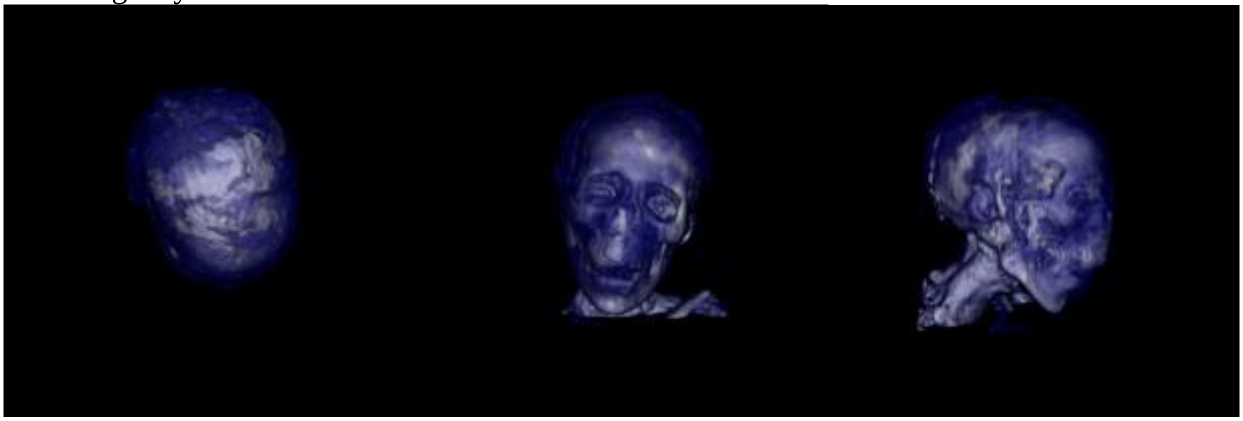
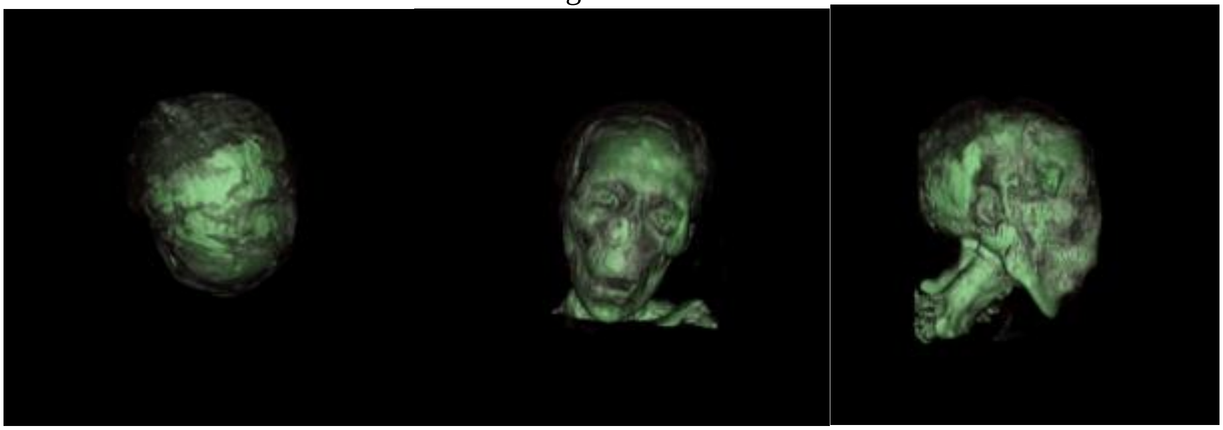
Assignment of DVR
Xiangzhen Sun

(1). Two tcl codes were translated into python codes.

(2). I have tested different colors and different opacities through modifying the transfer functions. Specifically, modifying the AddRGBPoint() has a significant effect on color presentation. The vol_ren.py file visualizes the mummy structure, and the color of skin can be modified in the following way:



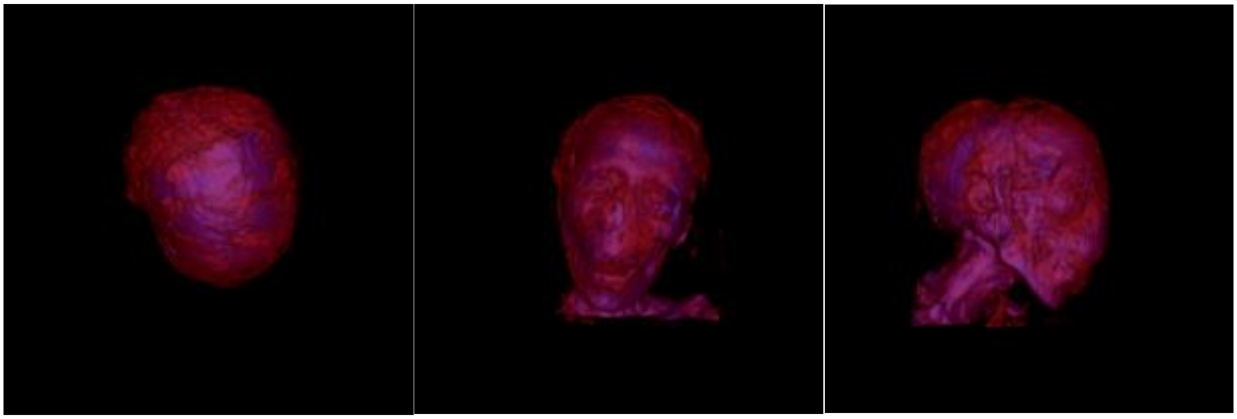
Original version



Parameters are set as follows:

```
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(0, 0.0)
opacityTransferFunction.AddPoint(50, 0.0)
opacityTransferFunction.AddPoint(55, 0.1)
opacityTransferFunction.AddPoint(80, 0.1)
opacityTransferFunction.AddPoint(100, 1.0)
opacityTransferFunction.AddPoint(120, 1.0)

colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(0, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(50, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(55, 0.0, 0.0, 0.6)
colorTransferFunction.AddRGBPoint(80, 0.0, 0.0, 0.6)
colorTransferFunction.AddRGBPoint(100, 1.0, 1.0, 1.0)
colorTransferFunction.AddRGBPoint(120, 1.0, 1.0, 1.0)
```

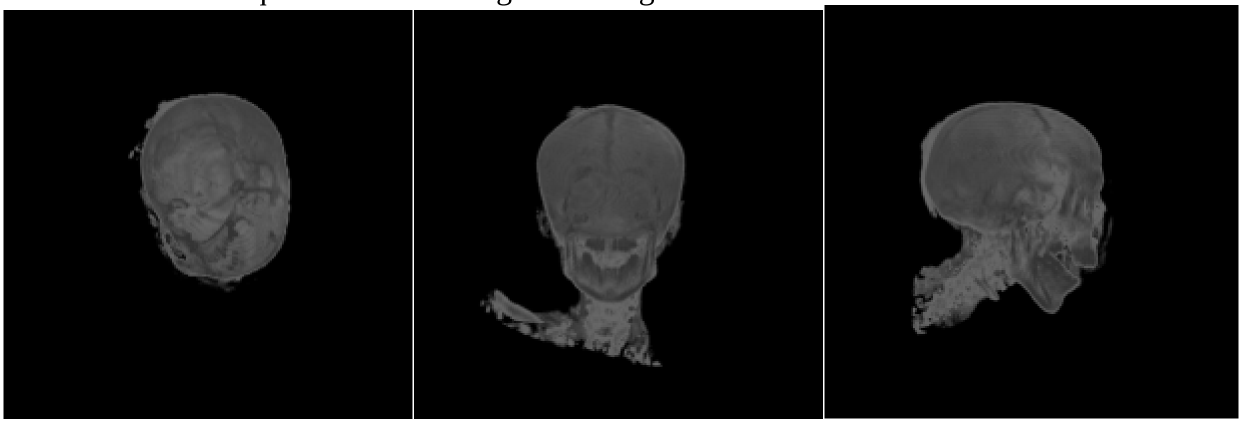


Parameters are set as follows:

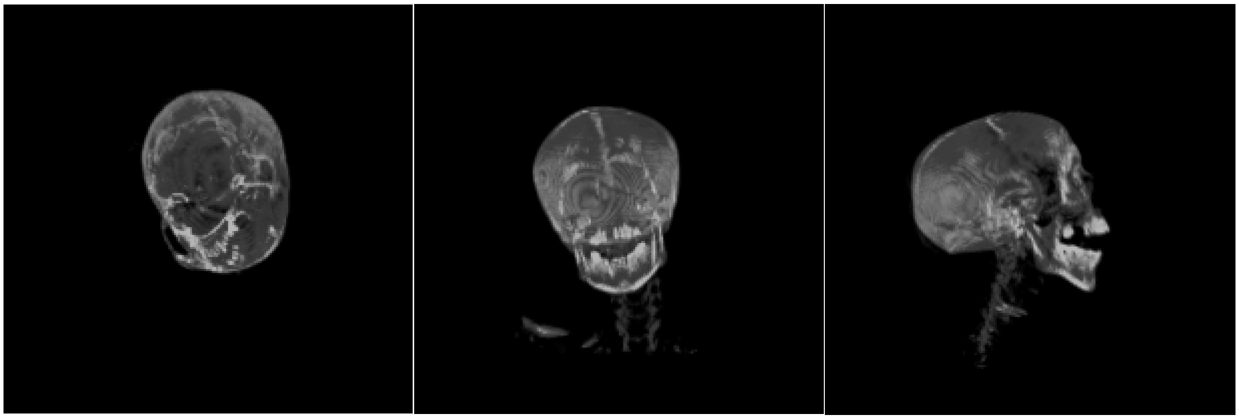
```
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(0, 0.0)
opacityTransferFunction.AddPoint(50, 0.0)
opacityTransferFunction.AddPoint(55, 0.1)
opacityTransferFunction.AddPoint(80, 0.1)
opacityTransferFunction.AddPoint(100, 1.0)
opacityTransferFunction.AddPoint(120, 1.0)

colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(0, 1.0, 0.8, 0.7)
colorTransferFunction.AddRGBPoint(10, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(30, 1.0, 0.5, 0.4)
colorTransferFunction.AddRGBPoint(60, 0.0, 0.0, 0.0)
colorTransferFunction.AddRGBPoint(80, 1.0, 1.0, 0.9)
colorTransferFunction.AddRGBPoint(100, 0.5, 1.0, 0.5)
```

The vol_mip.py file creates the maximum intensity projection of the above image. By changing the opacity transfer function, we were actually changing the color contrast of the image, other than the color itself. I acquired the following interesting results:



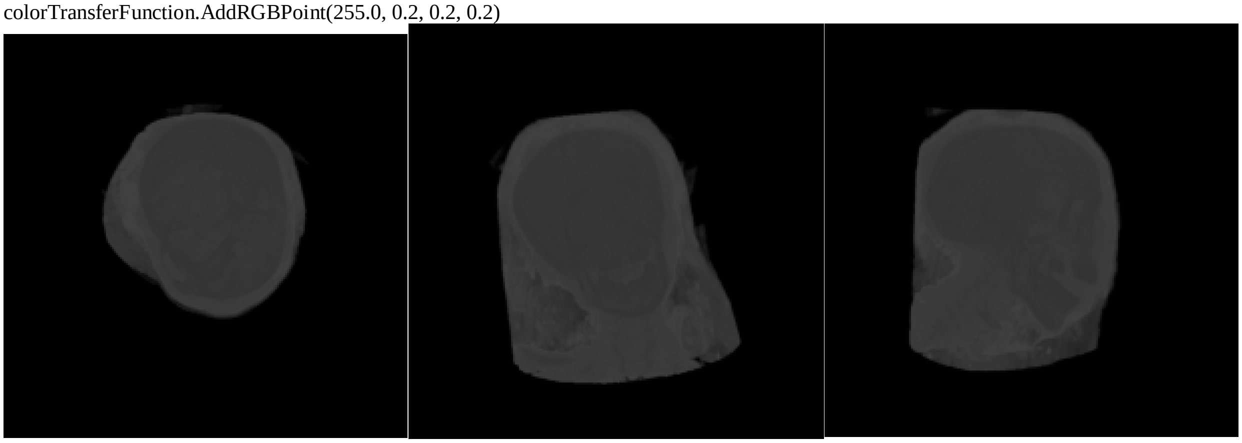
Original Version



Parameters as set as follows:

```
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(0, 0.0)
opacityTransferFunction.AddPoint(75, 0.0)
opacityTransferFunction.AddPoint(95, 0.0)
opacityTransferFunction.AddPoint(110, 0.4)
opacityTransferFunction.AddPoint(120, 0.6)
opacityTransferFunction.AddPoint(145, 1.0)
opacityTransferFunction.AddPoint(255, 1.0)

colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(75.0, 0.4, 0.4, 0.4)
colorTransferFunction.AddRGBPoint(95.0, 0.8, 0.8, 0.8)
colorTransferFunction.AddRGBPoint(110.0, 0.6, 0.6, 0.6)
colorTransferFunction.AddRGBPoint(120.0, 0.6, 0.6, 0.6)
colorTransferFunction.AddRGBPoint(145.0, 0.4, 0.4, 0.4)
colorTransferFunction.AddRGBPoint(255.0, 0.2, 0.2, 0.2)
```



Parameters are set as follows:

```
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(0, 0.0)
opacityTransferFunction.AddPoint(100, 0.0)
opacityTransferFunction.AddPoint(125, 0.0)
opacityTransferFunction.AddPoint(200, 0.4)
opacityTransferFunction.AddPoint(250, 1.2)
opacityTransferFunction.AddPoint(500, 1.5)
opacityTransferFunction.AddPoint(1025, 2.0)

colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBPoint(100.0, 0.4, 0.4, 0.4)
colorTransferFunction.AddRGBPoint(125.0, 1.2, 1.2, 1.2)
colorTransferFunction.AddRGBPoint(200.0, 0.6, 0.6, 0.6)
colorTransferFunction.AddRGBPoint(250.0, 0.6, 0.6, 0.6)
colorTransferFunction.AddRGBPoint(500.0, 0.4, 0.4, 0.4)
colorTransferFunction.AddRGBPoint(1025.0, 0.1, 0.1, 0.1)
```

(3). Filters Analysis

In vol_mip.py, there was only one filter used, namely vtkWindowToImageFilter(). In vol_ren.py, except for the vtkWindowToImageFilter(), a vtkOutlineFilter was also employed. So we start our discussion from the vtkWindowToImageFilter():

The purpose of using vtkWindowToImageFilter() was to read the data in a vtkWindow and use it as input to the imaging pipeline. This is useful for saving an image to a file in our case. The window can be read as either RGB or RGBA pixels; in addition, the depth buffer can also be read. RGB and RGBA pixels are of type unsigned char, while Z-Buffer data is returned as floats. Use this filter to convert RenderWindows or ImageWindows to an image format.

The vtkWindowToImageFilter() in both files are not replaceable, because we need to save window image to local path as png files. Thus no other filter can replace it.

Next, we move on to the discussion of vtkOutlineFilter, which only appeared in vol_ren.py. The purpose of using vtkOutlineFilter() is easy to describe: it generates a wireframe outline of the given data: mummy.128.vtk. The outline consists of the twelve edges of the dataset bounding box. This filter takes in the input dataset as source and generates a outline as bounding. If the vtkOutlineFilter() were used in the vol_mip.py, then it is unnecessary because a CT-scan does not need boundary to interfere. However, in our second file vol_ren.py, we were ray-casting the image of the whole mummy skin, which needs outlines and boundaries to implicitly depict the layers of the original structures. So it is also unreplaceable.