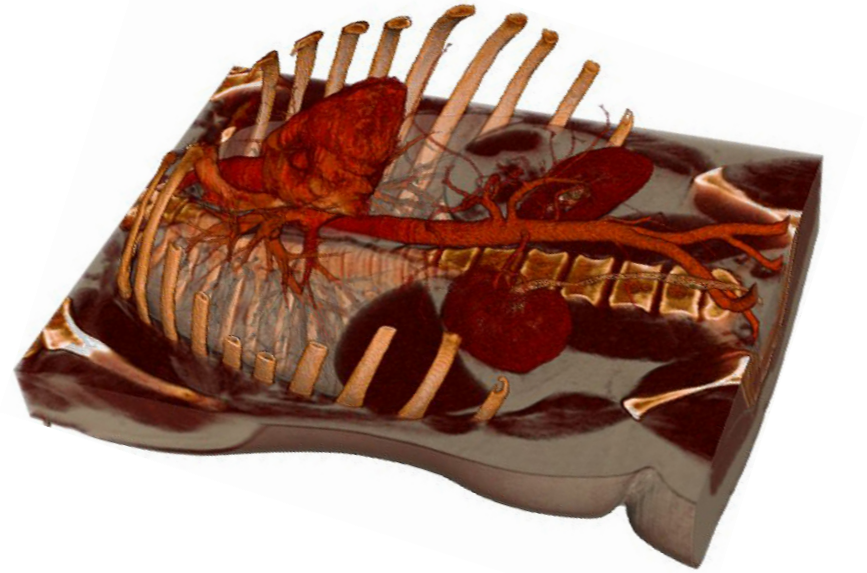
FACULTY OF ENGINEERING AND ARCHITECTURE

E016712: Computer Graphics

Medical Visualization Part 1

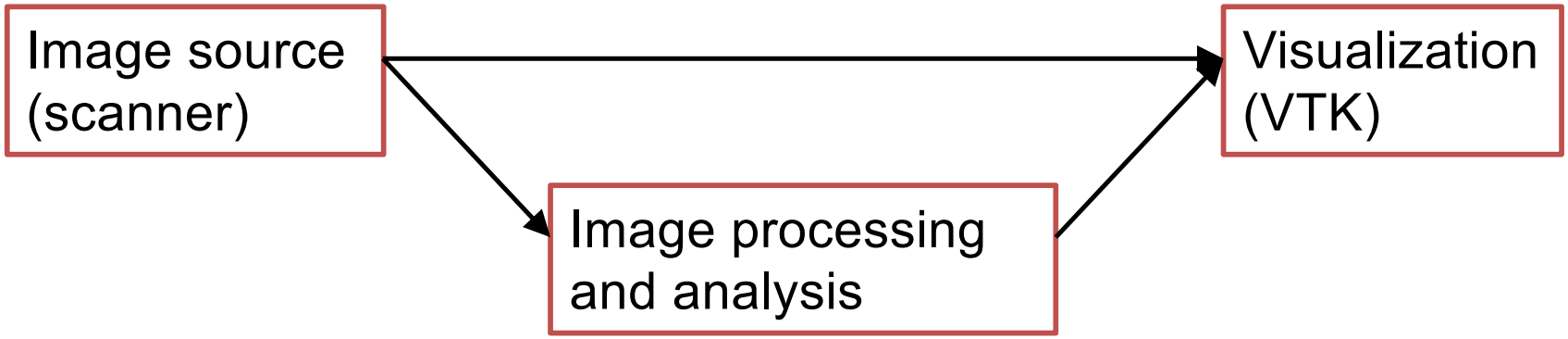


Lecturers: Aleksandra Pizurica and Danilo Babin

GHENT UNIVERSITY

Medical image visualization and analysis

- 2 paths leading to visualization

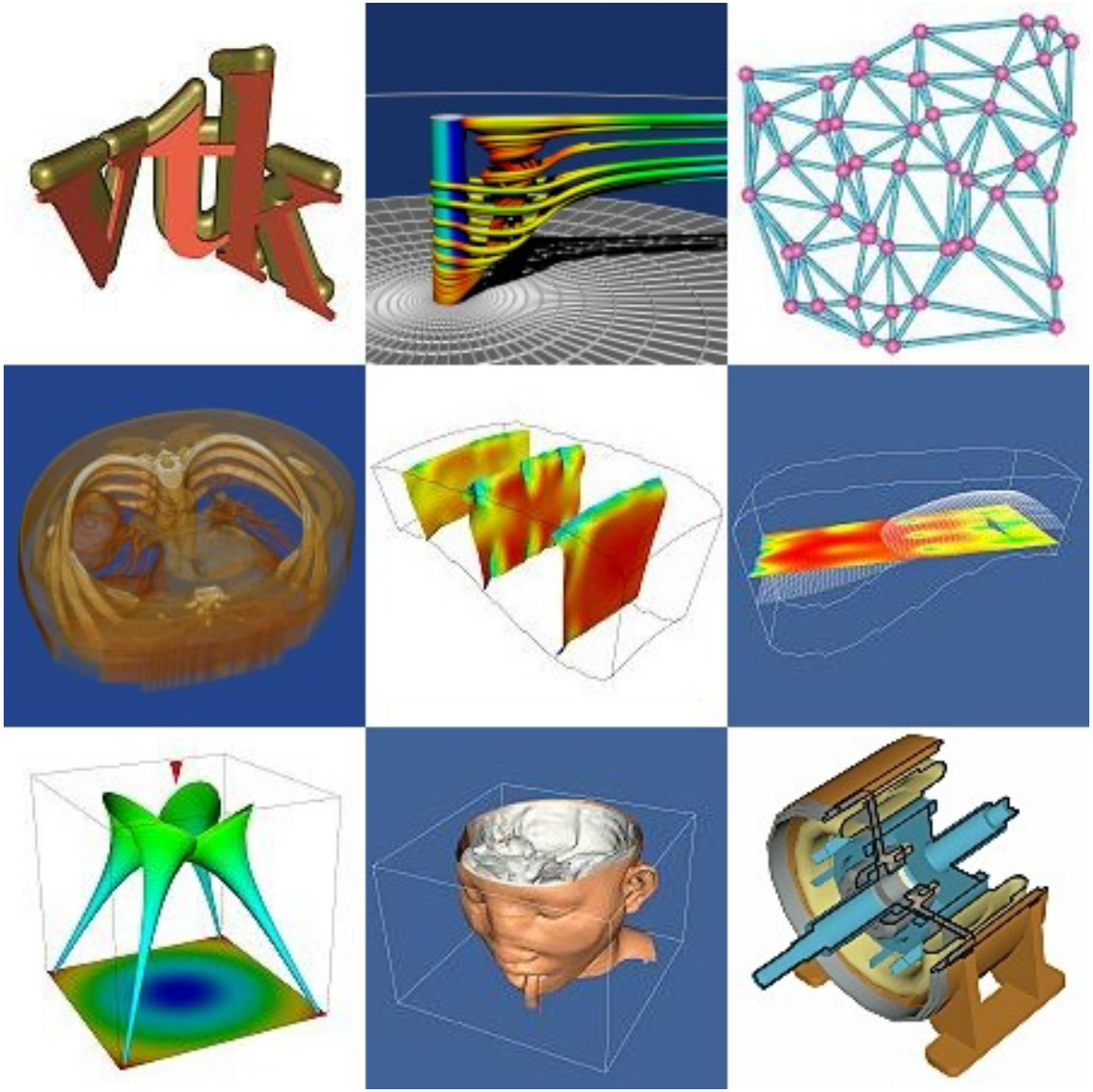


Outline

- Presentation outline:
  - Visualization Toolkit (VTK)
  - Imaging modalities & visualization

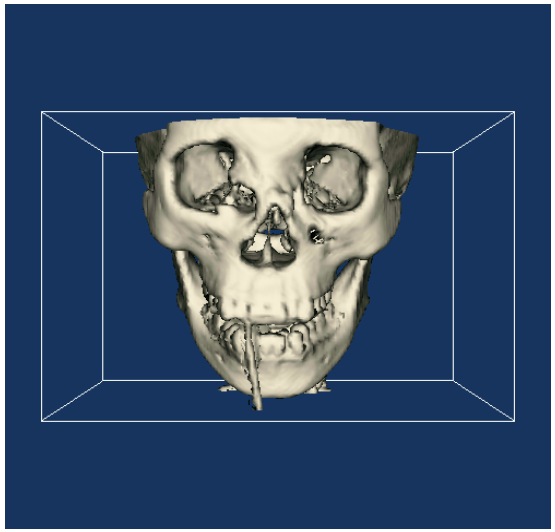
Visualization Toolkit (VTK)

- Open source (BSD)
- 3D visualization
- Image processing
- Object Oriented design
- C++, Tcl, Python, Java.
- VTK + Qt

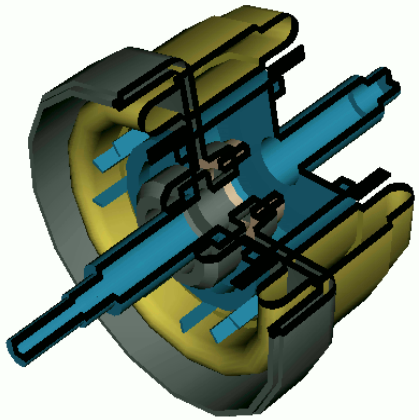


Visualization Toolkit (VTK)

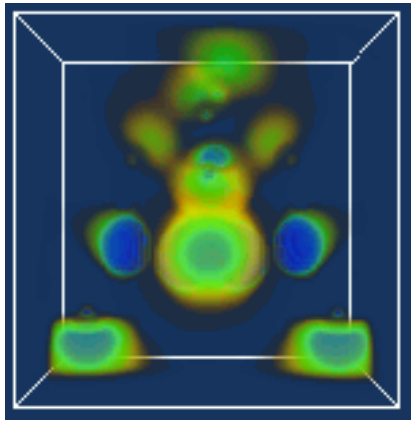
Medical Visualization

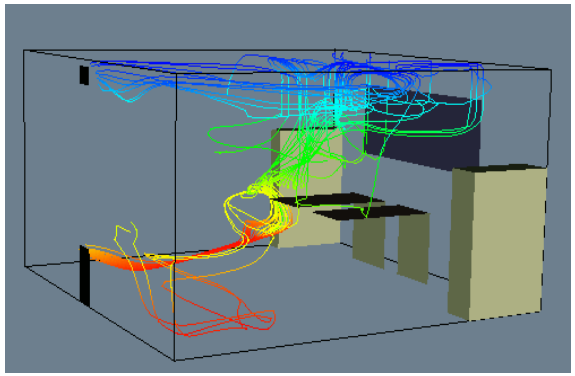


Modeling

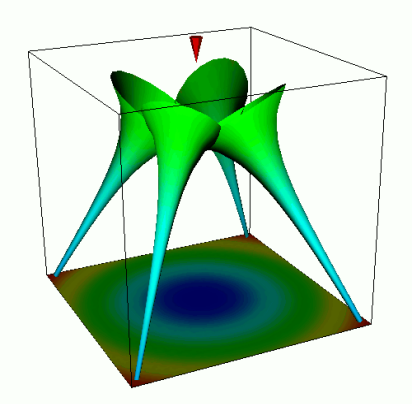


Volume Rendering





Flow Visualization



Tensor Visualization

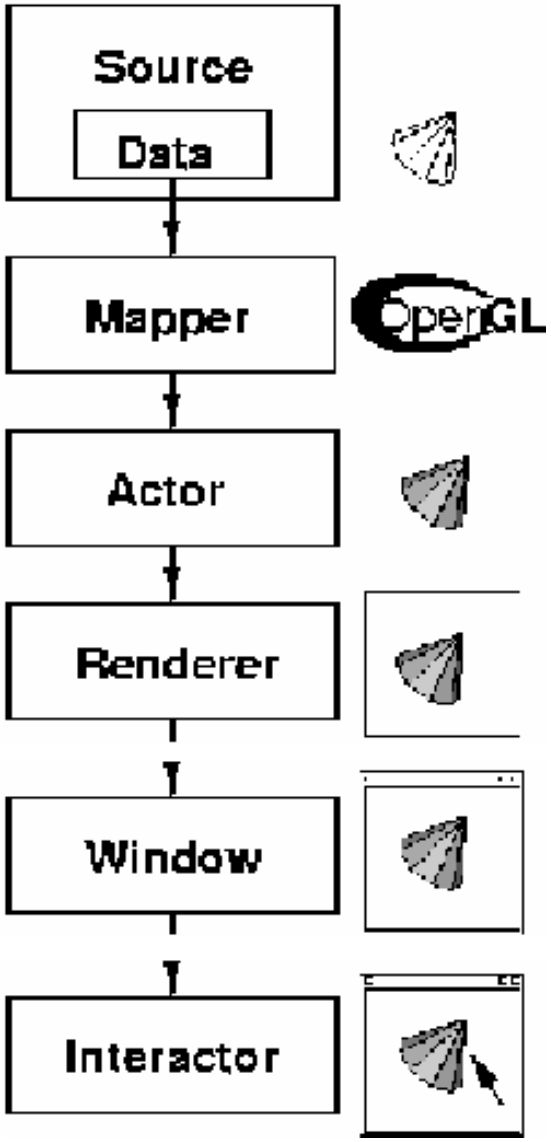
VTK in medical visualization

- Siemens syngo.via Frontier uses Fraunhofer MeVisLab as plugin basis (VTK supported).

- OsiriX DICOM Viewer (Mac only) – extremely popular with radiologists.

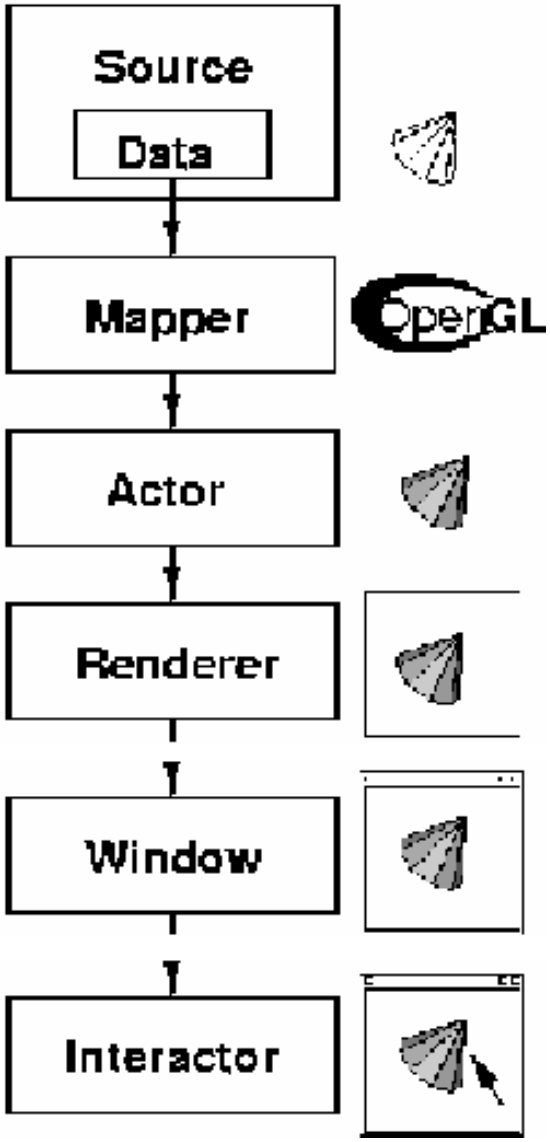
- Horos (free open source branch of Osirix)
- 3D Slicer
- ParaView
- BioImage Suite
- …

VTK: pipeline



- Reads data from a file or creates the data from scratch.
- Moves the data from VTK to OpenGL.
- Object properties (color, position…).
- Projects 3D scene to the screen.
- The window with title bar.
- Enables mouse and keyboard user interaction.

VTK: pipeline code




```
vtkSmartPointer< vtkSphereSource > sphere =
vtkSmartPointer<vtkSphereSource>::New();

vtkSmartPointer< vtkPolyDataMapper > mapper =
vtkSmartPointer<vtkPolyDataMapper> ::New();
mapper->SetInputConnection(sphere->GetOutputPort());

vtkSmartPointer< vtkActor > actor = vtkSmartPointer< vtkActor >::New();
actor->SetMapper(mapper);

vtkSmartPointer< vtkRenderer > renderer = vtkSmartPointer< vtkRenderer > ::New();
renderer->AddActor(actor);

vtkSmartPointer< vtkRenderWindow > renWin = vtkSmartPointer<
vtkRenderWindow>::New();
renWin->AddRenderer(renderer);

vtkSmartPointer< vtkRenderWindowInteractor > iren =
                vtkSmartPointer<vtkRenderWindowInteractor >::New();
Iren->SetRenderWindow(renWin);

renWin->Render();
Iren->Start();
```

Imaging modalities & visualization

- Imaging modalities (selected ones):
  - X-ray based: CT, DSA, 3DRA
  - Magnetic Resonance Imaging (MRI)
  - Ultrasound, IVUS, OCT
  - Hyperspectral

- Visualization
  - Multiplane reconstruction
  - Surface rendering (marching cubes)
  - Volume rendering
    - Texture mapping
    - Maximum intensity projection
    - Ray casting

X-ray angiography

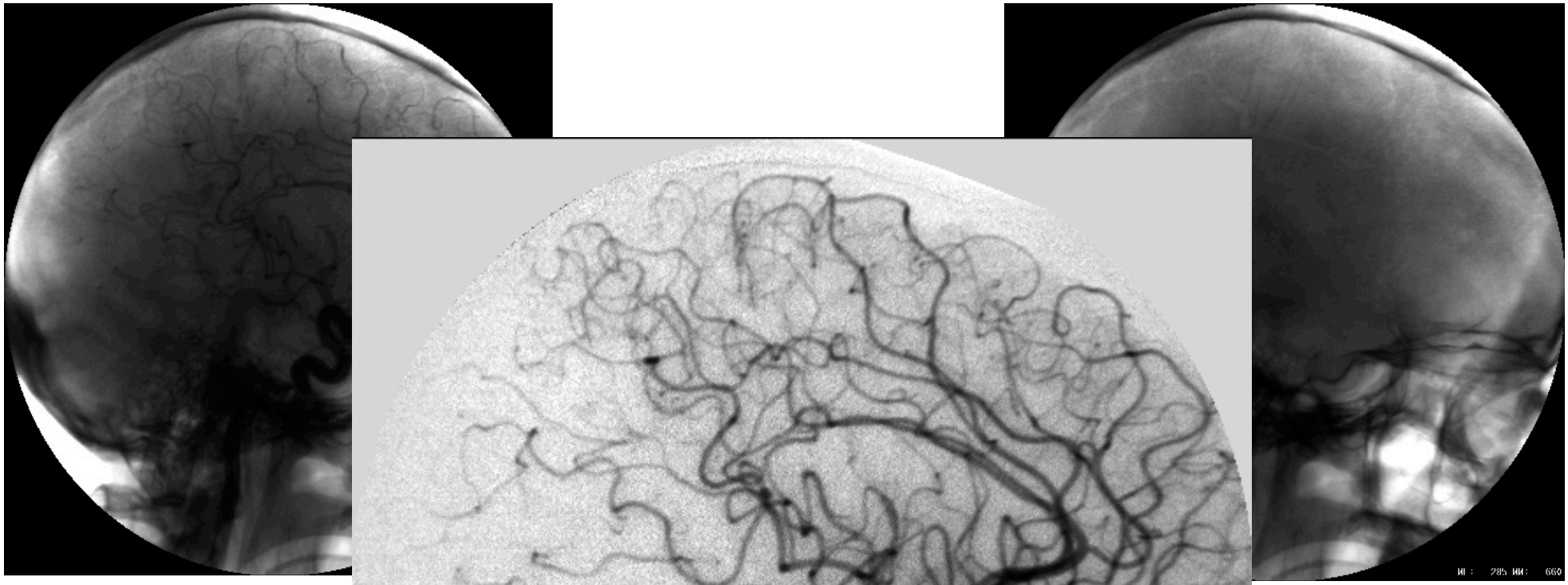
- X-rays developed by Röntgen in 1895.

- The way to visualize vessels: inject contrast agent that is well visible on x-ray (angiography).

- Problem: bones (skull, chest) occlude visibility of important blood vessels.

- Solution: subtraction of contrasted injected and non-contrast injected images (digital subtraction angiography).

Digital Subtraction Angiography (DSA)



Contrast-enhar

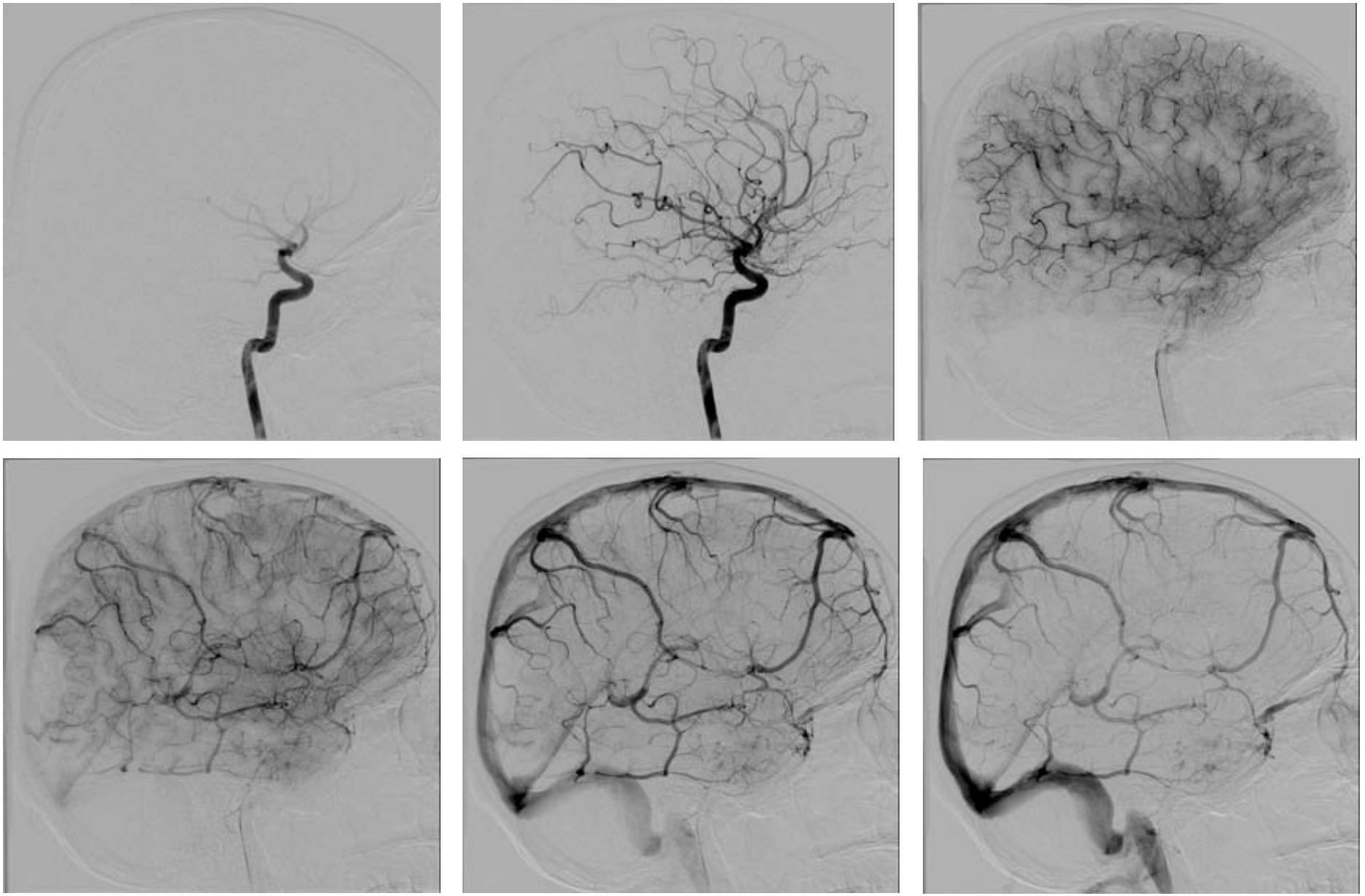
ray

895

=

DSA

DSA (2D + time): tracking contrast fluid propagation

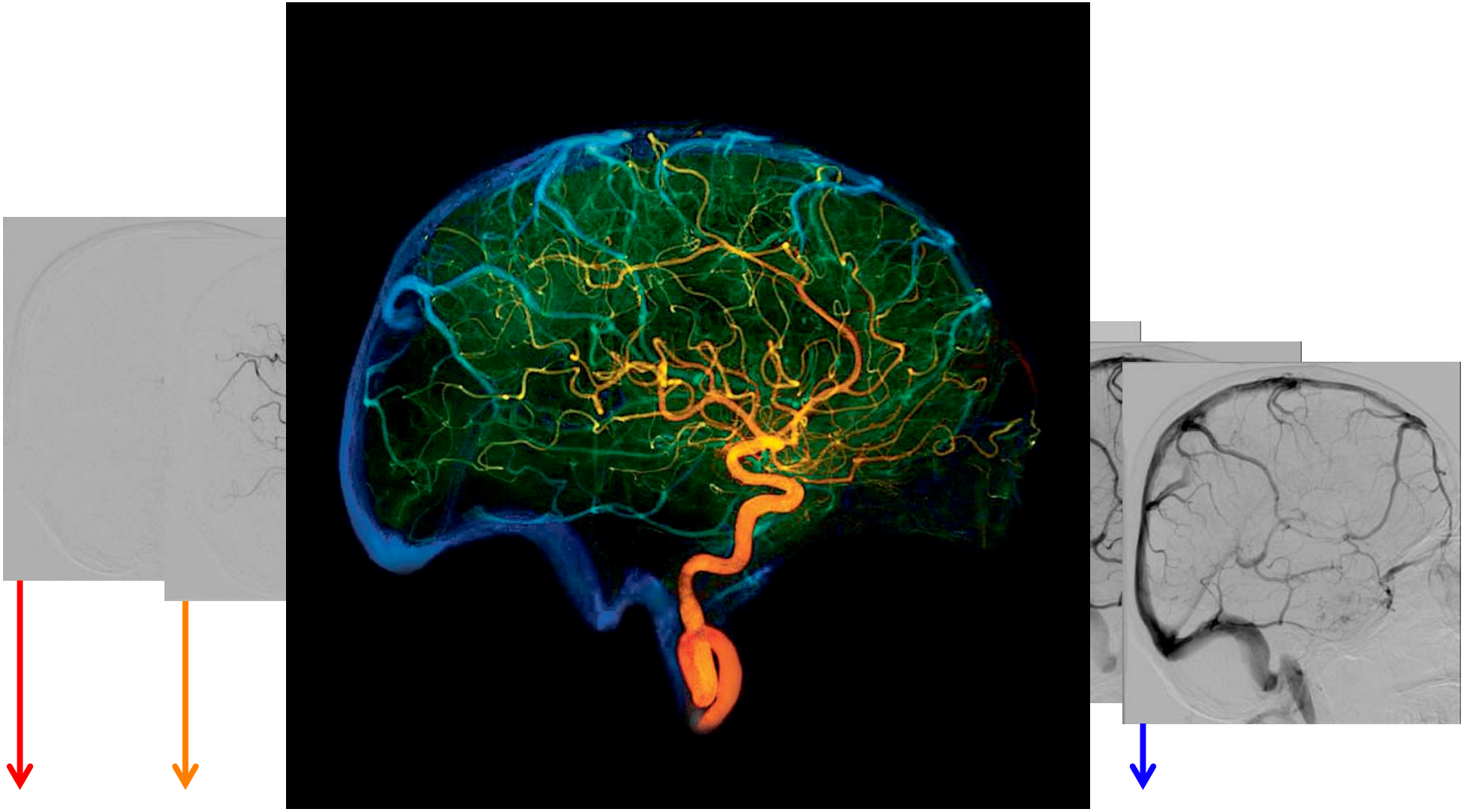


DSA (2D + time): tracking contrast fluid propagation

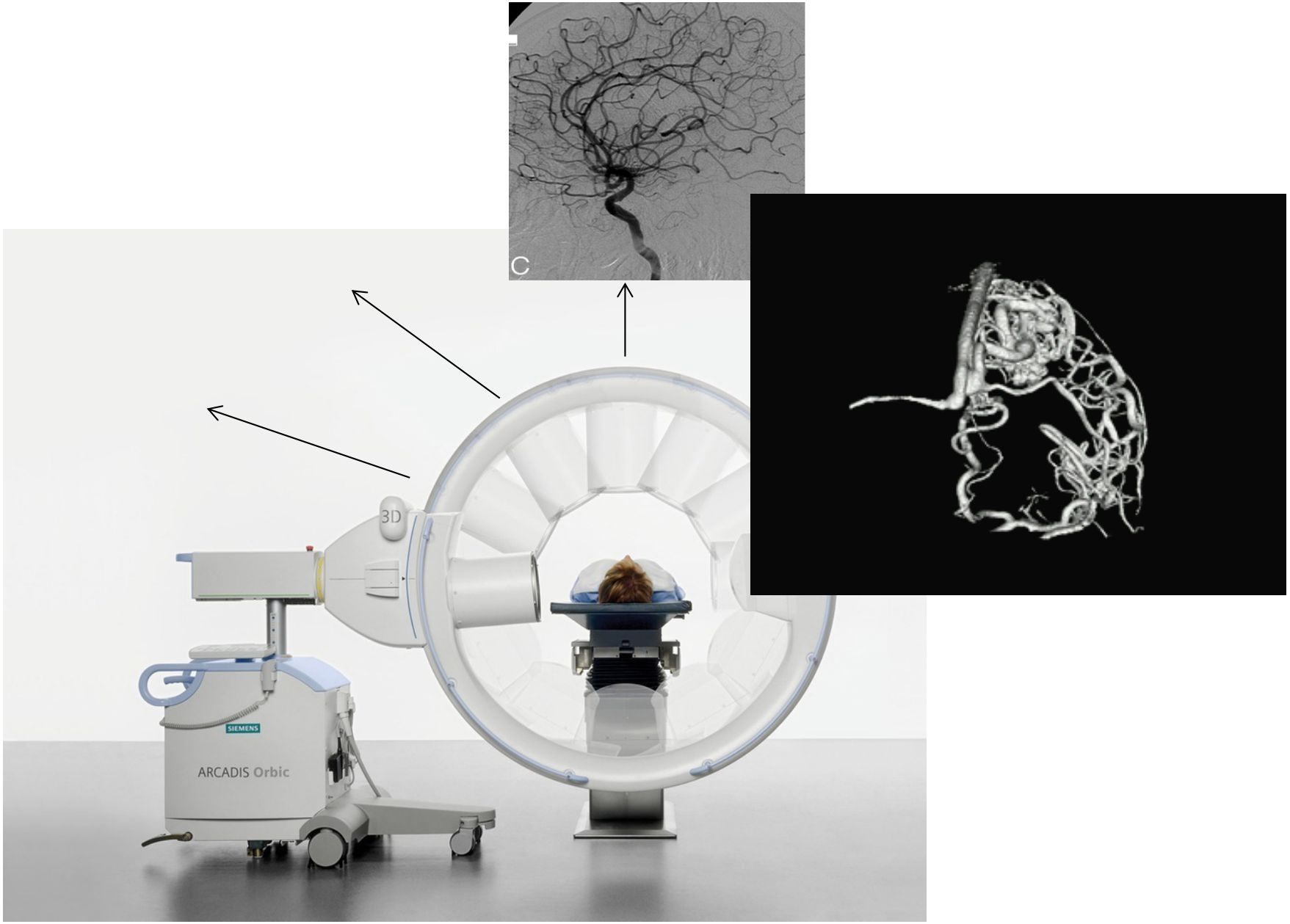
- Problem: as temporal resolution increases (more images), more time is needed for physicians to analyze images.

- Solution: represent the whole time series on a single color-coded image by giving each time frame a color code.

Color DSA (Siemens syngo iFlow)



3D Rotational Angiography (3DRA)

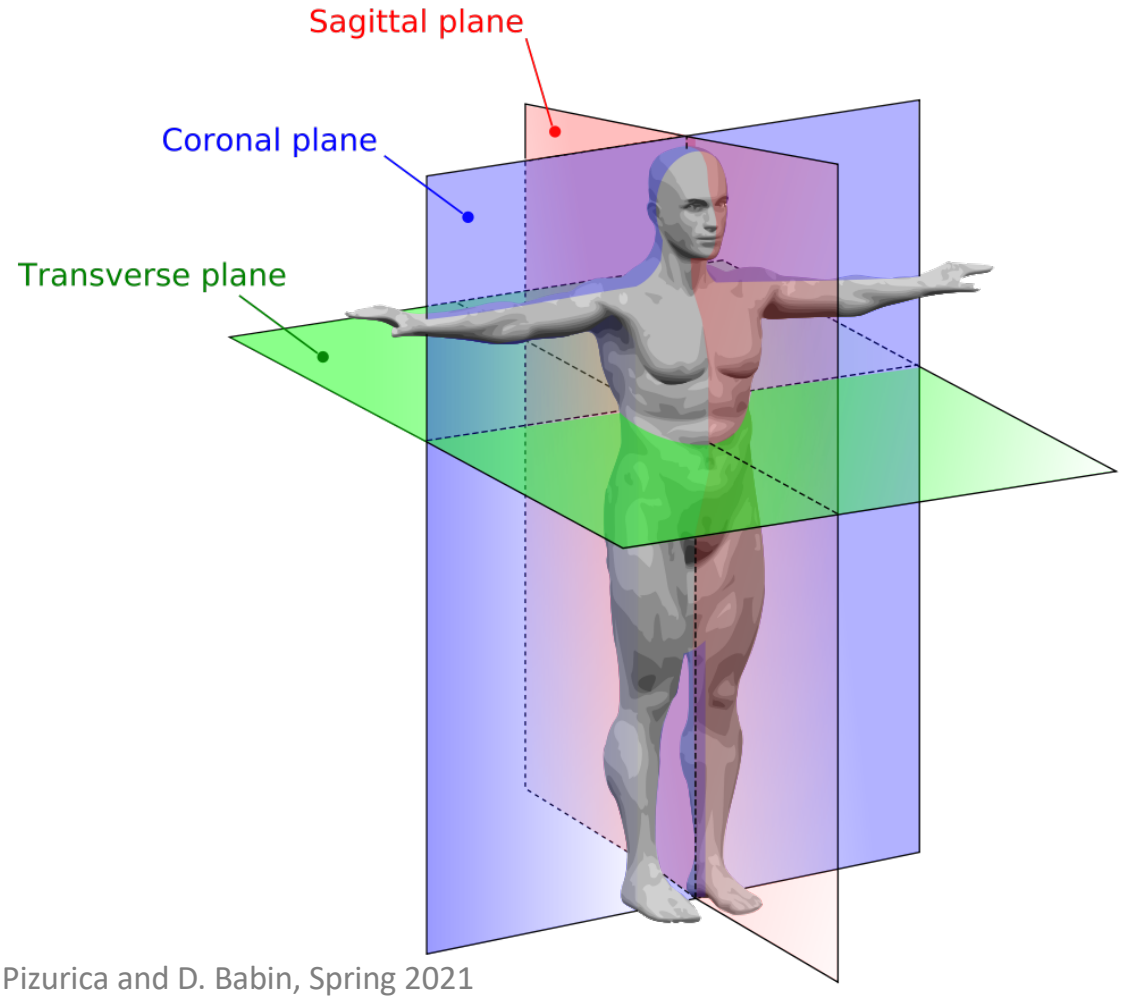


Visualization of 3D images

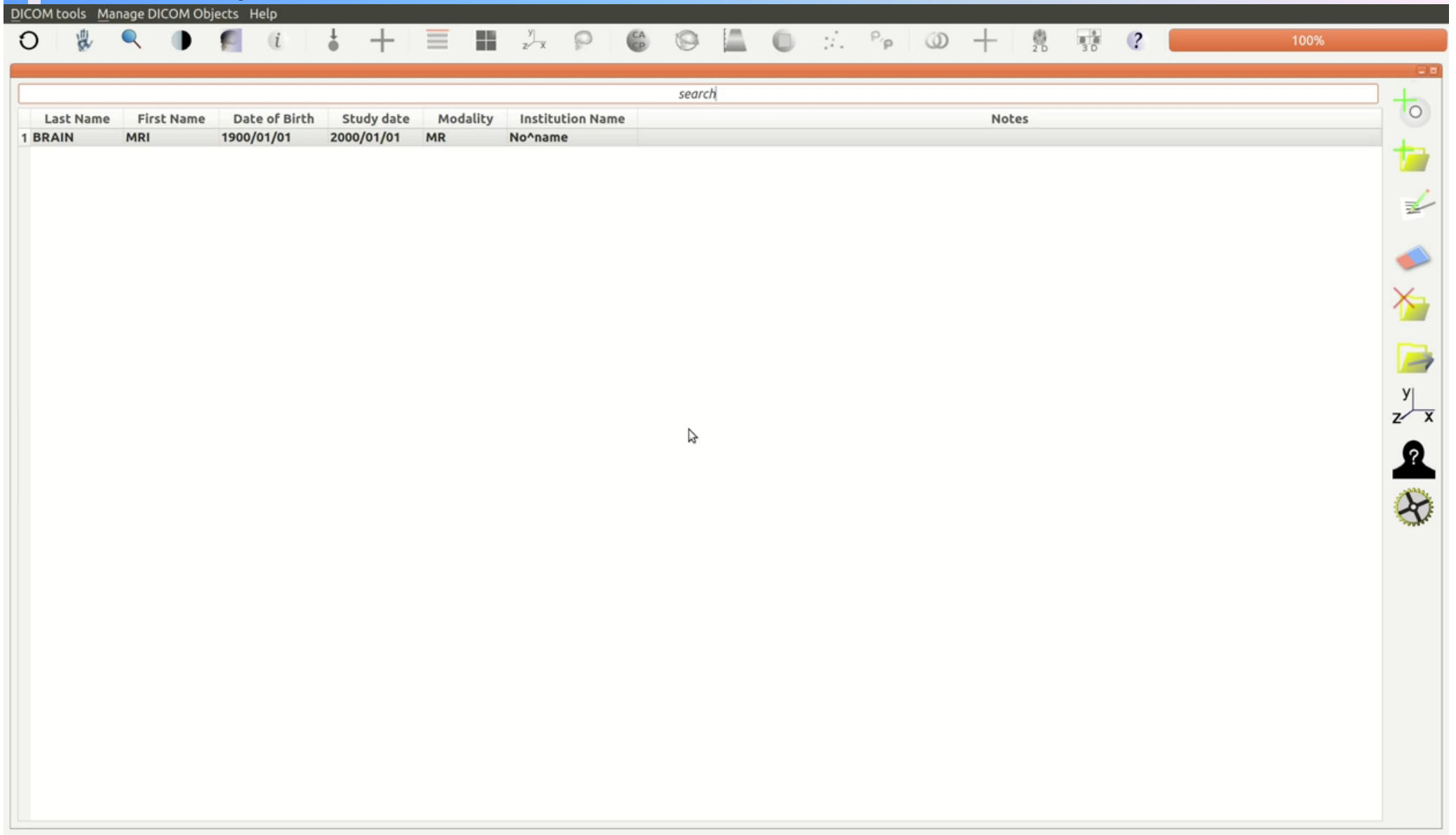
- Visualization
  - Multi-planar reconstruction
  - Surface rendering (marching cubes)
  - Volume rendering
    - Texture mapping
    - Maximum intensity projection
    - Ray casting

Multi-planar reconstruction

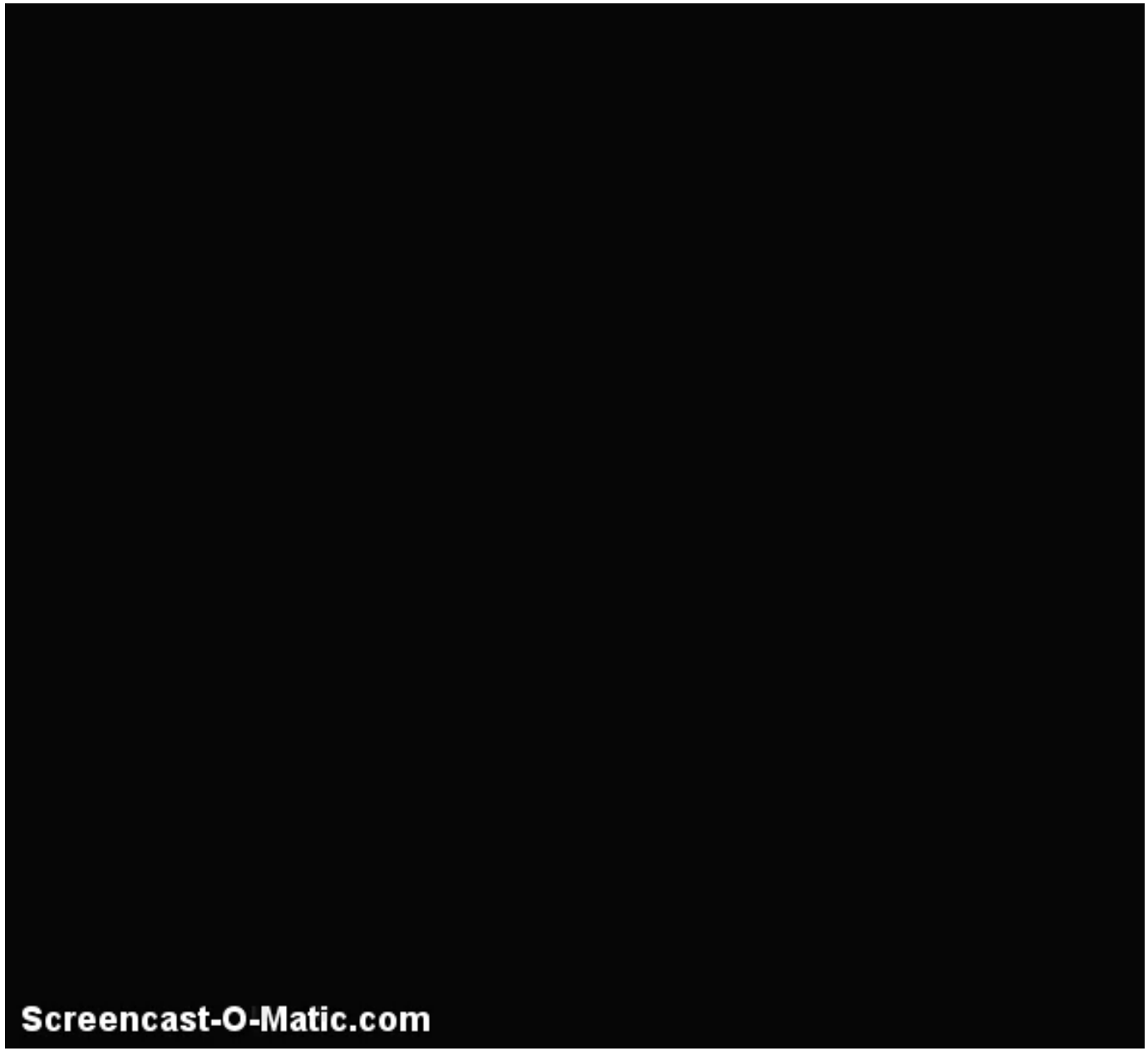
- Cutting through the volume of the 3D image with 2D planes
- Three plane directions: transversal, coronal and sagittal
- Oblique: slicing in any plane orientation



Multi-planar reconstruction



Multi-planar reconstruction in 3D



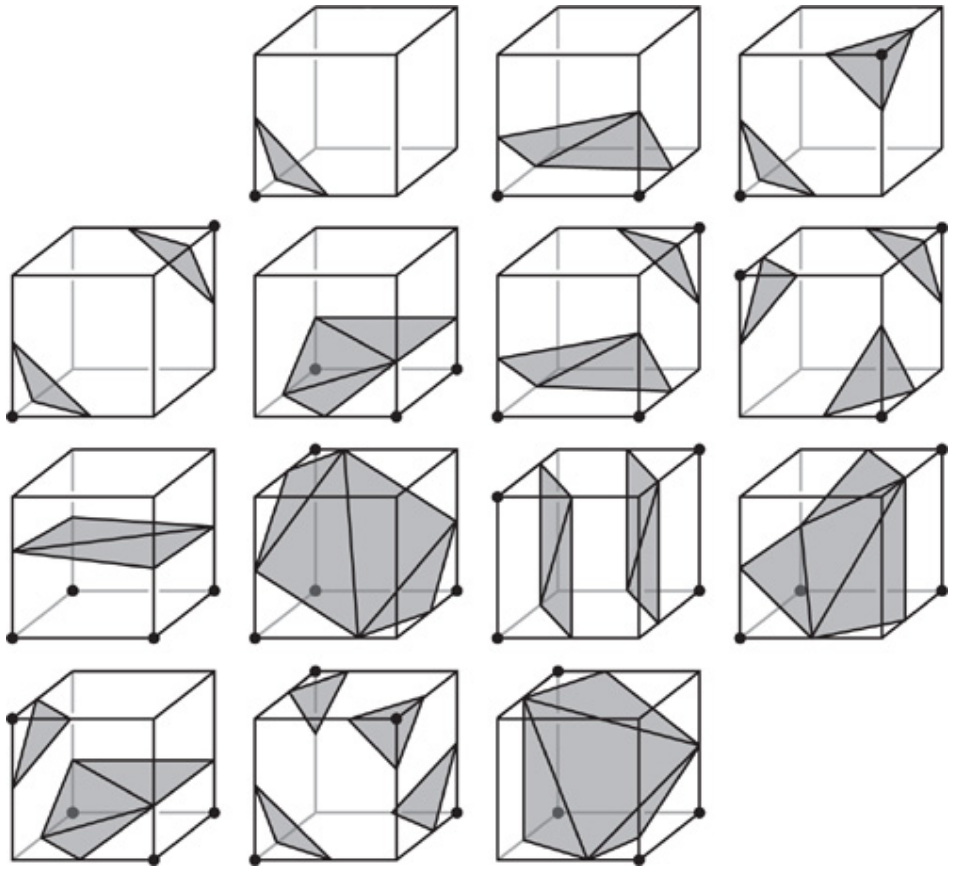
Multi-planar reconstruction: VTK code

- VTK has vtkImagePlaneWidget for this purpose

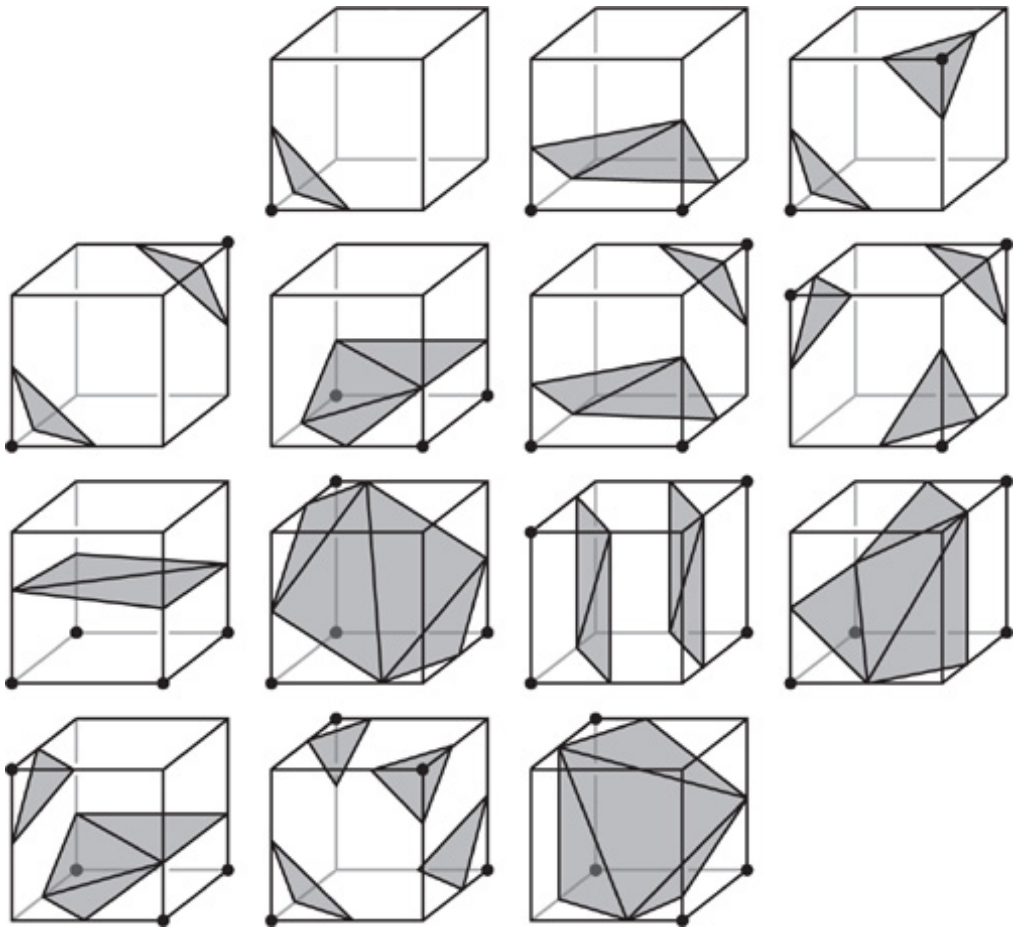
```
vtkSmartPointer<vtkImagePlaneWidget> transversal_plane =
        vtkSmartPointer<vtkImagePlaneWidget>::New();

transversal_plane->SetInteractor(interactor);

transversal_plane->SetInput(image);

transversal_plane->SetPlaneOrientationToZAxes();

transversal_plane->SetSliceIndex(0);
```

Surface reconstruction

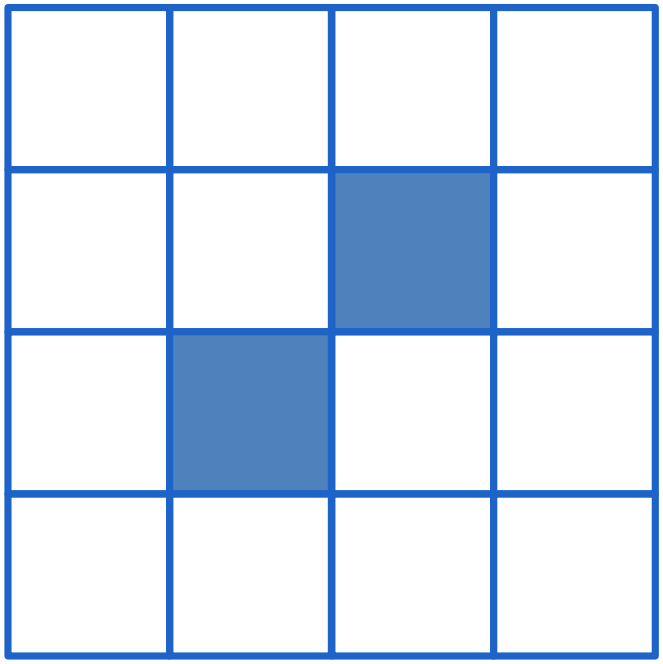
- Showing an iso-surface mesh from (segmented) image
- Basic meshing is performed using Marching Cubes algorithm



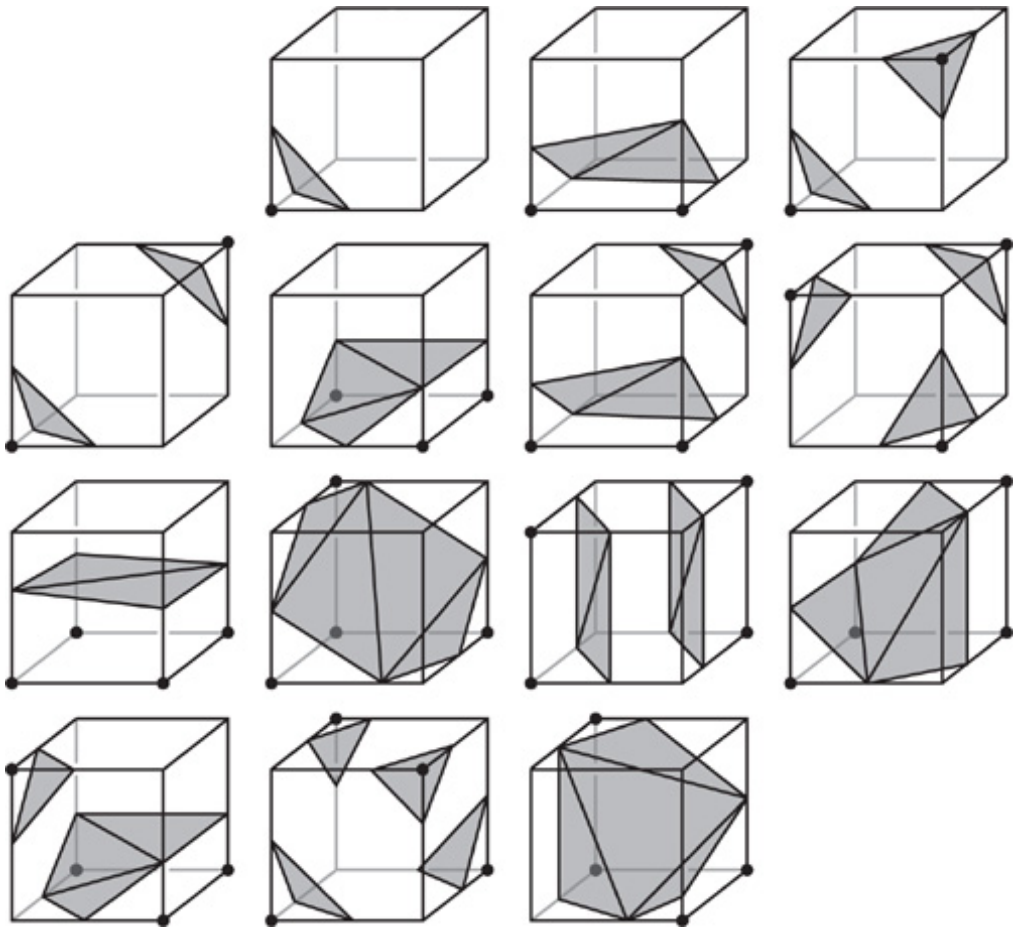
Surface reconstruction



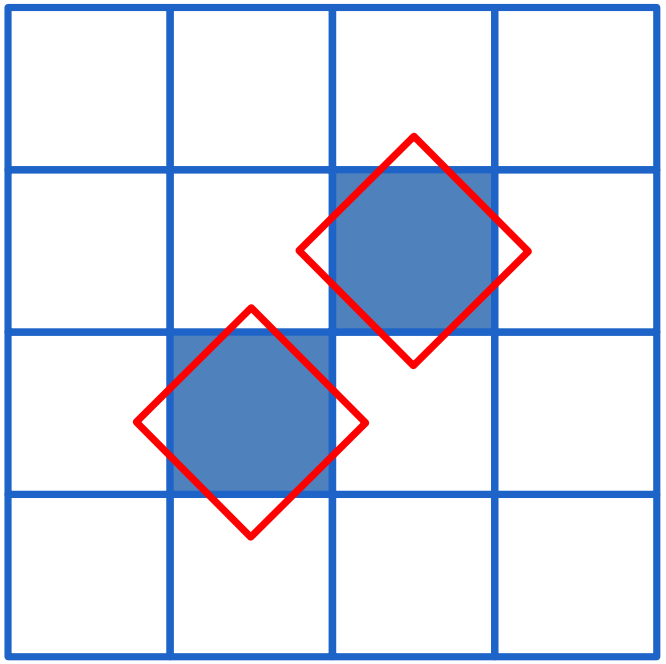
For the image below, what is the result of the marching cubes algorithm?



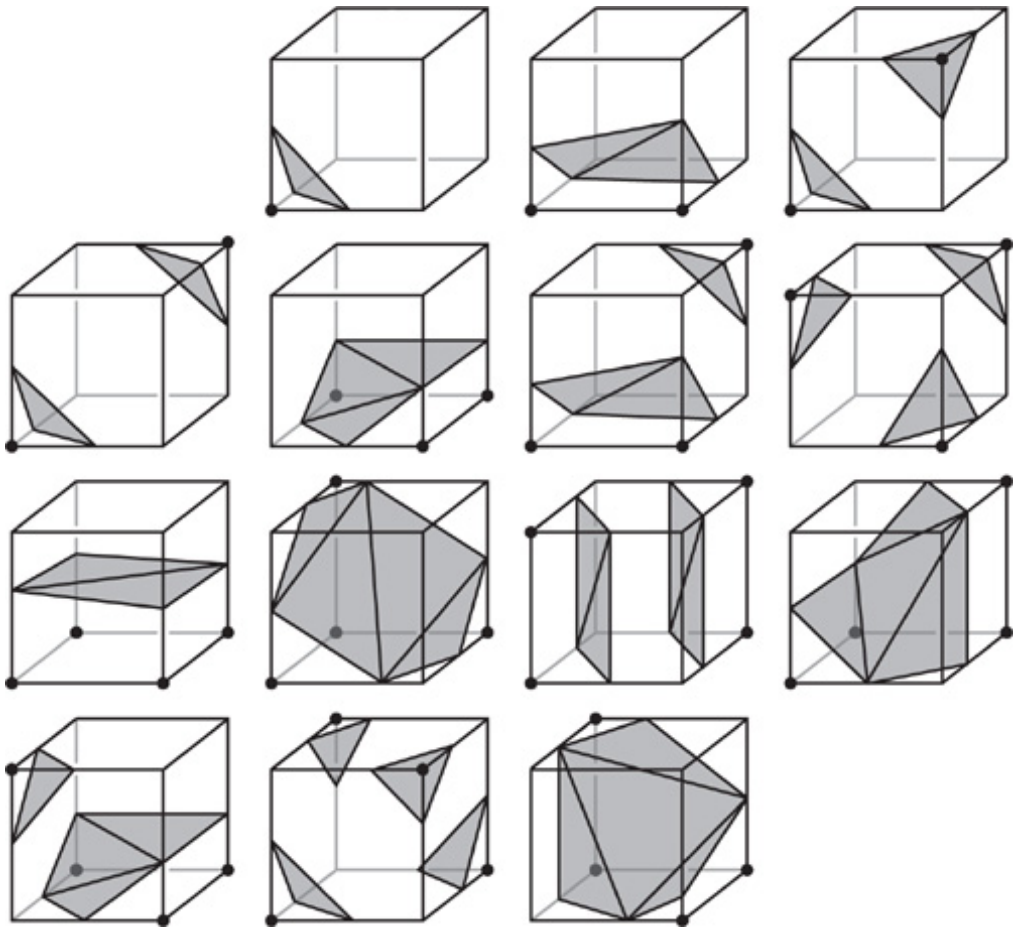
Surface reconstruction



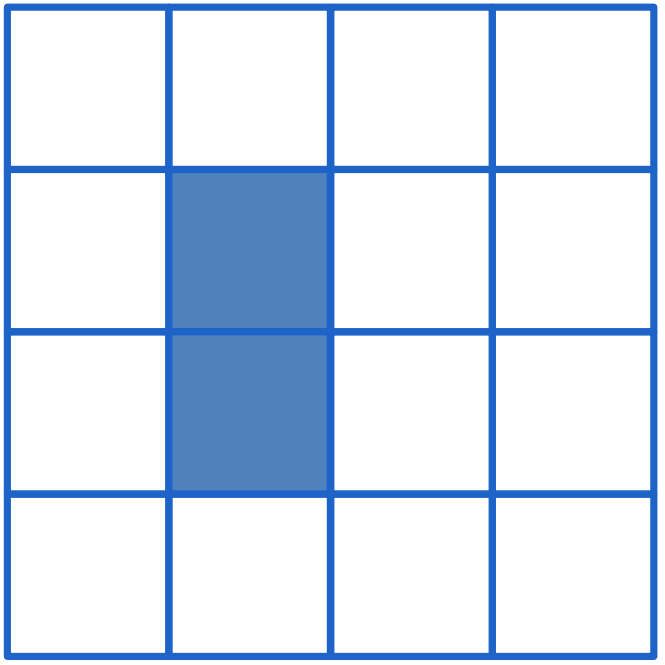
For the image below, what is the result of the marching cubes algorithm?



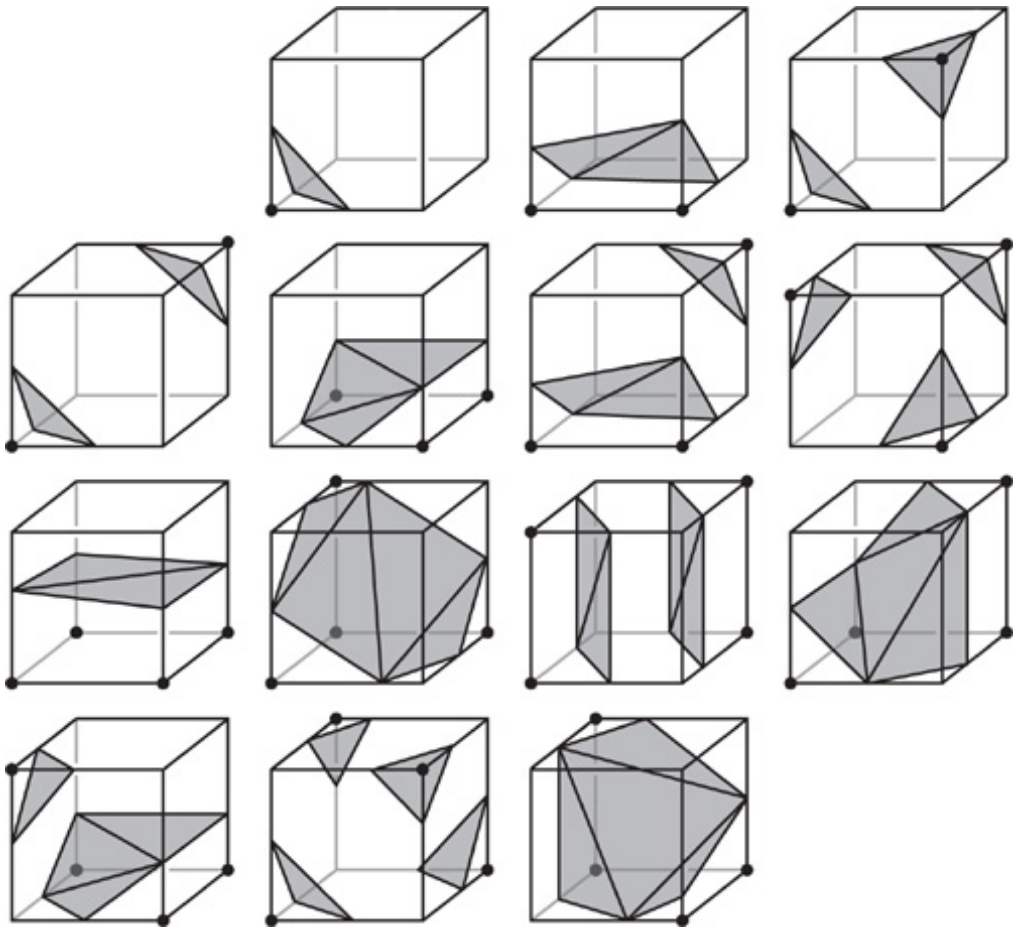
Surface reconstruction



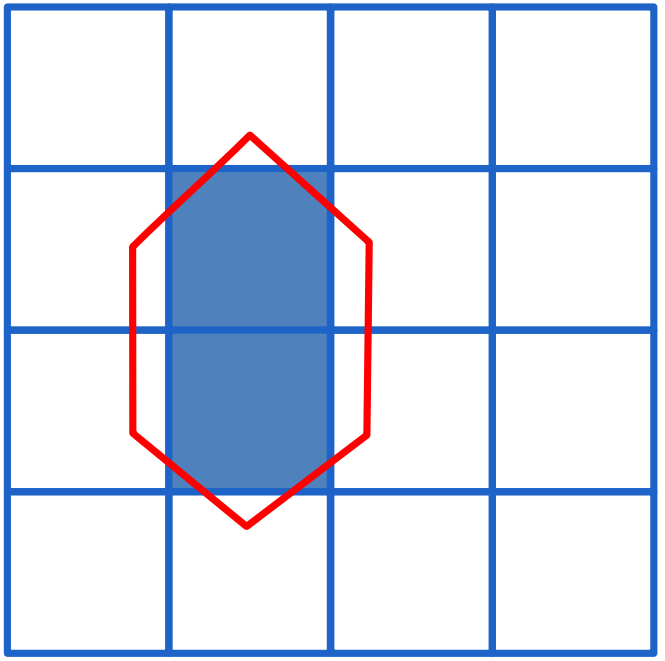
For the image below, what is the result of the marching cubes algorithm?



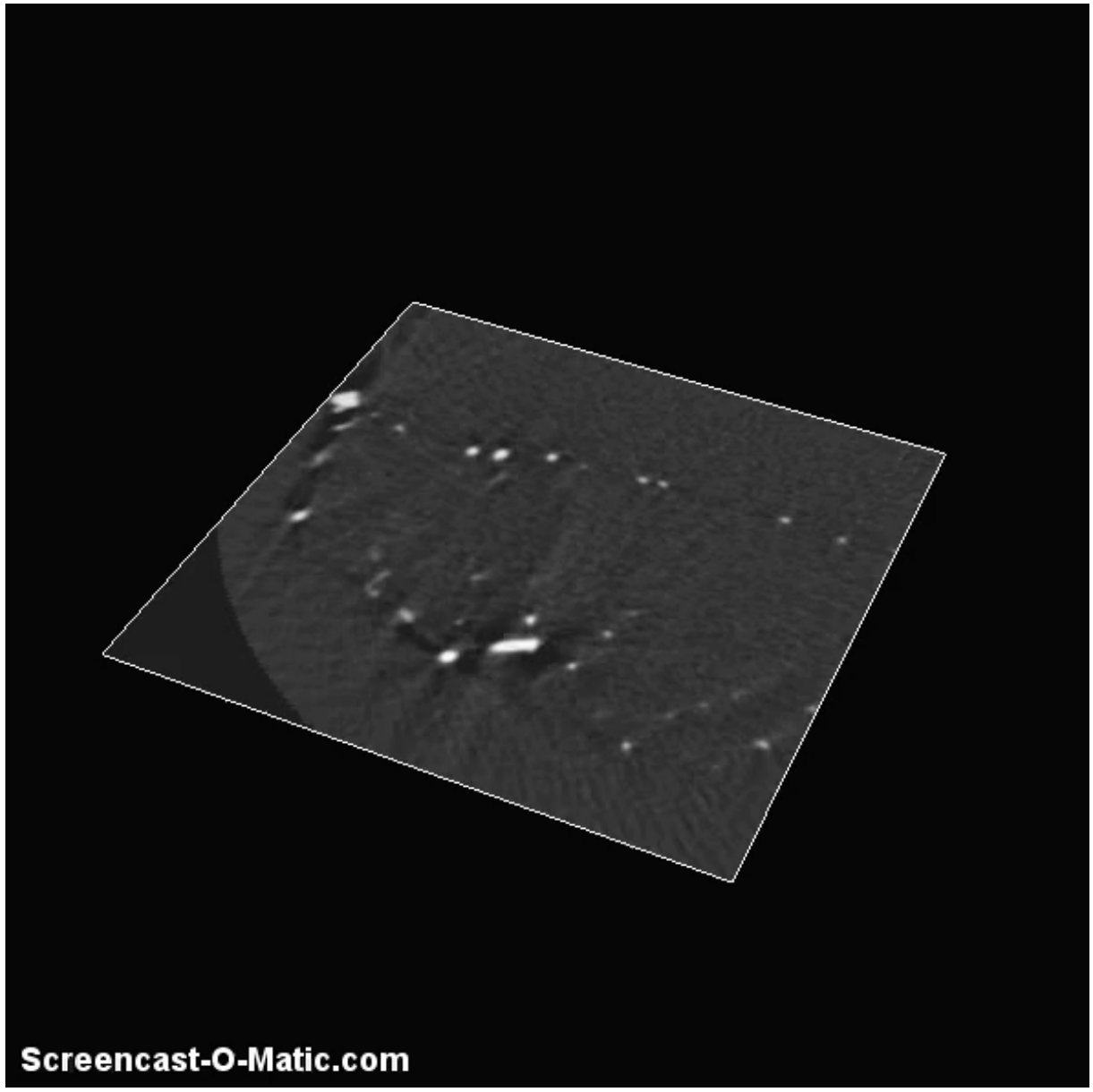
Surface reconstruction



For the image below, what is the result of the marching cubes algorithm?



Surface reconstruction: VTK Demo



Surface reconstruction: VTK Code

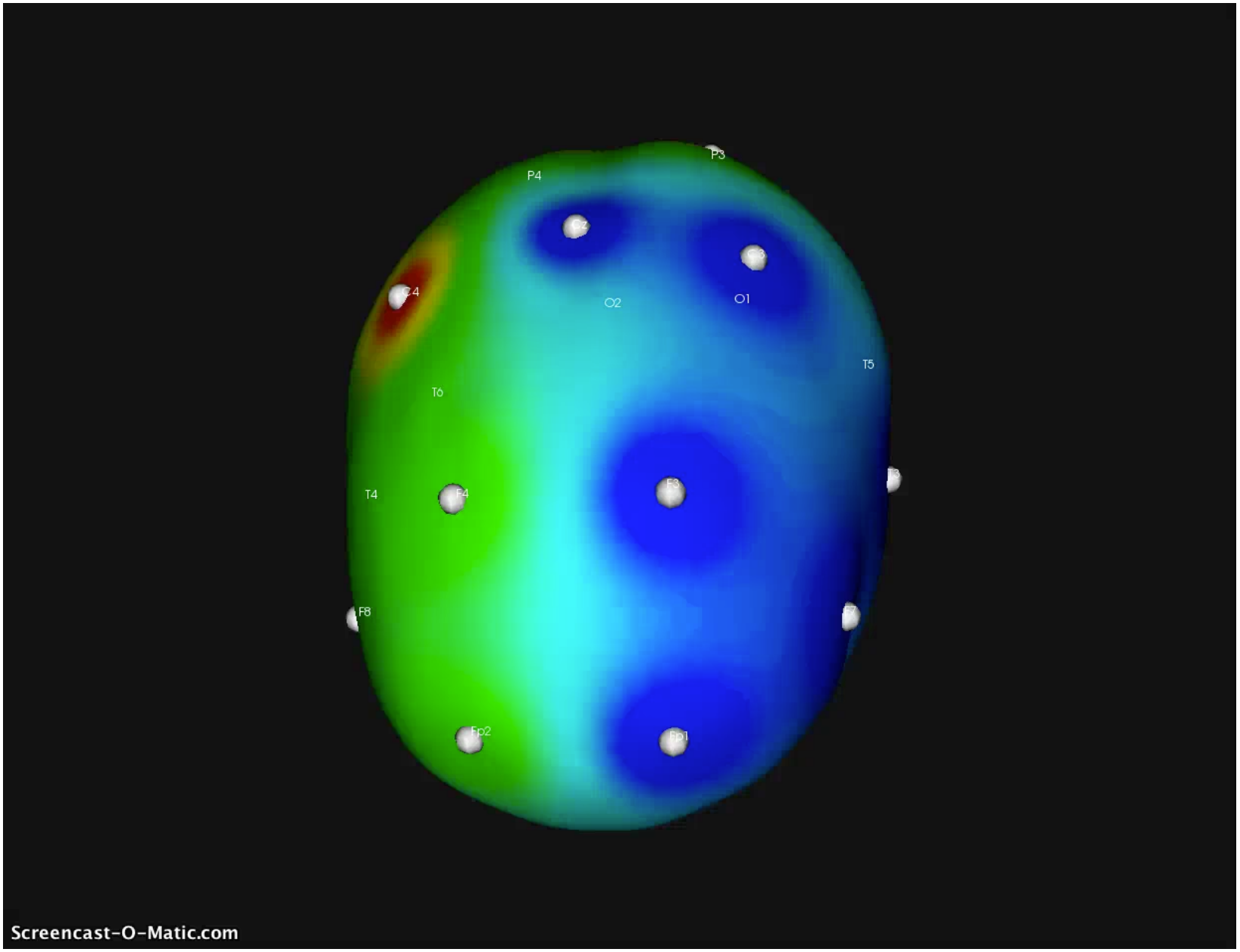
- VTK has vtkContourFilter for this purpose

```
vtkSmartPointer<vtkContourFilter> filter =
        vtkSmartPointer<vtkContourFilter>::New();

filter->SetInputData((vtkDataObject*)(image));

filter->SetValue(0,value);

filter->Update();

mapper->SetInputData(filter->GetOutput());
```

Mapping color values to 3D mesh

- Visualize scalars on mesh surface using a color map
- Application: Electroencephalography (EEG)

Mapping color values to 3D mesh

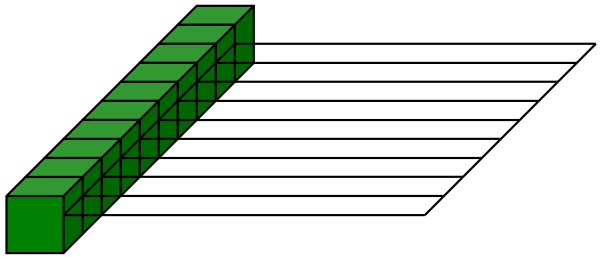


Mapping color to VTK poly data points

```
vtkSmartPointer<vtkPolyData> poly_data =
        vtkSmartPointer<vtkPolyData>::New();
vtkSmartPointer<vtkPolyDataMapper> mapper =
        vtkSmartPointer<vtkPolyDataMapper>::New();
vtkSmartPointer<vtkUnsignedCharArray> poly_data_point_scalars =
        vtkSmartPointer<vtkUnsignedCharArray>::New();

poly_data_point_scalars->SetNumberOfComponents(1);
for(int i=0; i<poly_data->GetNumberOfPoints(); i++){
double scalar_value = i;
poly_data_point_scalars->InsertNextTuple(&scalar_value);
}

poly_data->GetPointData()->SetScalars(poly_data_point_scalars);
mapper->SetInputData(poly_data);
mapper->SetColorModeToMapScalars();
mapper->SetScalarRange(0, 100);
```

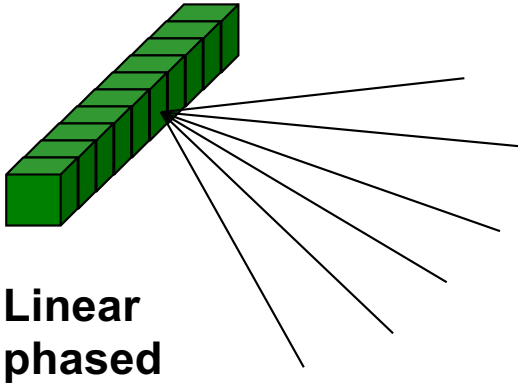
Ultrasound imaging

- Ultrasound is frequency over 20kHz
- Resolution increases with frequency (1.5MHz for 1mm)
- Penetration decreases with frequency



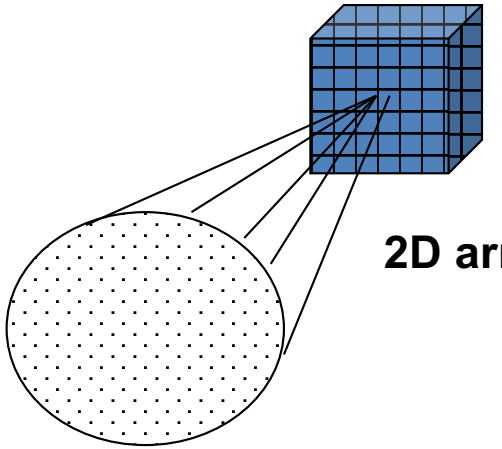
Linear

- High frequency: good resolution, shallow penetration
- Low frequency: low resolution, deep penetration



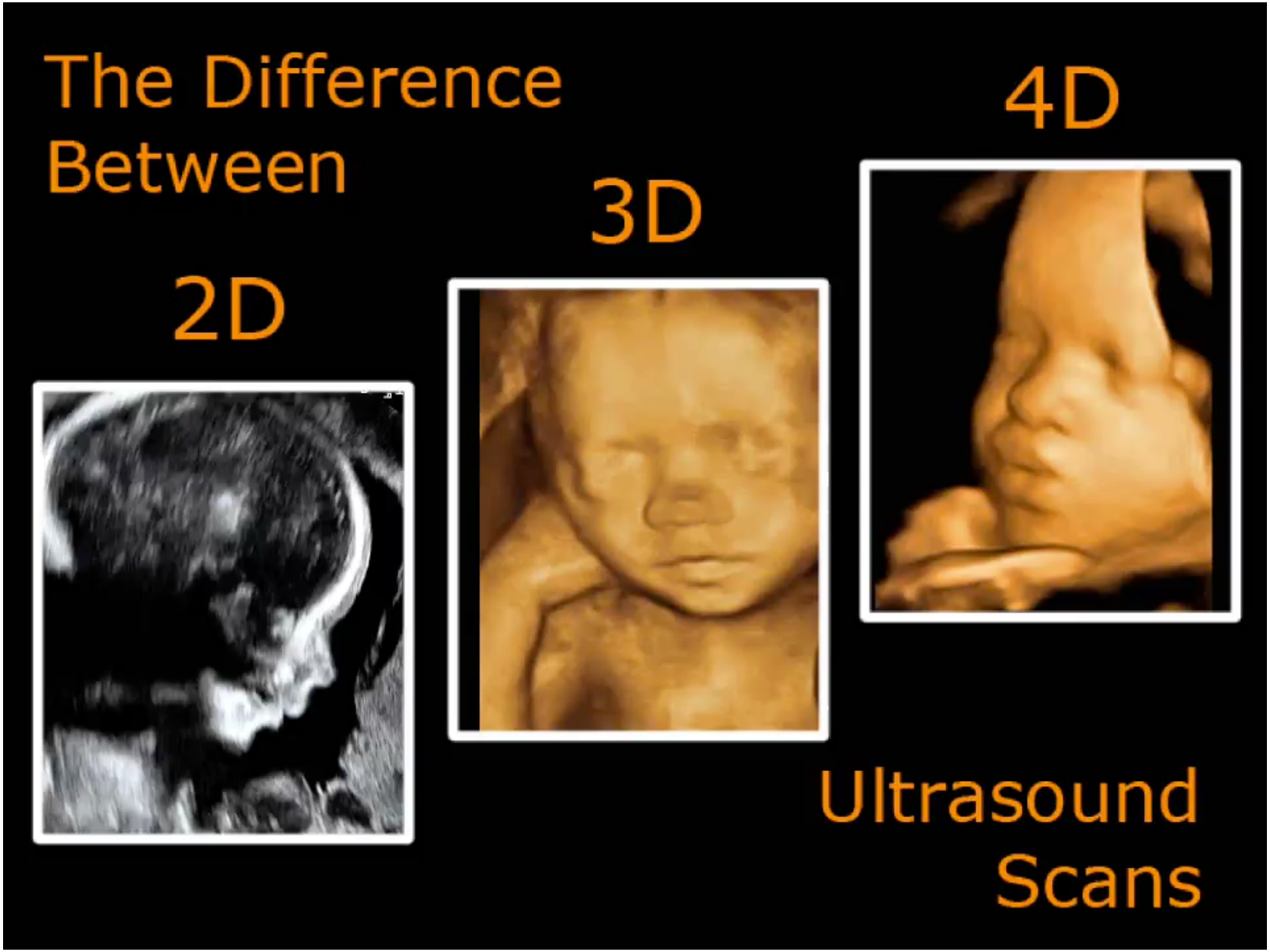
Linear phased

- Cardiology, gynaecology: 2-8 MHz
- Ophthalmology, peripheral vessels: 20MHz
- Intra-arterial: 20-50MHz



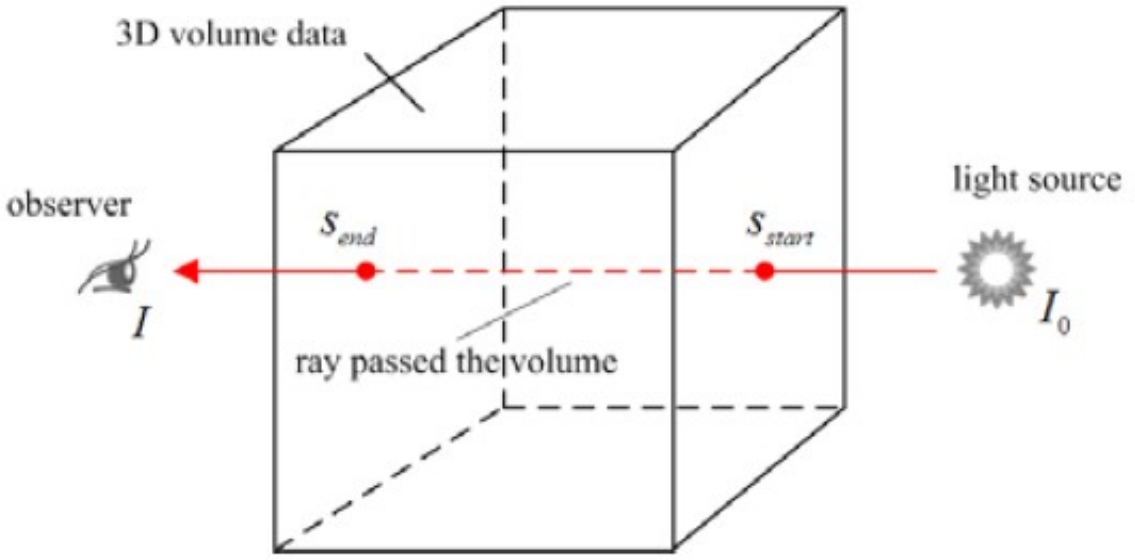
2D array

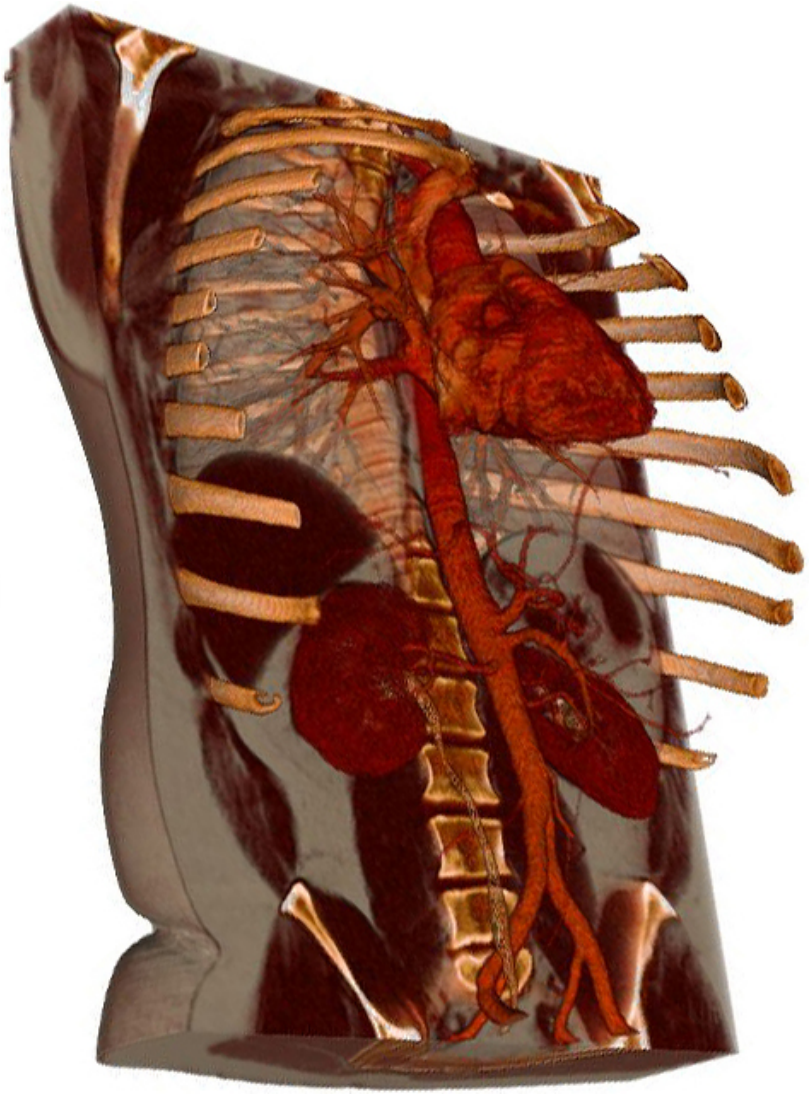
Ultrasound imaging



Volume rendering

- Ray casting
- Maximum Intensity Projection (MIP)
- Texture rendering
- Important: setting transfer function!





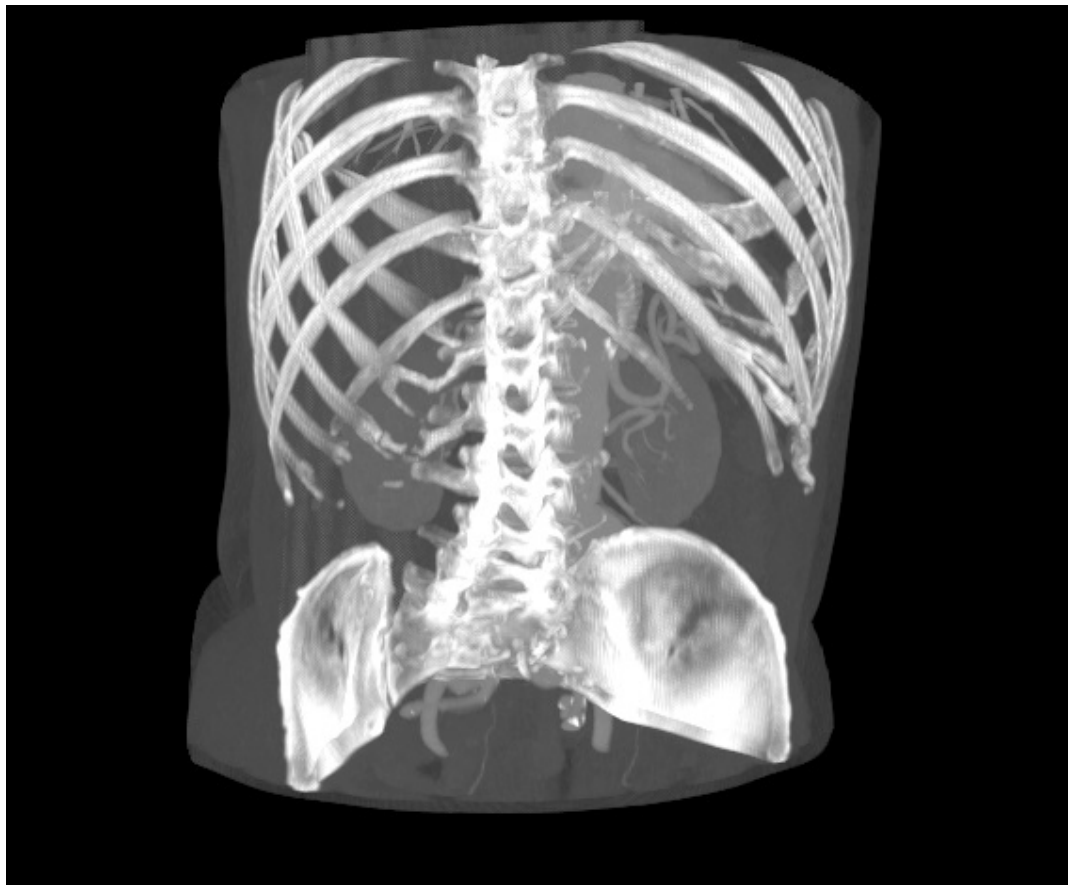
Volume rendering: Maximum intensity projection

- Display the maximum intensity of voxel along the ray path.

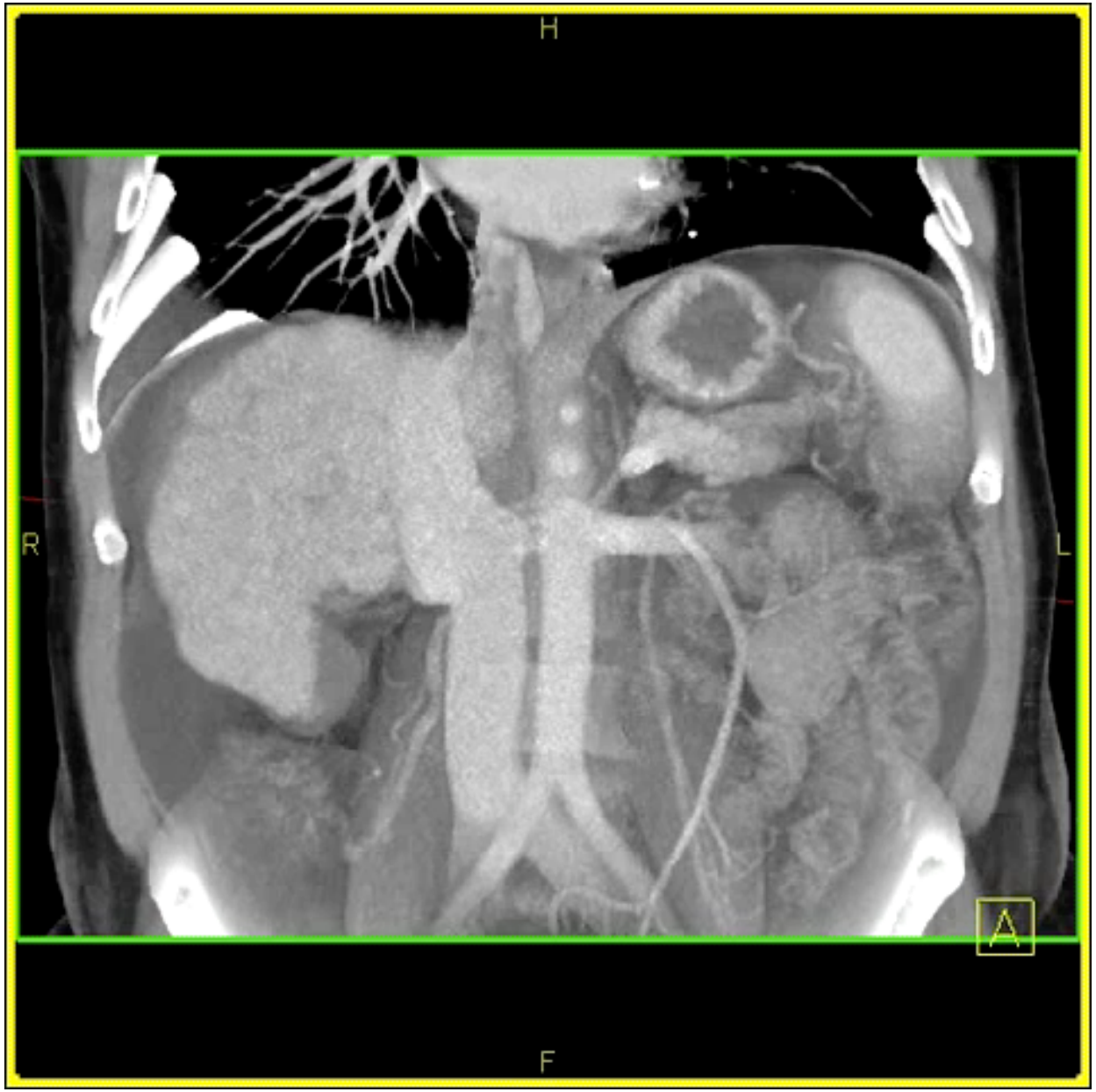
- Problem: if performed on the whole volume, the result is a simple x-ray-like (similar to x-ray) image

- Solution: use variable width slices to perform MIP

- Limitation of MIP: object orientation unclear.



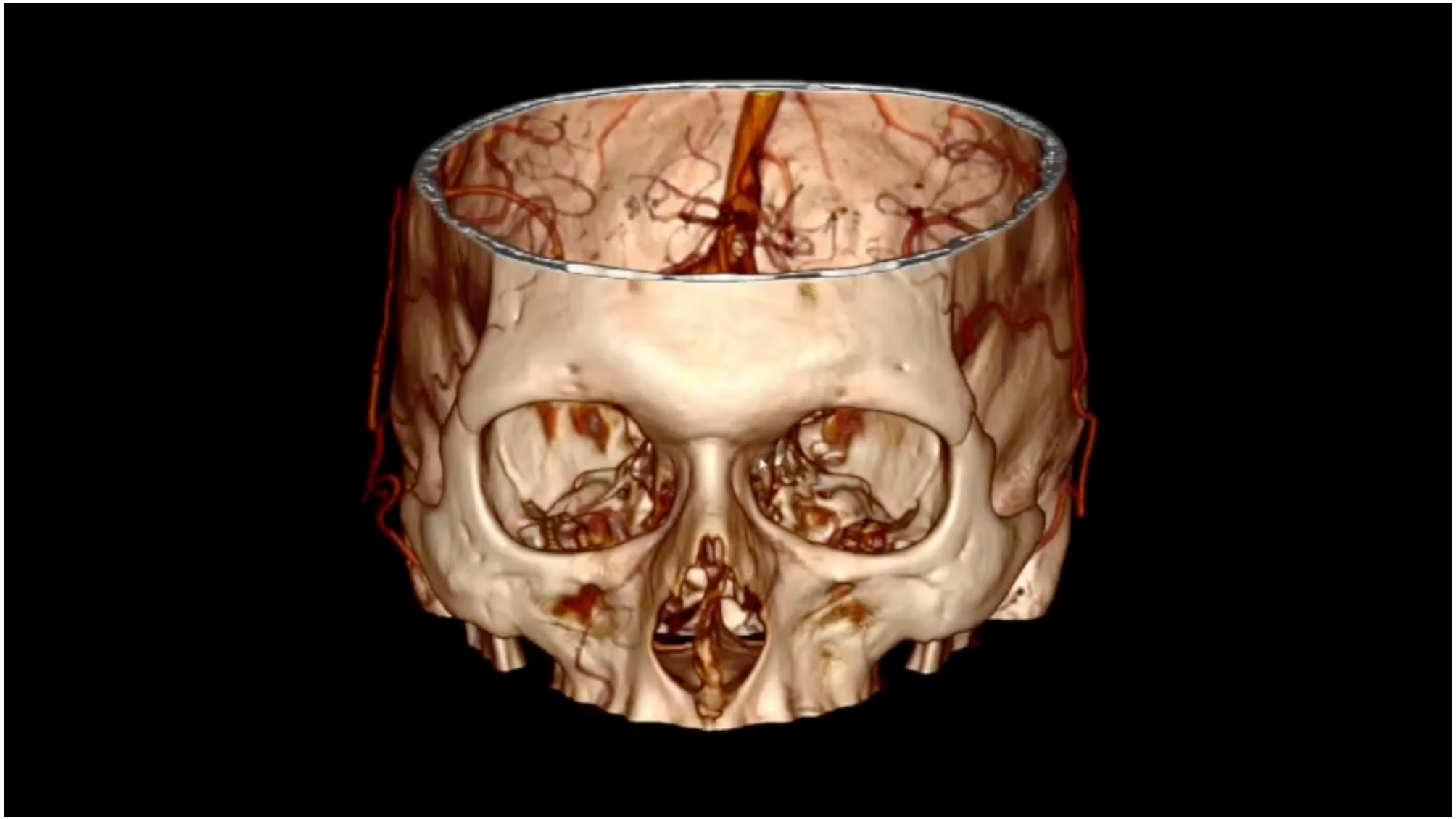
Volume rendering: Maximum intensity projection



Volume Rendering - ray casting

- Transfer functions (color and opacity) define how a given pixel intensity is mapped.
- Visualization apps implement interactive transfer function adjustment (using mouse movement)
- Level of Detail (LOD) can be used to reduce computational demand when rotating/moving the scene by reduction of render resolution.

Volume Rendering



Volume Rendering: Ray casting code

- Define how intensities map to colors and opacity

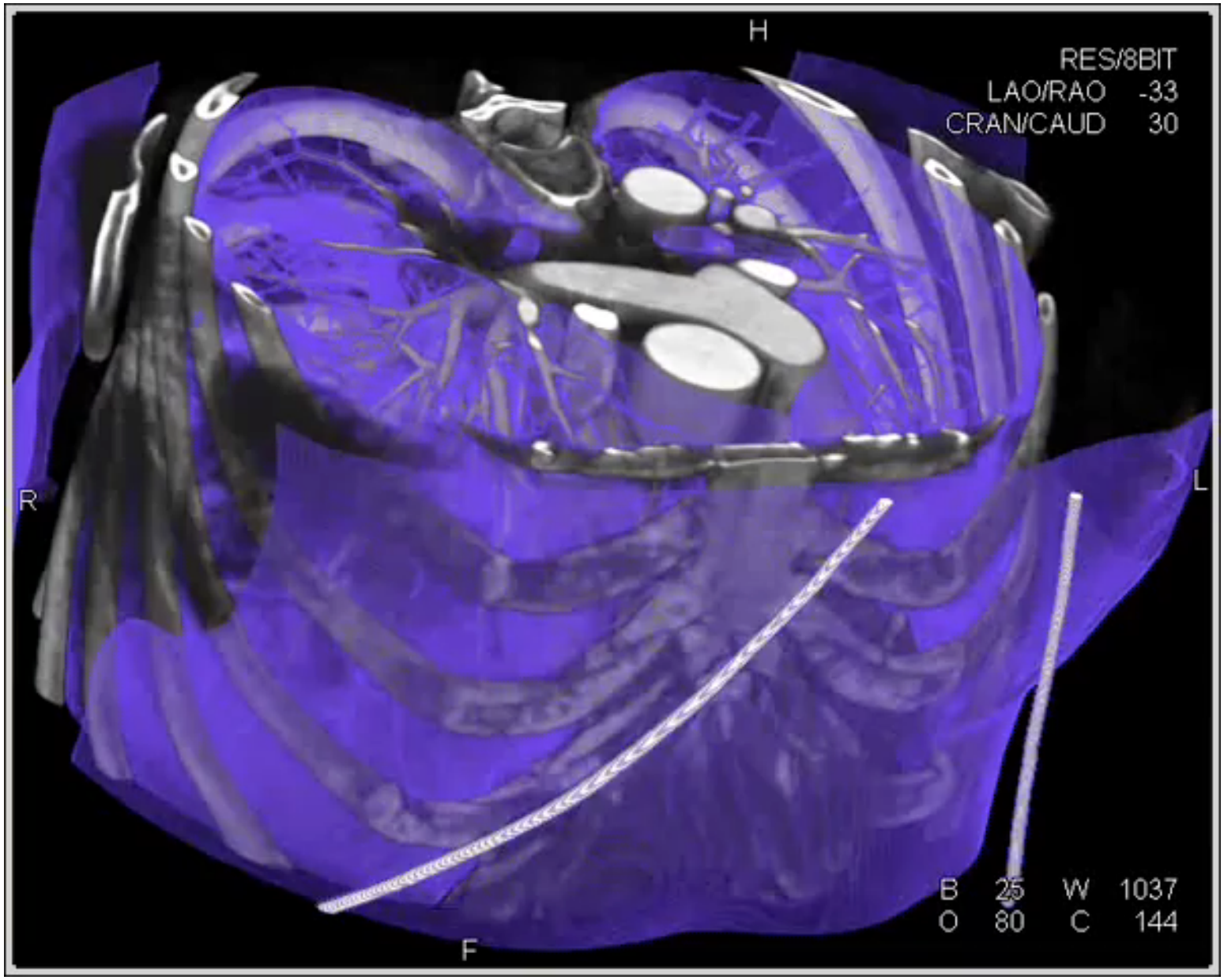
```
vtkSmartPointer<vtkPiecewiseFunction> opacity =
        vtkSmartPointer<vtkPiecewiseFunction>::New();
opacity->AddPoint(0.0,0.0);
opacity->AddPoint(80.0,1.0);
opacity->AddPoint(80.1,0.0);
opacity->AddPoint(255.0,0.0);

vtkSmartPointer<vtkColorTransferFunction> color =
        vtkSmartPointer<vtkColorTransferFunction>::New();
color->AddRGBPoint(0.0 ,0.0,0.0,1.0);
color->AddRGBPoint(40.0 ,1.0,0.0,0.0);
color->AddRGBPoint(255.0,1.0,1.0,1.0);
```

Volume Rendering: Ray casting code

- Define how intensities map to colors and opacity

```
vtkSmartPointer<vtkSmartVolumeMapper> mapper =
        vtkSmartPointer<vtkSmartVolumeMapper>::New();
mapper->SetBlendModeToComposite();
mapper->SetInputData(imageData);

vtkSmartPointer<vtkVolumeProperty> prp =
        vtkSmartPointer<vtkVolumeProperty>::New();
prp->ShadeOff();
prp->SetInterpolationType(VTK_LINEAR_INTERPOLATION);
prp->SetScalarOpacity(opacity);
prp->SetColor(color);

vtkSmartPointer<vtkVolume> volume =
        vtkSmartPointer<vtkVolume>::New();
volume->SetMapper(mapper);
volume->SetProperty(prp);
renderer->AddViewProp(volume);
```

Combining Visualizations



Combining Visualizations: Coronary OCT