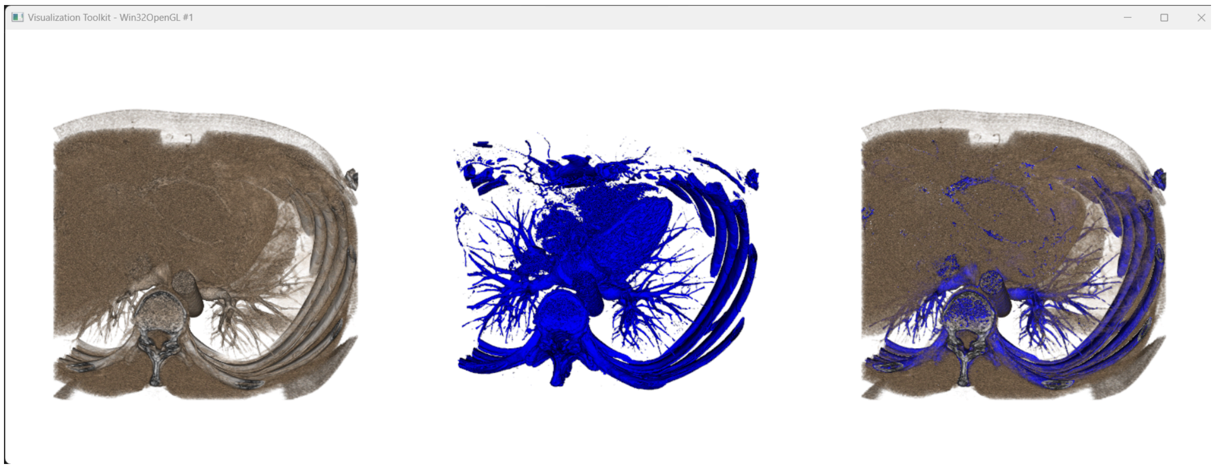
# Graphics & Animation

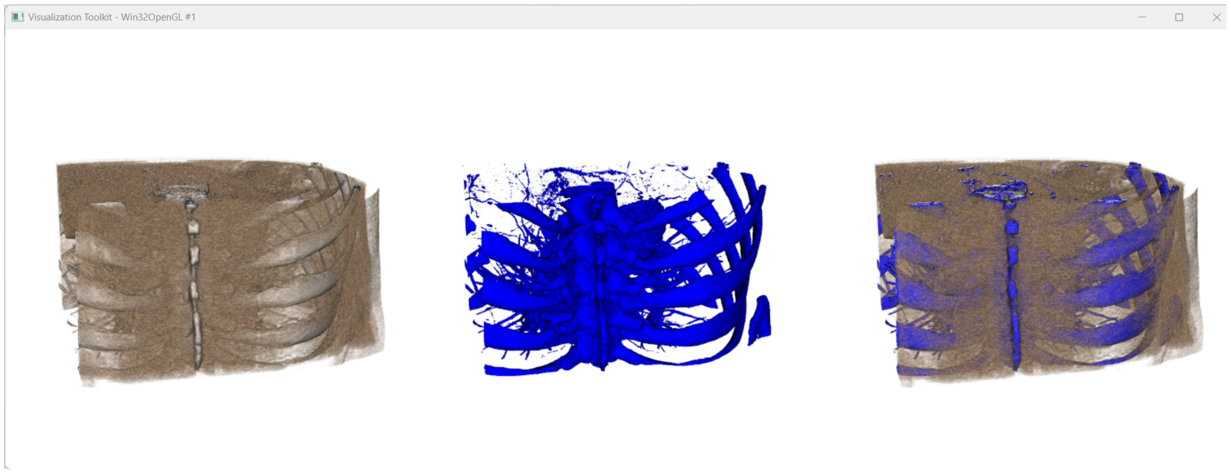
## Assignment 2

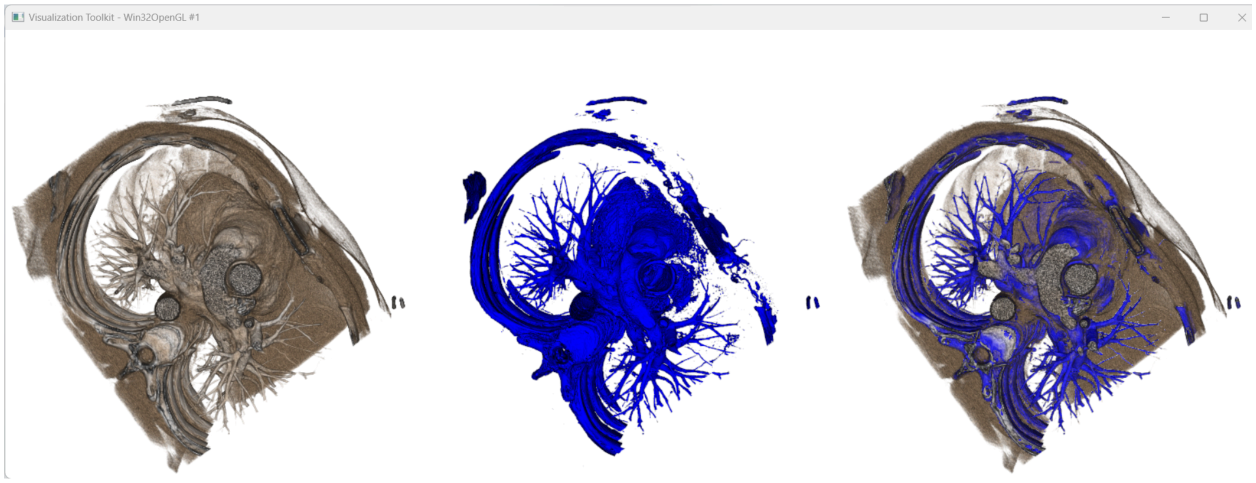
## Dataset details

- Dimension: (512, 512, 188)
- Voxel Resolution: (0.48828125, 0.48828125, 0.6999969482421875)
- Minimum Pixel Intensity: -1024.0
- Maximum Pixel Intensity: 3071.0
- Dataset-size: 96.9 MB

## Output Image as screen-shots in various rotations







## Source Code Highlights

Full source code available at: [https://github.com/dharanUoA/MM804Assignment2](https://github.com/dharanUoA/MM804Assignment2)

1. Reading the data-set

```Python
# Specify the path to your DICOM files
dicom_directory = "CT"

# Create a DICOM reader
reader = vtk.vtkDICOMImageReader()
reader.SetDirectoryName(dicom_directory)
reader.Update()
```

2. Color Transfer function for volume rendering

```Python
volumeProperty = vtk.vtkVolumeProperty()

# Create a color transfer function
colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(-1024, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(-512, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(0, 0, 0, 0) # Ignore the negative points
colorTransferFunction.AddRGBPoint(100, 1.0, 0.8, 0.6)  # Soft tissues in brown
colorTransferFunction.AddRGBPoint(500, 1.0, 1.0, 1.0)  # Bones in white

volumeProperty.SetColor(colorTransferFunction)
```

3. Opacity transfer function for volume rendering

Python
```python
# Create an opacity transfer function
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(-1024, 0.0)
opacityTransferFunction.AddPoint(-512, 0.0)
opacityTransferFunction.AddPoint(0, 0.0) # Ignore the negative points
opacityTransferFunction.AddPoint(100, 0.2) # Soft tissues
opacityTransferFunction.AddPoint(500, 0.8)  # Bones

volumeProperty.SetScalarOpacity(opacityTransferFunction)
```

Create a volume using the properties defined

Python
```python
# Create a volume mapper
volumeMapper = vtk.vtkSmartVolumeMapper()
volumeMapper.SetInputConnection(reader.GetOutputPort())

# Create a volume
volume = vtk.vtkVolume()
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)
```

4. Divide the renderer window in 3 columns (viewports)

```python
# Create a renderer and render window
renderer1 = vtk.vtkRenderer()
renderer1.AddVolume(volume)
renderer1.SetBackground(1.0, 1.0, 1.0)
renderer1.SetViewport(0.0, 0.0, 0.33, 1.0)

renderer2 = vtk.vtkRenderer()
renderer2.SetBackground(1.0, 1.0, 1.0)
renderer2.SetViewport(0.33, 0.0, 0.66, 1.0)

renderer3 = vtk.vtkRenderer()
renderer3.SetBackground(1.0, 1.0, 1.0)
renderer3.SetViewport(0.66, 0.0, 1.0, 1.0)

renderWindow = vtk.vtkRenderWindow()
renderWindow.SetSize(1650, 550)
renderWindow.AddRenderer(renderer1)
renderWindow.AddRenderer(renderer2)
renderWindow.AddRenderer(renderer3)

# Set up a render window interactor
renderWindowInteractor = vtk.vtkRenderWindowInteractor()
renderWindowInteractor.SetRenderWindow(renderWindow)
```

5. Display iso-surface extracted at suitable intensity for viewport 2

```python
# iso surface using marching cube algorithm
isoSurface = vtk.vtkMarchingCubes()
isoSurface.SetInputConnection(reader.GetOutputPort())
isoSurface.SetValue(0, 200)

# Create a mapper for the iso-surface
isoMapper = vtk.vtkPolyDataMapper()
isoMapper.SetInputConnection(isoSurface.GetOutputPort())

# Create an actor for the iso-surface
isoActor = vtk.vtkActor()
isoActor.SetMapper(isoMapper)

# Add iso actor to viewport 2
```

```
renderer2.AddActor(isoActor)
```

## 6. Display both results of viewport 1 and 2 in viewport 3

Python

```python
# Add volume and iso actor both in viewport 3
renderer3.AddVolume(volume)
renderer3.AddActor(isoActor)
```

## 7. Synchronize all 3 viewports

Python

```python
# sync all renderer cameras
camera = renderer1.GetActiveCamera()
renderer2.SetActiveCamera(camera)
renderer3.SetActiveCamera(camera)

renderer1.ResetCamera()
renderer2.ResetCamera()
renderer3.ResetCamera()
```