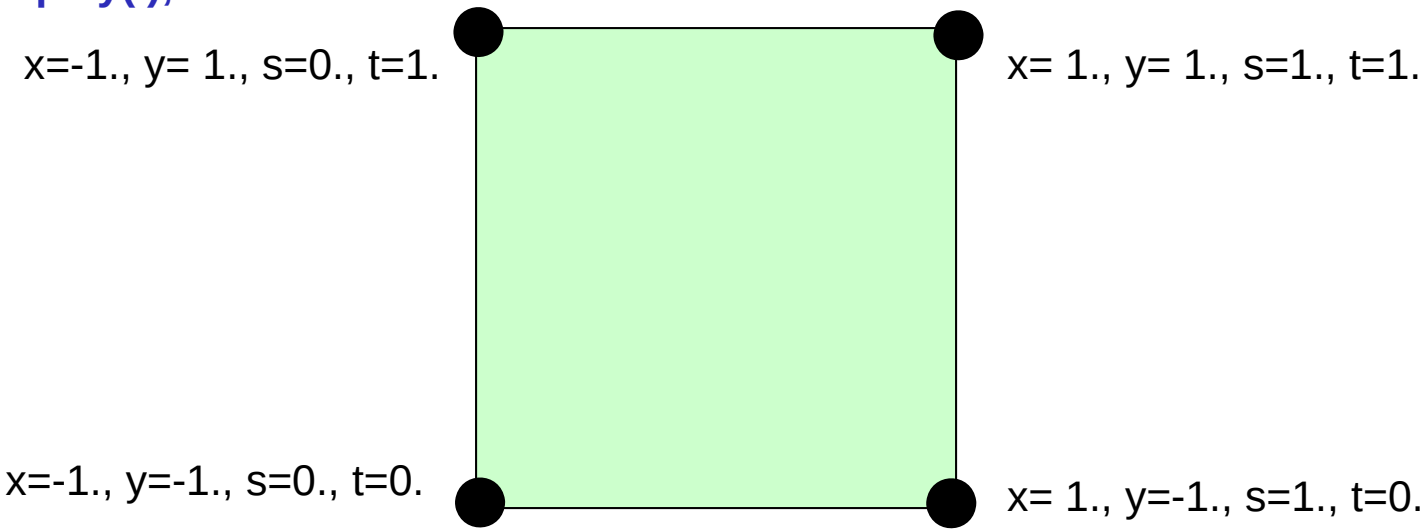
In a callback that is called whenever the opacity transfer function changes:

```
void
FillXY( void )
{
    float alpha;        // opacity at this voxel
    float r, g, b;      // running color composite

    for( int x = 0; x < NX; x++ )
    {
        for( int y = 0; y < NY; y++ )
        {
            r = g = b = 0.;
            for( int zz = 0; zz < NZ; zz++ )
            {
                // which direction to fill:

                int z;
                if( Zside == PLUS )
                    z = zz;
                else
                    z = ( NZ-1 ) - zz;

                if(  … this scalar value is not in the range you want to view …  )
                {
                    r = g = b = 0.;
                    alpha = 0.;
                }
                else
                {
                    r = Nodes[x][y][z].r;
                    g = Nodes[x][y][z].g;
                    b = Nodes[x][y][z].b;
                    alpha = MaxAlpha;
                }

                TextureXY[zz][y][x][0] = (unsigned char) ( 255.*r + .5 );
                TextureXY[zz][y][x][1] = (unsigned char) ( 255.*g + .5 );
                TextureXY[zz][y][x][2] = (unsigned char) ( 255.*b + .5 );
                TextureXY[zz][y][x][3] = (unsigned char) ( 255.*alpha + .5 );
            }
        }
    }
}
```

In Display( ), I:

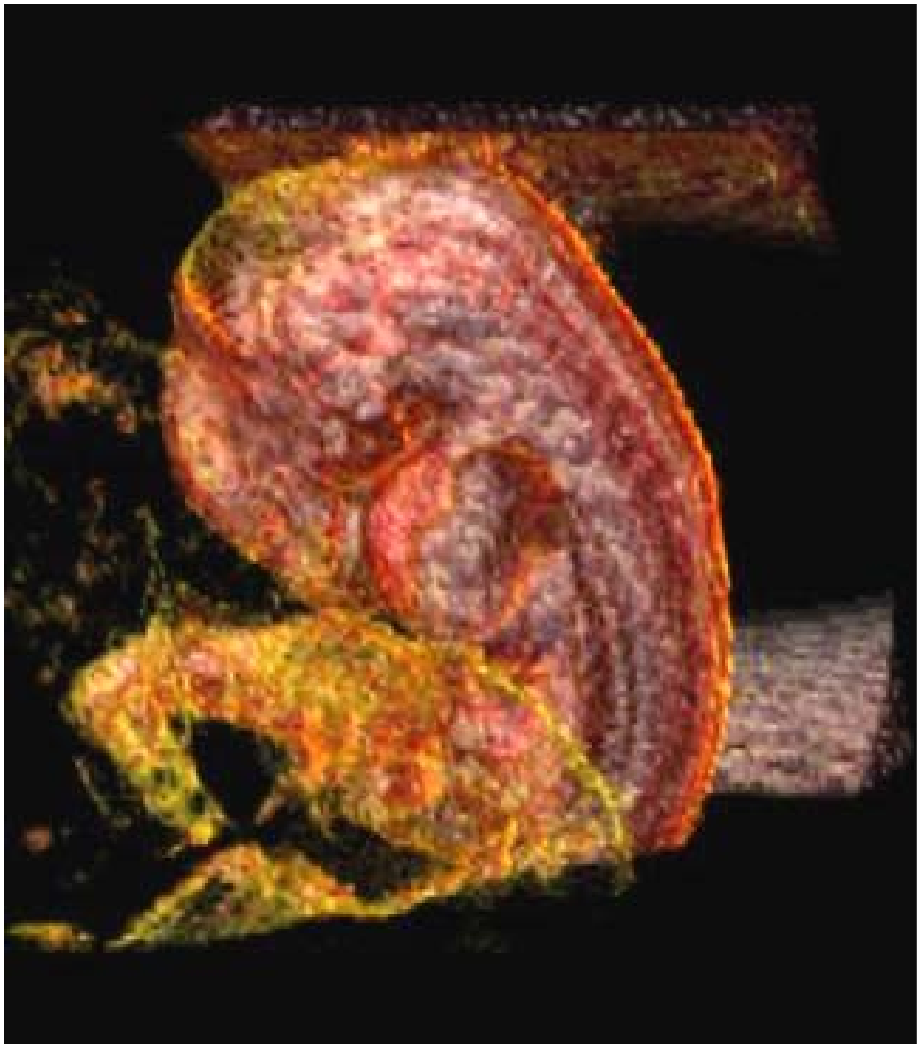
```
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );
glTexEnvf( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE );

int  filter = GL_NEAREST;
if( Bilinear )
    filter = GL_LINEAR;
else
    filter = GL_NEAREST;

glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, filter );
glTexParameterf( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, filter );
glPixelStorei( GL_UNPACK_ALIGNMENT, 1 );
glEnable( GL_TEXTURE_2D );


glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );
glEnable( GL_BLEND );

 DetermineVisibility( );

 float z0,  dz;
if( Major == Z )
{
    if( Zside == PLUS )
    {
        z0 = -1.;                                   // back-to-front
        dz = 2. / (float)( NZ - 1 );
    }
    else
    {
        z0 = 1.;                                    // front-to-back
        dz = -2. / (float)( NZ - 1 );
    }
```

In Display( ), II:

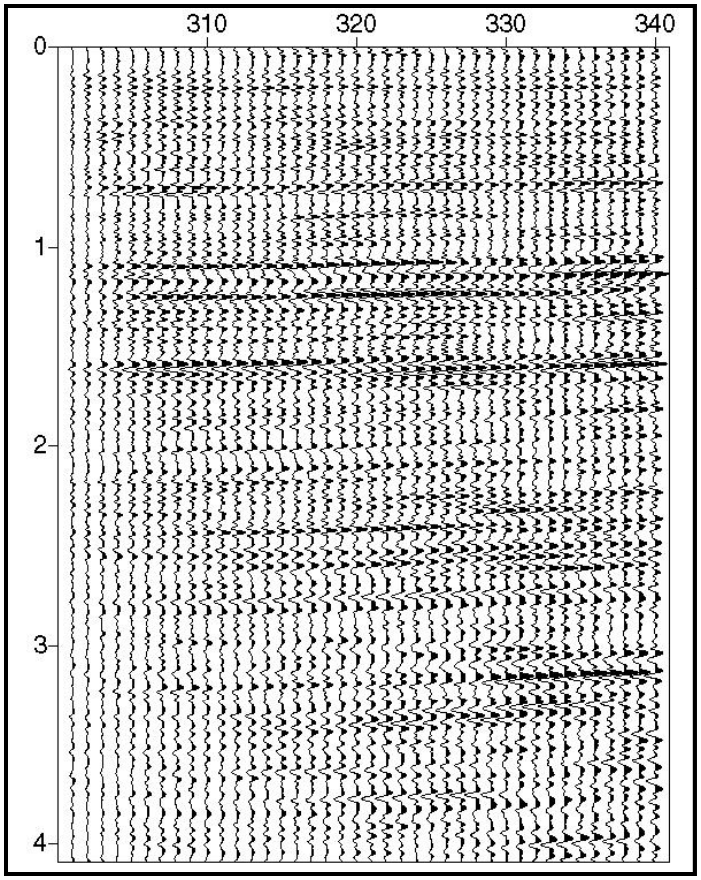


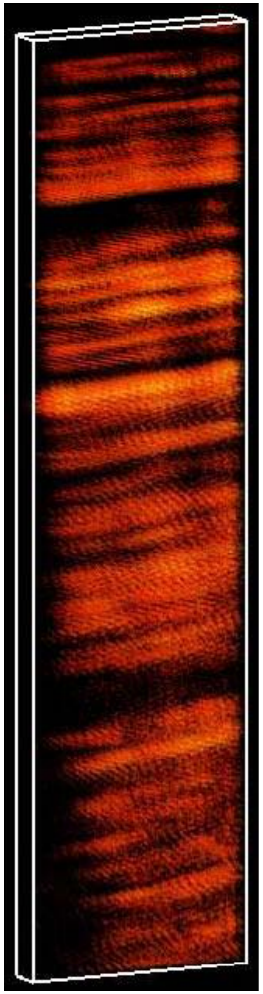

```
	glBegin( GL_QUADS );
	for( z = 0; z < NZ; z++, zcoord += dz )
	{
		glTexImage2D( GL_TEXTURE_2D, 0, 4, NX, NY, 0, GL_RGBA, GL_UNSIGNED_BYTE, &TextureXY[z][0][0][0] );

		glTexCoord2f( 0.f, 0.f );
		glVertex3f( -1.f, -1.f, zcoord );

		glTexCoord2f( 1.f, 0.f );
		glVertex3f( 1.f, -1.f, zcoord );

		glTexCoord2f( 1.f, 1.f );
		glVertex3f( 1.f, 1.f, zcoord );

		glTexCoord2f( 0.f, 1.f );
		glVertex3f( -1.f, 1.f, zcoord );
	 }
	glEnd( );

}		//  if( Major == Z )
```

Human Embryo

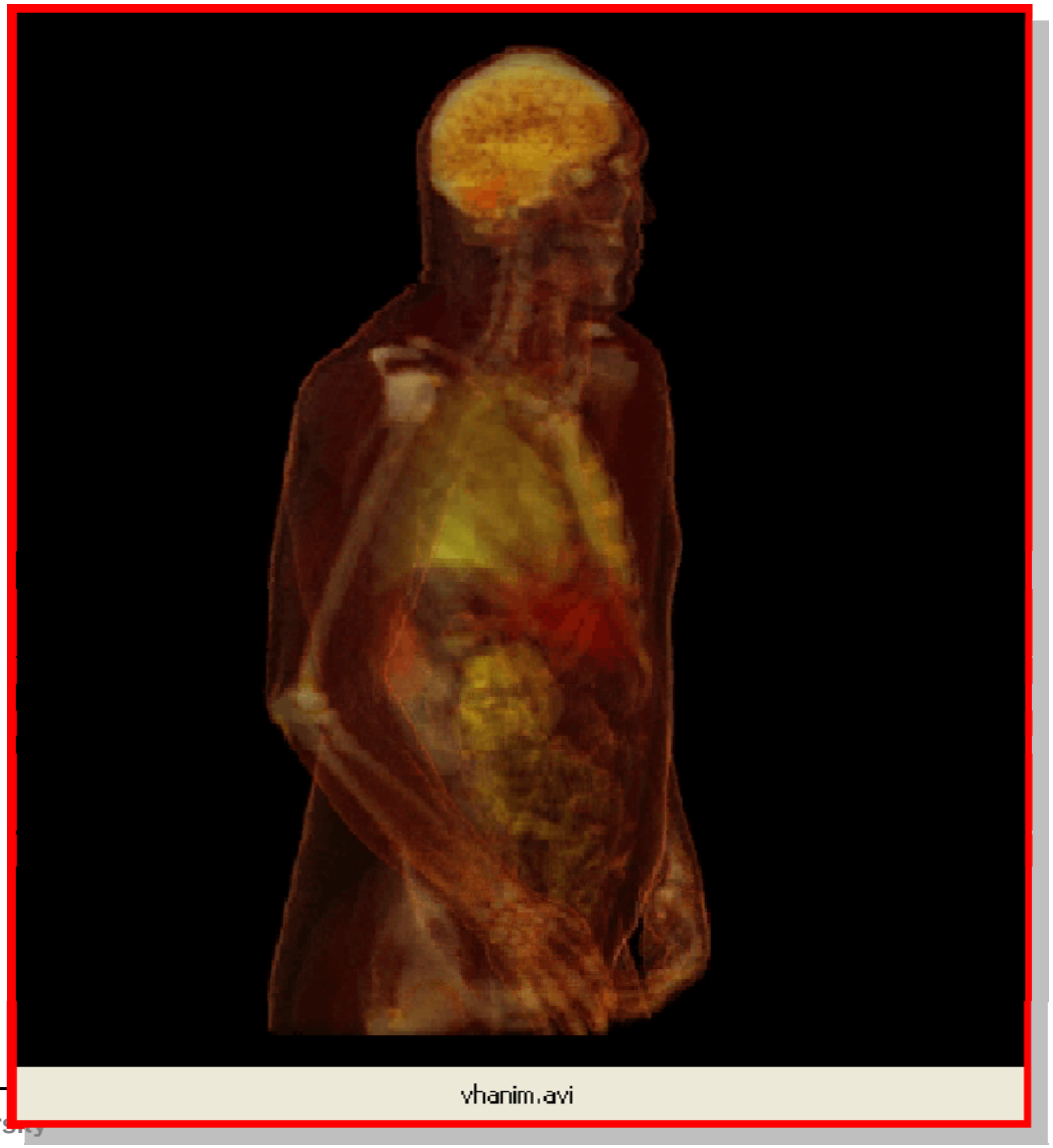


Geophysics

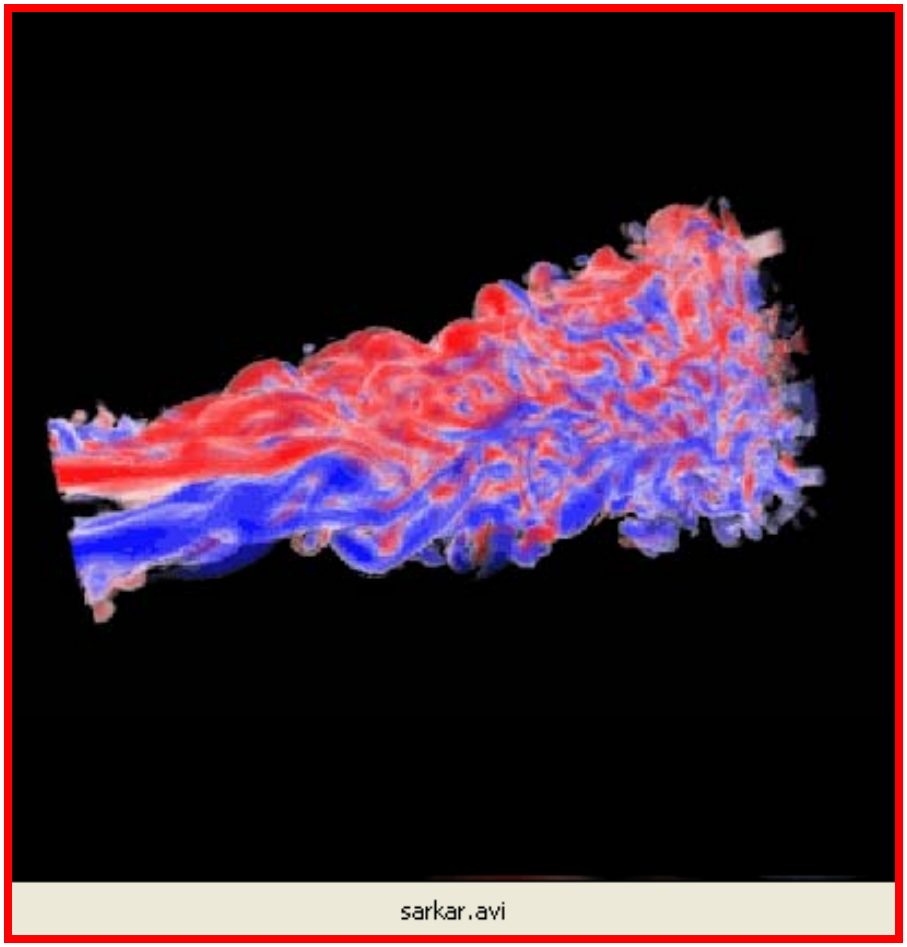


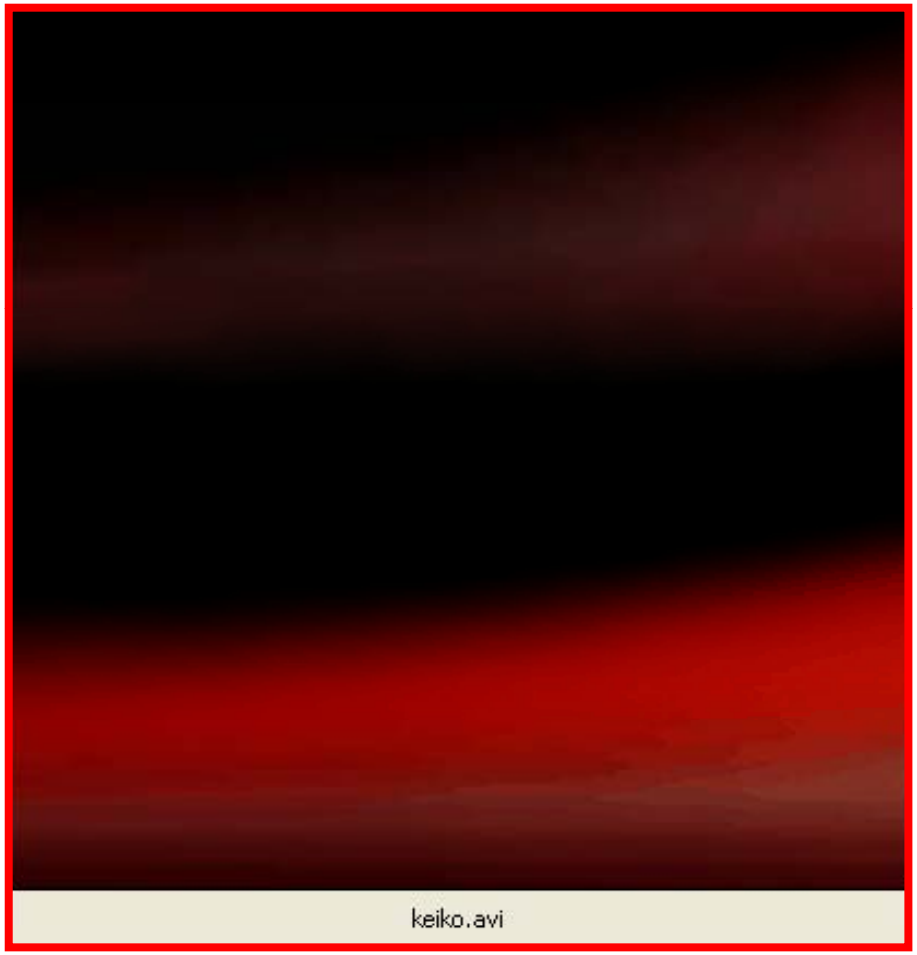


Volume Interaction: The Visible Human

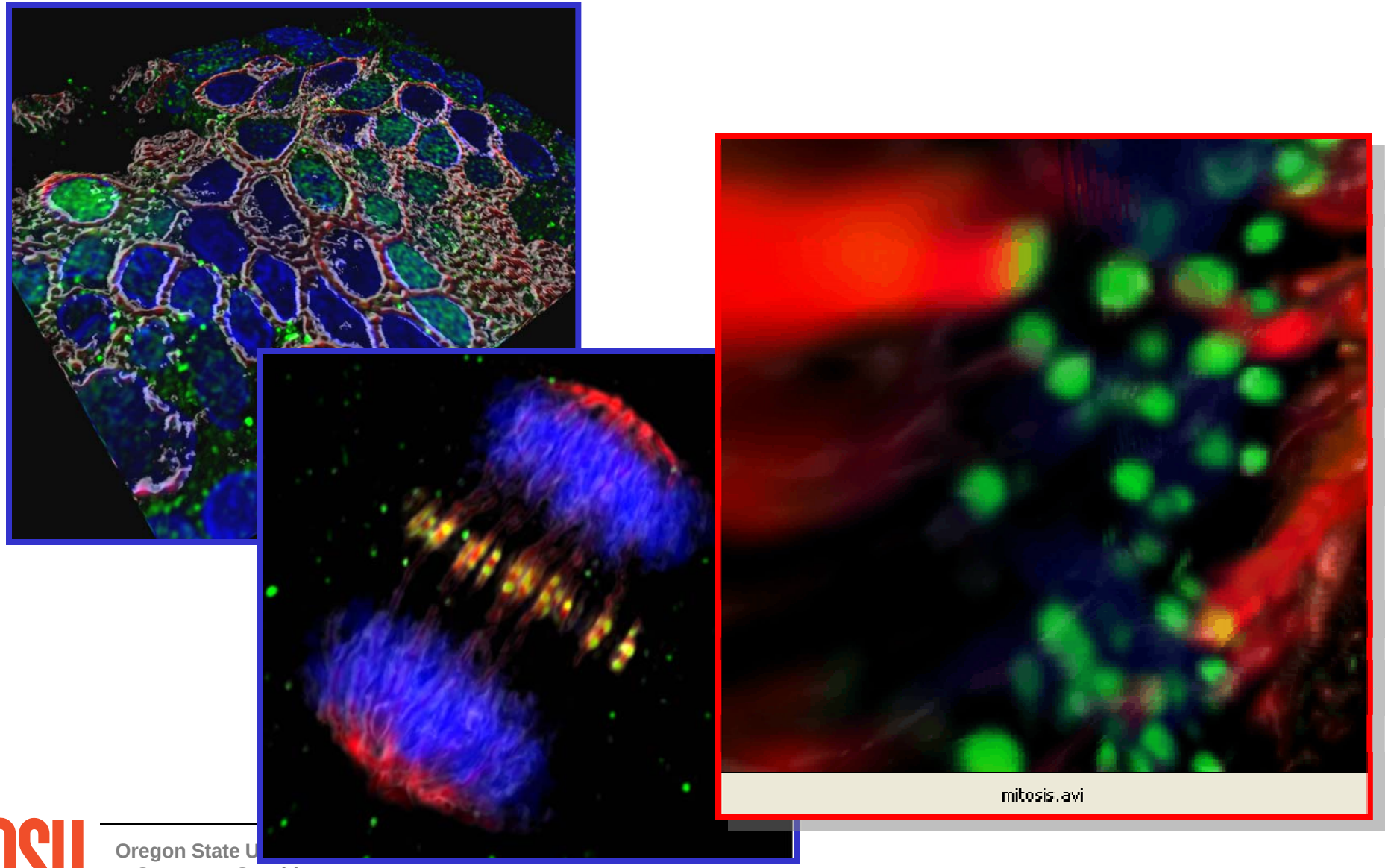


Interactive Volume Visualization for Computational Fluid Dynamics

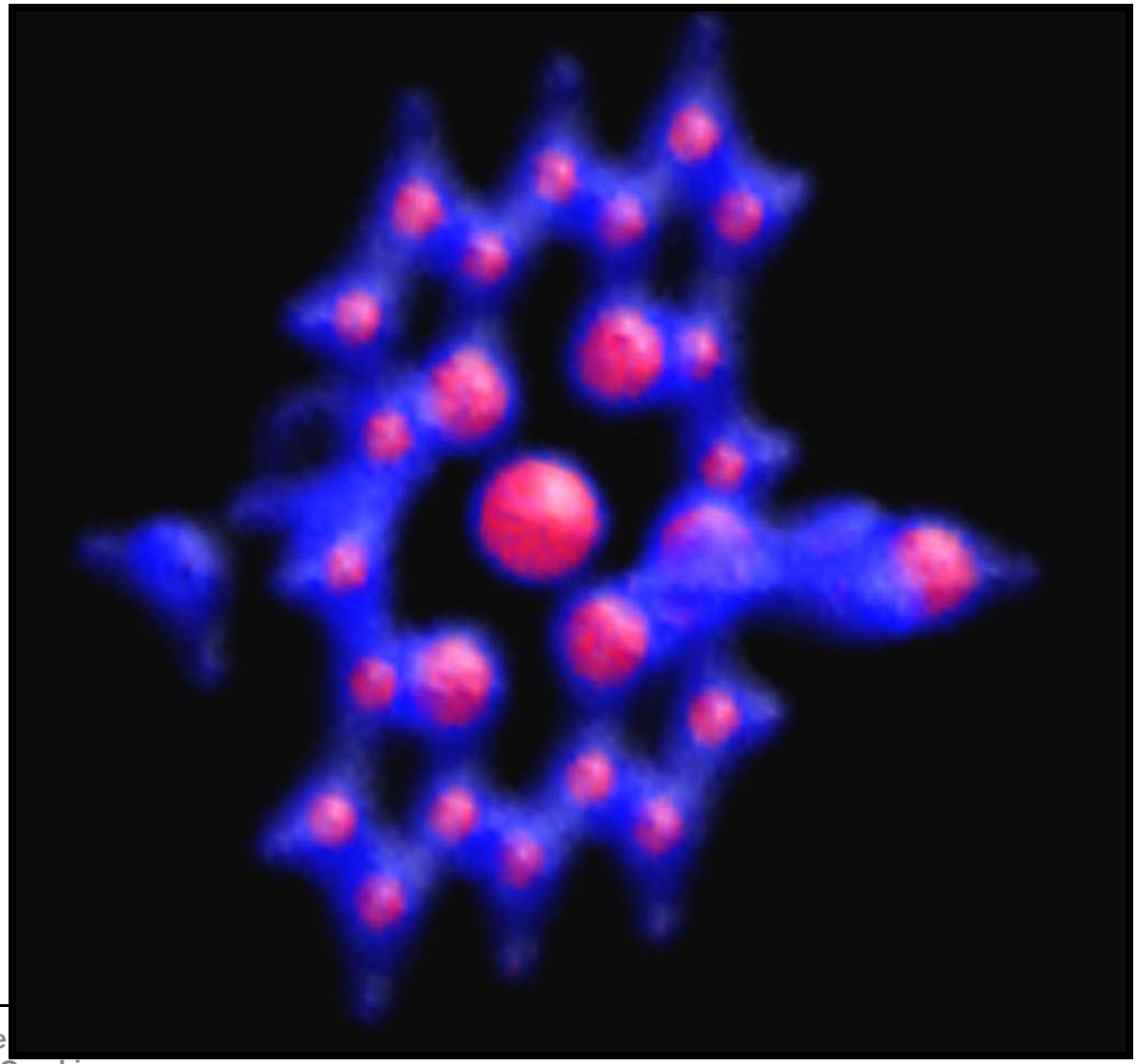




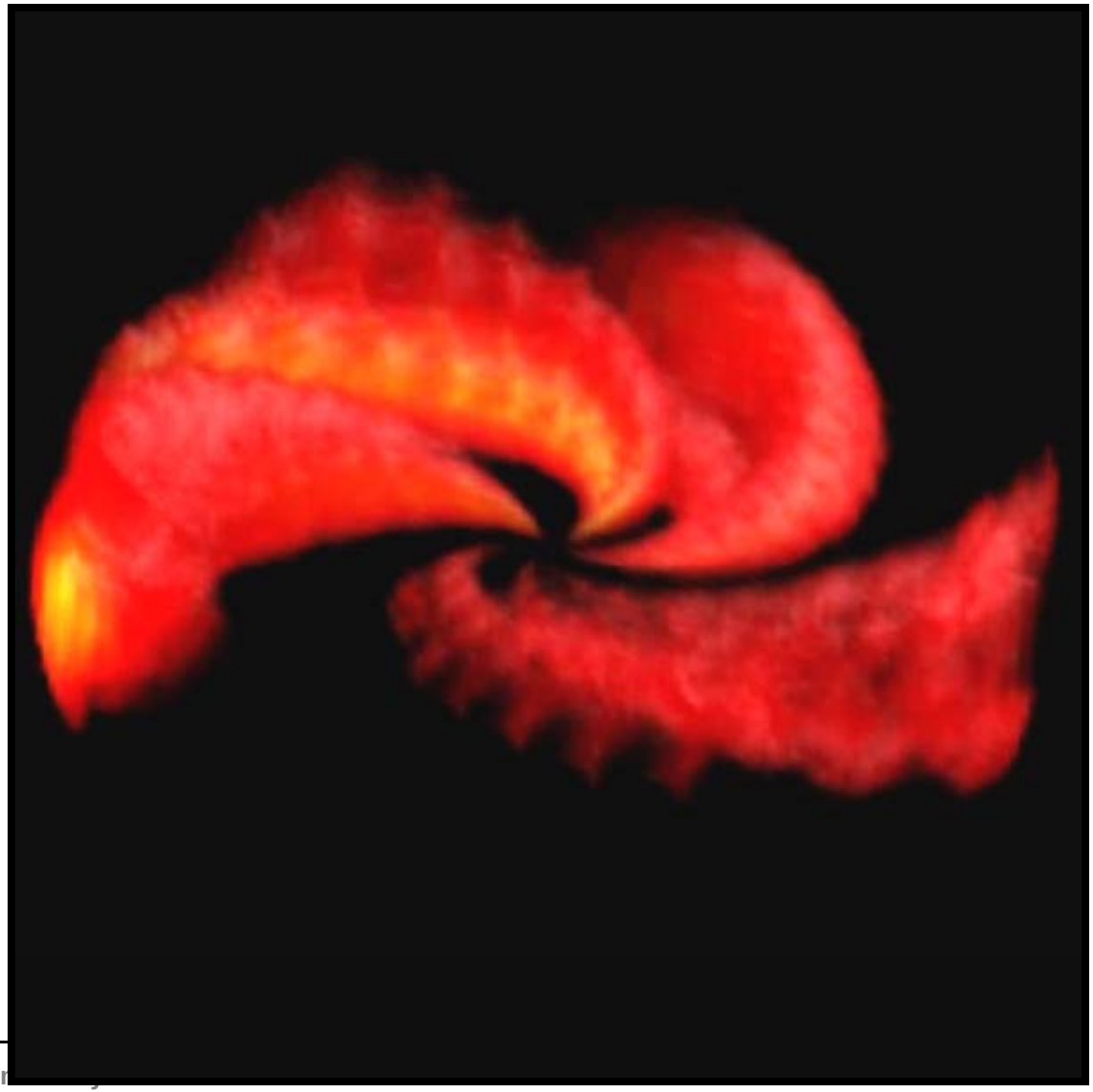
Volume Interaction in Cancer research



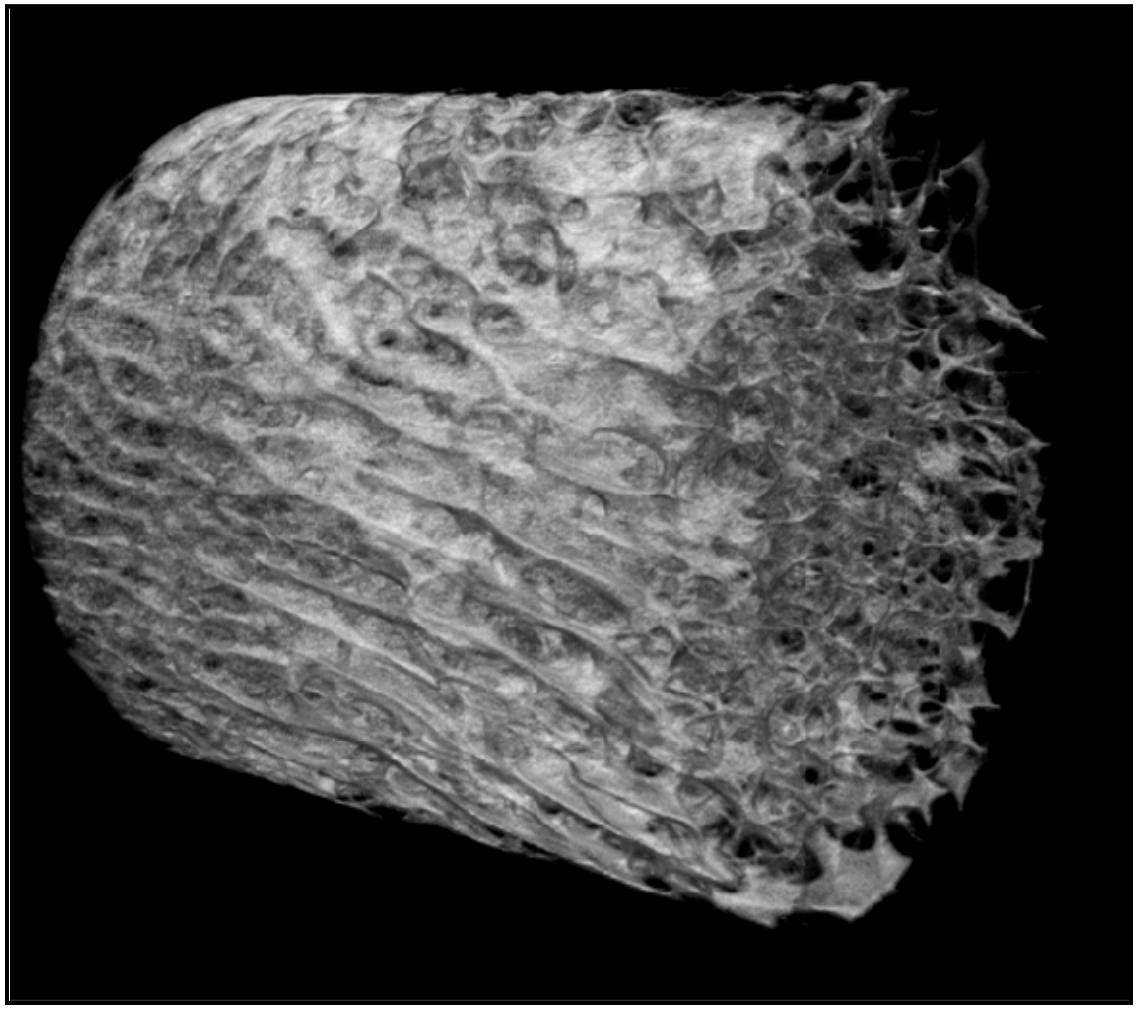
Molecular Science



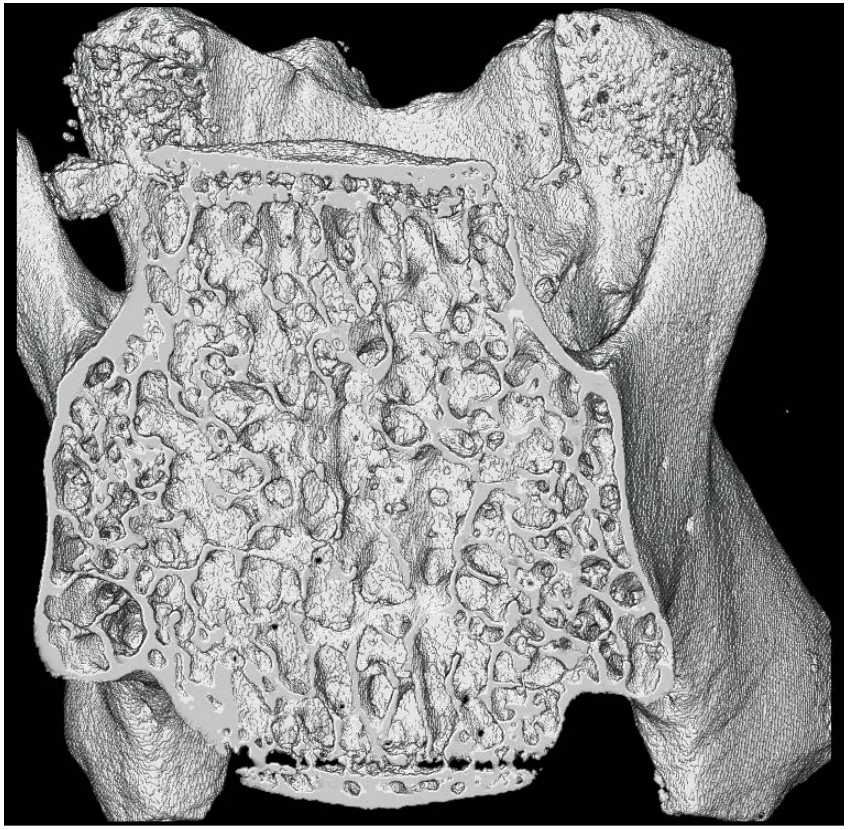
Solar Wind



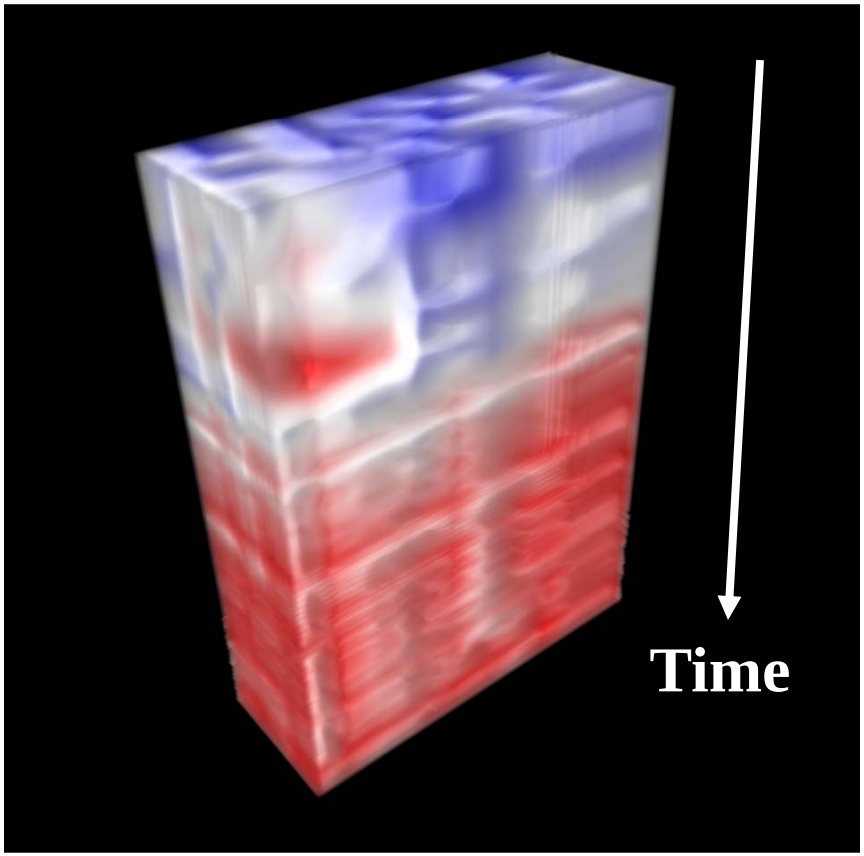
OSU Sheepbone

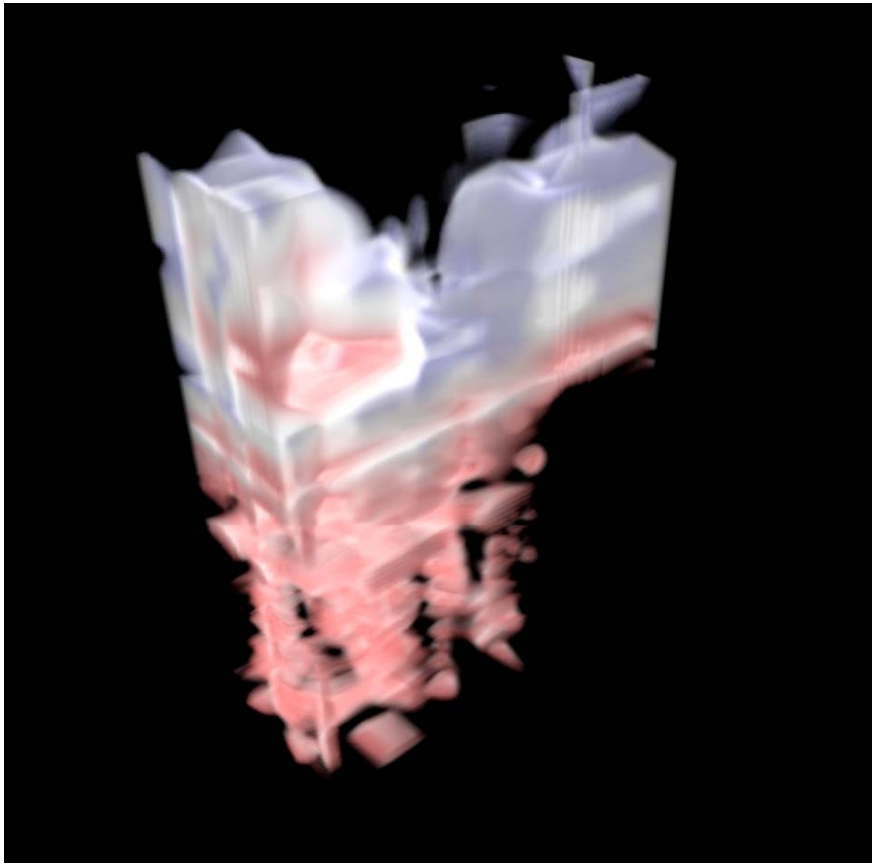


OSU Mouse Vertrebra



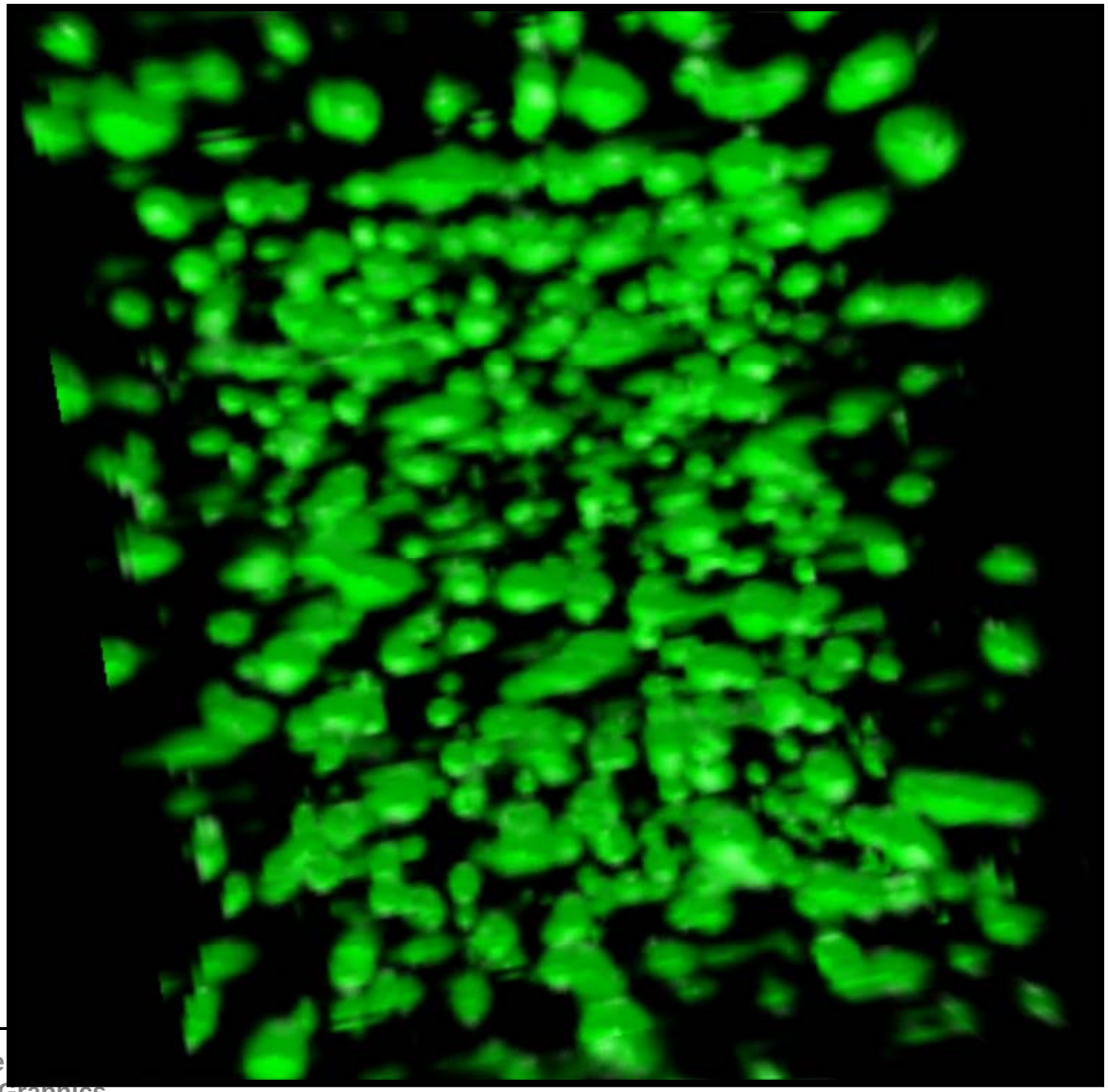
OSU Hillslope Water Saturation



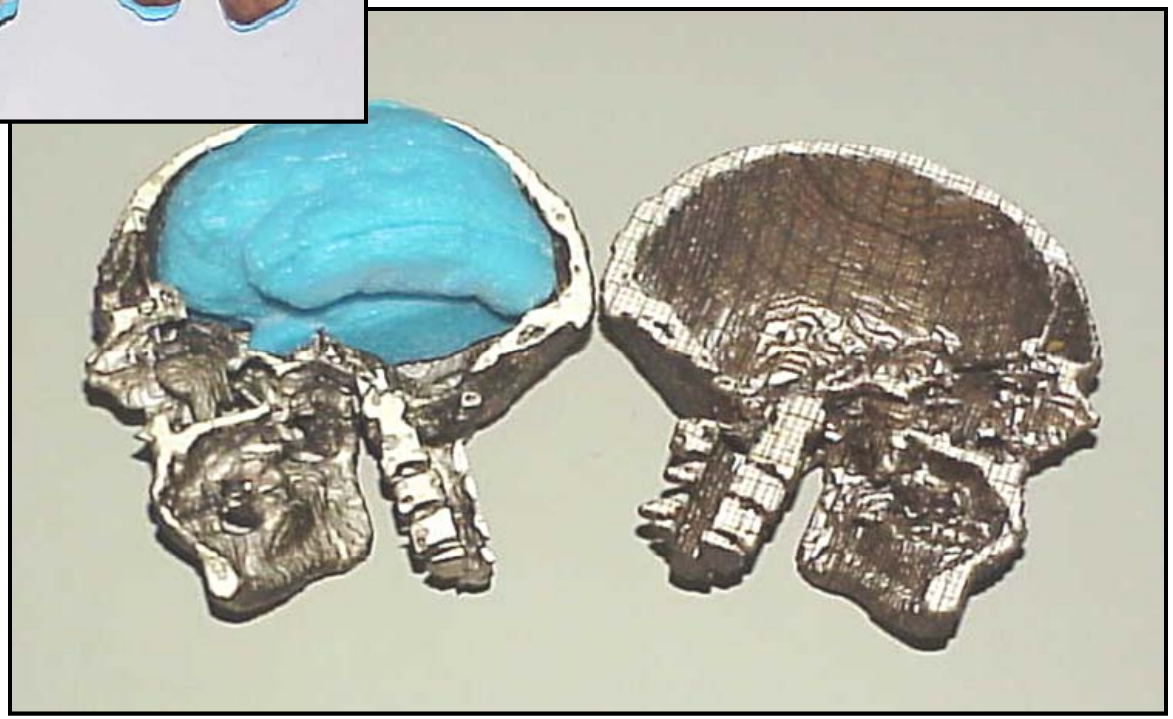


Professor Metoyer's Knee

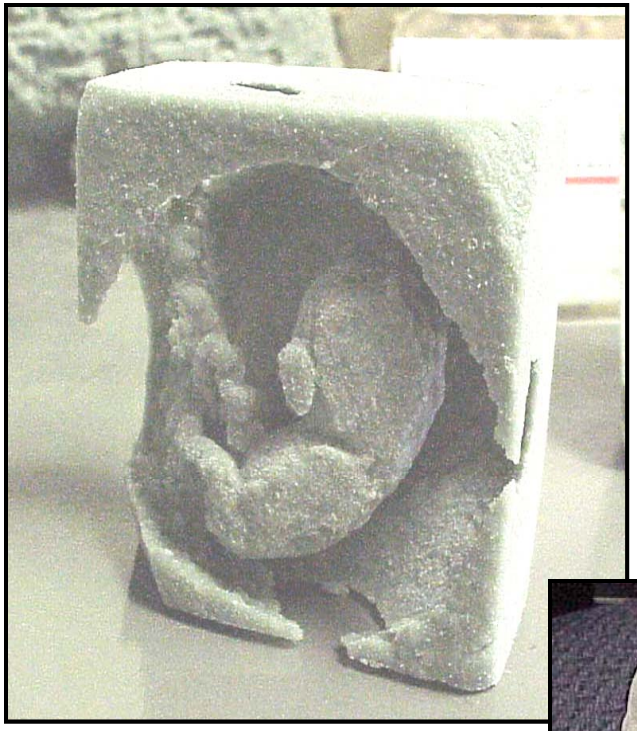
Foliage Density



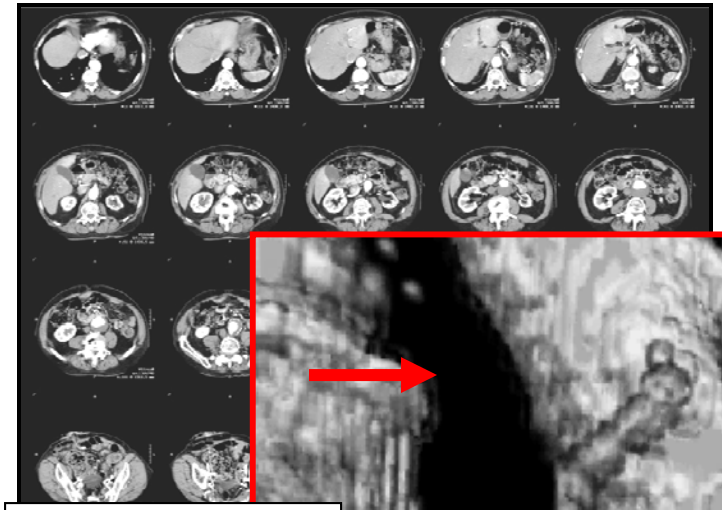
Isovolumes



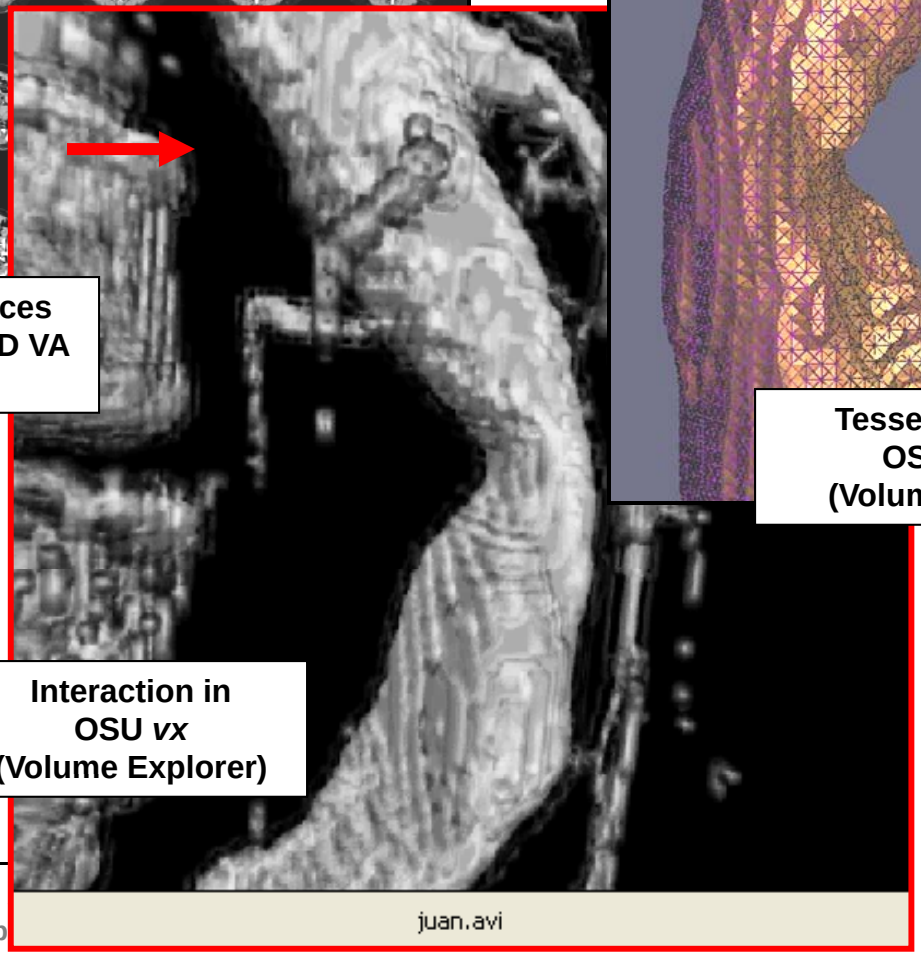
Isovolumes



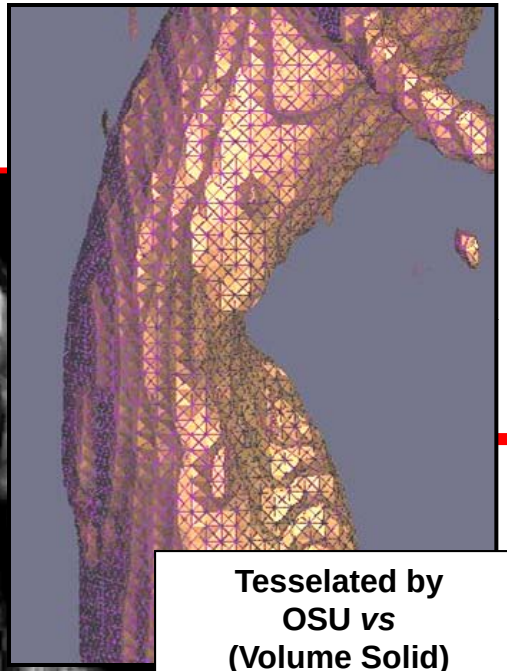
Putting the Tools Together:
Modeling and Making Anabolic Aortic Aneurysms



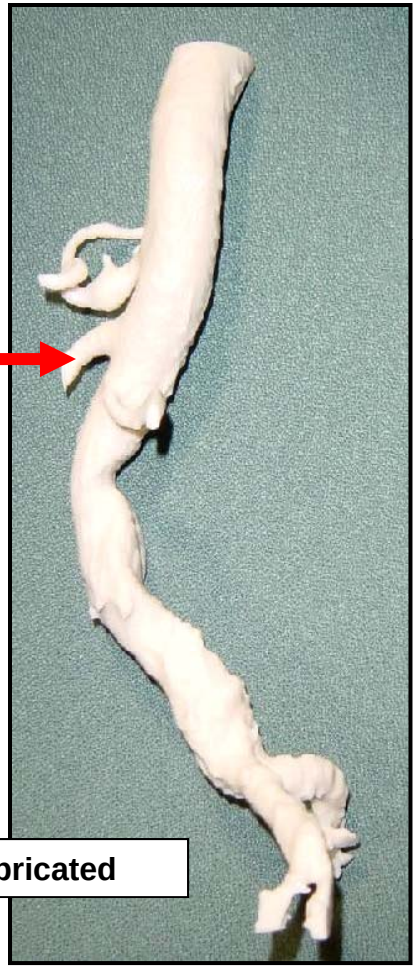
CAT scan slices from the UCSD VA Hospital



Interaction in OSU *vx* (Volume Explorer)



Tesselated by OSU *vs* (Volume Solid)



Fabricated

OSU

Oregon
Comp