Neuroimaging visualization

Scientific visualization is one of the most important topics of computer graphics. Accurate representation of measurements or calculation results is crucial for interpreting the data and developing accurate mathematical models and theories, as well as prototypes in practice. Numerous applications are in disciplines such as medical image processing, mechanics of particles, fluid dynamics, molecular biology, protein structure examination, etc. In many cases the obtained results can be understood and well represented only by visualization (either 2-D or 3-D). The advance in imaging technologies has made it possible to combine images with recorded signals, which is especially used in electro-encephalography (EEG) monitoring. Electrodes record the function of the brain, which is mapped to the brain model. This method is widely used in monitoring neonatal babies with diseases such as epilepsy.

Neuroimaging (and medical) visualization is most often implemented using the Visualization Toolkit (VTK). VTK is an open-source object-oriented library for visualization in C++, C#, Tcl, Python, Java, and is positioned at a higher level than other common rendering libraries (like OpenGL). To enhance the user experience and ease user interaction, VTK can easily be bound with many GUI libraries, such as Qt, FLTK, Tcl/Tk.

Task 1: Basic visualization

Basic visualization involves display of original (CT or MRI) images, with mesh representation of the head and brain (gray matter, white matter, lesion). The goal of this task is to effectively visualize the neonatal head model (later to be used for EEG monitoring).

The data set consists of a series of a single segmented VTK image "head_with_lesion.vtk" (2 folders with png slices are also provided, so you can view the images in an ordinary image viewer).

NOTICE 1: The user interaction in VTK is implemented using keys. You have free choice of keys. In other words, to call certain functionality you can use any key you wish (however, what each of the keys does will have to be printed on the screen menu).

NOTICE 2: If you use a GUI toolkit (e.g. Qt), you do not have to make a printed menu on the screen (so, disregard the previous notice). Also, you do not have to use keys for user interaction – you can simply use Qt widgets. IT IS HIGHLY RECOMMENDED THAT YOU USE A GUI TOOLKIT! If you want to use Qt with VTK you will have to build VTK with Qt support (consult online documentation).

NOTICE 3: If you use the 3-D image class supplied in project materials, each (3D) voxel gray value can be accessed using indexes (discrete coordinates) of the voxel (e.g. image(1,2,3) will return the gray voxel value at coordinates (1,2,3)). There are 26 neighboring voxels for a voxel with indexes (i,j,k),in 3D (9 above, 9 below and 8 on the same level):
(i-1,j-1,k-1), (i-1,j-1,k), (i-1,j-1,k+1), (i-1, j,k-1), (i-1,j,k), (i-1,j,k+1), (i-1,j+1,k-1), (i-1,j+1,k), (i-1,j+1,k+1);
(i,j-1,k-1), (i,j-1,k), (i,j-1,k+1), (i, j,k-1), (i,j,k+1), (i,j+1,k-1), (i,j+1,k), (i,j+1,k+1);
(i+1,j-1,k-1), (i+1,j-1,k), (i+1,j-1,k+1), (i+1, j,k-1), (i+1,j,k), (i+1,j,k+1), (i+1,j+1,k-1), (i+1,j+1,k), (i+1,j+1,k+1).

a) Displaying images:

- Load given VTK file(s). Examine the (legacy!) VTK file header and explain what each of the elements in the header means. Is the data in the given files compressed?
- Incorporate the use of the three planes (sagittal, transversal and coronal) by pressing the "s", "t" and "c" keys respectively (or using widgets in Qt), where each of the planes has to show the cut from the original data set (use vtkImagePlaneWidget). Incorporate scrolling through the slices by pressing "+" or "-". The interaction in VTK is implemented either by subclassing the vtkCommand or by subclassing interactor style (e.g. subclassing vtkInteractorStyleTrackballCamera).
- Make the basic menu on the screen using vtkTextActor (not needed in case Qt is used).

b) Visualizing meshes and centerlines:

- Use the marching cubes algorithm (vtkContourFilter) to render the mesh with the following characteristics. Use different keys to show different segmented parts of the head (skin, brain, grey matter and lesion). How does the mesh look like? Smooth or not smooth enough? Try using a Smooth filter on the mesh to obtain a nicer visualization. Another possibility is to smooth the image and use the contour filter, try it out.
- Make an interactive visualization. In other words, you have to be able to turn on or off the visibility of all created actors (image planes, meshes). Notice that, in order for some of the meshes to be visible, at the same time with the mesh of the head, you must select the correct opacity for the mesh of head (and preferably assign it white color).

Task 2: EEG visualization

The advance in imaging technologies has made it possible to combine images with recorded signals, which is especially used in electro-encephalography (EEG) monitoring. Electrodes record the function of the brain, which is mapped to the brain model. The goal of this task is to implement a EEG visualization system using simulated data.

- Allow the user to pick 8 positions on the model of the brain (use vtkPointPicker). These positions will represent positions of electrodes. Color the brain mesh as in Fig .2 by assigning values 0, 50 or 100 (in a color table with range [0,100] the colors will vary from red to blue) and by coloring all intermediate points in the mesh based on the distance from the electrodes. Derive an equation for linear dependency on the distance and for dependency on the squared distance. Use random values of 0, 50 and 100 to show colors on the mesh.
- Position spheres as electrodes (similarly as in Fig. 2). See if you can use Glyph filter here.
- Use the timer to generate random values for each electrode (the values of 0, 50, 100) for each electrode every 5 seconds and to recalculate the colors for the rest of the mesh to obtain a simulation of EEG activity.
- There should be other render windows which will show the signals (use vtkChartXY) for each electode at the given moment like in Fig.2 (multiple render windows are more easily manipulated in Qt). The windows showing the signals should be of fixed size (in terms of the time interval T which is displayed in them). This means that only the last T samples of the signals will be visible.

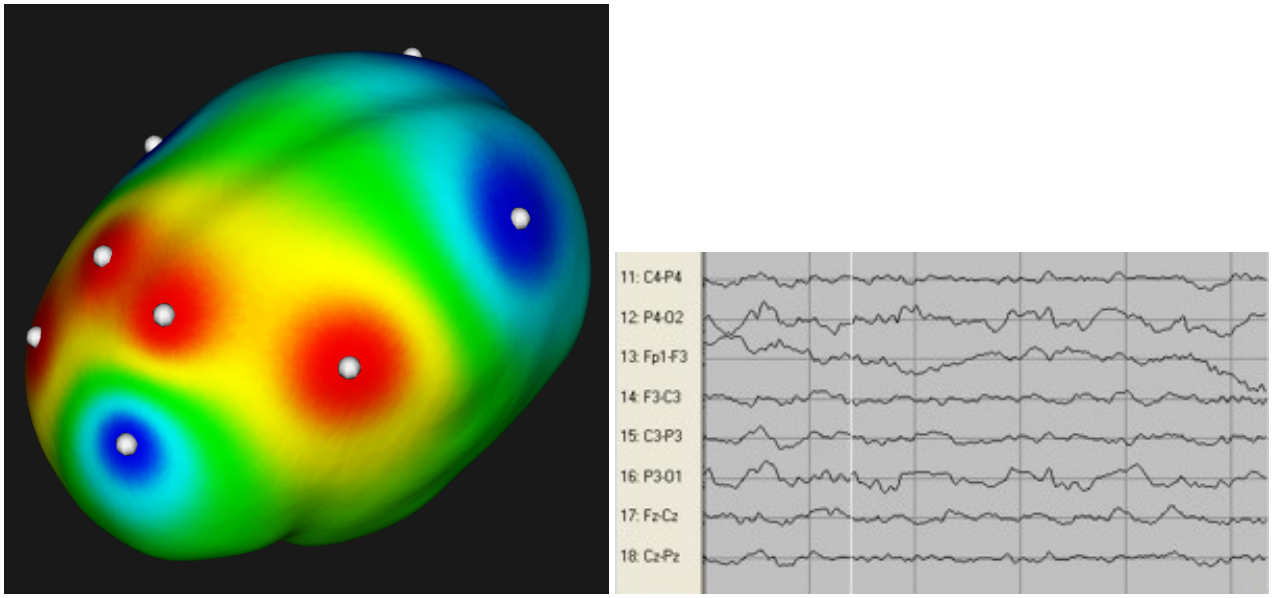


Fig. 2 An example of EEG activity visualization

In order to obtain the coloring of the mesh, you will have to manipulate vtkPolyData.

Task 3: Visualization of Digital Subtraction Angiography Images

Digital Subtracted Angiography (DSA) and is the modality on which the interventional neuroradiology is based. DSA is an example of a temporal subtraction, which means that a number of images are taken from the same view point at periodic time intervals. This allows us to follow the propagation of contrast agent (fluid that stands out in X-ray images) through the blood vessel system over time. To visualize DSA images, a single color image is produced where each color represents a different time moment in which the (highest concentration of) contrast fluid flows through a given point (pixel in the image). For example, in Fig. 3 the blood vessels colored in red are the ones through which the contrast fluid passed, while the blue blood vessels are the ones through which the contrast fluid passed the last.

The data set consists of multiple series of DSA images (PNG) of brain blood vessels from 2 views. a single segmented VTK image

- Devise a method that will convert the series of DSA images (which are gray images) into a single color image as in Fig. 3. Take into account that there are different ways in which a pixel can be assigned its color (e.g. coloring by the frame in which the maximum of contrast fluid is found, or using the color to represent also the whole interval in which the contrast is visible at the given pixel (averaging over multiple frames)).

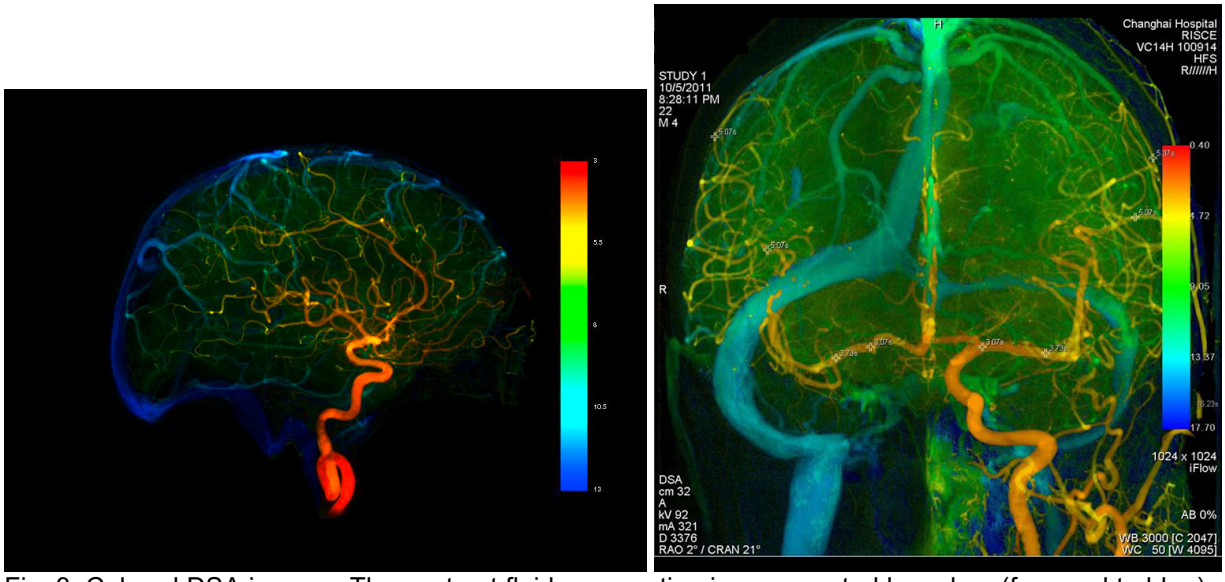
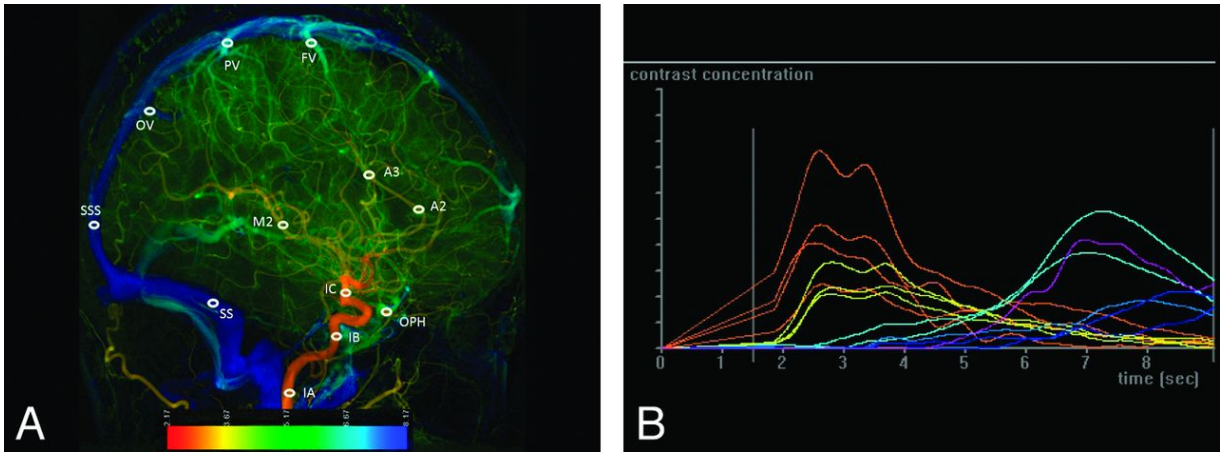


Fig. 3: Colored DSA images. The contrast fluid propagation is represented by colors (from red to blue)

Another interesting application is the visualization of the contrast agent flow curves at selected pixels (or group of pixels) in DSA images. Fig. 4B shows the contrast agent flow curves calculated at marked pixels (it could be that the nearest neighbors in marked pixels are also taken into account for this calculation) in Fig. 4A

- Devise a visualization method that will allow the user to select a point in the colored DSA image and which will show the calculated flow at that pixel in the separate window (similarly as in Fig. 4).



To read PNG files refer to vtkPNGReader. To display a color scalar bar take a look at vtkScalarBarActor and vtkScalarBarWidget. To map gray pixel values to colors take a look at vtkLookupTable and vtkImageMapToColors.