SECV 3233 – Data Visualisation

Farhan bin Mohamed
Mohd Shafry Mohd Rahim

Volume Visualisation and Flow Visualisation

"Get your insights right"

Number of Credits: 35% of a 3 credit module
Recommended Hours: 20 – 30 hours
Submission deadline (Presentation): 25 June 2023

Group Project: 3 – 4 persons

Important! The given library callings are in the previous version of VTK. Please update the library calling and methods according to the latest version of the VTK.

Part 1 – Volume Visualisation

Problem Statement

Create a volume-rendering program using the Visualization Toolkit (VTK). Explore various datasets to find good iso-surfaces and transfer functions.

Tips: To clarify the goal of this assignment: When you are developing a real visualization application, it is likely you'll be given a volume in some format you have never seen and without knowledge of where interesting surfaces are located. You'll either need to convert the volume to a format you understand or search your API (VTK) for a class that reads your volume. You will also have to add some code to either identify interesting surfaces (i.e., using a histogramming approach) or allow you to interact with the volume to select good surfaces (say, by interactively changing the iso-value used for contouring until you find good surfaces). This option essentially asks you to do both.

1A (maximum marks: 25)

Download and compile (or reproduce) the VTK volume rendering programs discussed in class, namely an iso-surface extraction program and a ray marching program.

- **5 points:** Get the programs running, load in the head model shown in class.
- **10 points:** Add a keyboard call-back, allowing you to change the iso-value (for the iso-surface program) and the ray step size (for the ray marching program).
- **10 points:** Load in at least three other volumes (not necessarily simultaneously). *A number of alternative models are provided on the class web page,* **but at least one model you use must be downloaded from elsewhere** *(links are also provided). Datasets from other departments or your own research qualify.*

farhan@utm.my

- *Hint:* It might be nice if you add an additional keyboard call-back that allows you to alternate rendering modes between ray marching and iso-surface extraction in the same program.

1B (maximum marks: 25)

Add to your program(s) the ability to interact with the data in more complex ways. You can do this by adding additional mouse clicks/keyboard callbacks, or by using VTK widgets.

- **10 points:** Locate interesting surfaces by varying the iso-value used to extract iso-surfaces in your datasets.
  Answer in your submitted README file (on a cloud shared folder via e-learning): How many important surfaces does each of your volumes have? What do these surfaces correspond to (e.g., "skin")?
- **10 points:** For each volume, setup a transfer function for the dataset that uses appropriate colors and clearly displays multiple relevant iso-surfaces (e.g., the skin and bone from the head dataset shown in class). You must either show *all* important iso-surfaces (using the transfer function) or discuss why a larger number of surfaces make the image hard to comprehend. If your chosen datasets do not have clearly defined "surfaces," make sure your transfer function has a smoothly changing opacity to allow visibility of the internal structure.
- **5 points:** Add some additional user-controllable parameter(s) to the ray marching program. This is an open ended problem, and you can select any parameter (other than the ray step size, from Problem 1) to implement.
  Some suggestions might include: dynamically modifying the opacity transfer function, dynamically modifying the color transfer function, adding a multi-dimensional transfer function.
- **IMPORTANT!** Please make sure all user-controls are either *obvious* on screen when I run your code *or* are clearly documented in a submitted "README" file you turn in with your assignment.

Part 2 – Flow Visualisation

Problem Statement

Create a simple flow visualization application in VTK using a "hedgehog," a glyph-based representation, and streamlines.

Tips: In addition to the VTK text, I have tried to specifically mention all required classes by name, so you can search Google for Kitware's online class documentation if you need additional ideas how to start. However, the modifications to the code for this assignment should be relatively straightforward.

2A (maximum marks: 50)

Download, compile and improve the sample VTK flow visualization applications from the e-learning system.

- **10 points:** A hedgehog visualization. Note, the default program already displays using a hedgehog, but the lines are way too long. Determine appropriate hedgehog scaling factors for "testData1.vtk" "testData2.vtk" and "carotid.vtk." Place images of "an appropriate size" in your pdf document, or explain why this is not feasible.

- **10 points:** Instead of using a hedgehog, use a glyph-based visualization. In order to avoid creating a complex glyph such as an arrowhead, you should use a simple cone instead of an arrow.
  - You will need to replace the hedgehog inside *mapper->SetInputConnection(hhog->GetOutputPort())* with your new glyph-based filter specified using the *vtkGlyph3D* class.
  - The vtkGlyph3D filter requires a *SetInputConnection()* to attach the input vector data.
  - A *SetSourceConnection()* is required to specify the glyph (in this case a cone from a *vtkConeSource*).
  - Other methods you may need inside the vtkGlyph3D class include *SetScaleFactor()* and *SetScaleModeToScaleByVector().*

  Explore what size cones are most appropriate for the three datasets specified above and place images on your web page.

- **15 points:** Now, you'll implement a streamline based visualization. Again, this is performed using a filter analogous to *vtkGlyph3D* or *vtkHedgeHog*. The class you need is called *vtkStreamLine*. You can test a straightforward implementation by simply creating a *vtkStreamLine* class, and connecting the appropriate inputs and outputs.

  Unfortunately, by default *vtkStreamLine* uses only a single stream (not particularly useful for displaying details in a complex vector field).

  Fortunately, you can use numerous streamlines by specifying their start points using *streamLines->SetSource(startPoints),* where *startPoints* is a vtkPointSet.

  - Stream starting points must be in the range $0 \le x \le 76$, $0 \le y \le 49$, and $0 \le z \le 45$ for the carotid dataset.
  - Stream starting points must be in the range $0 \le x, y \le 36$, $z = 0$ for the testData1 dataset.
  - Stream starting points must be in the range $0 \le x, y \le 357$, $z = 0$ for the testData2 dataset.

  Note that the points in a *vtkPointSet* can be specified with the method SetPoints(), which takes an input of type *vtkPoints*.

  You can specify points in the *vtkPoints* class using the method *InsertPoint().*

  You may specify additional streamlines by either (or both):

  - Starting streamlines on a regular grid throughout the datasets.
  - Allowing the user to interactively add streamlines by clicking to add points where additional streams start.

  Explore to find an appropriate number of streamlines for each of the three datasets, and place the results on your web page.

- **15 points:** Using the *vtkStreamPoints* class, you can create sampling points along streamlines. When used in conjunction with the the *vtkGlyph3D* class, you can have your cones / arrows aligned tip-to-base, rather than in a regular grid pattern.

  Apply these techniques on other engineering/non-medical related dataset (e.g.; hurricane, engine block tensors, etc.)

farhan@utm.my

Dataset descriptions:

Volumetric datasets:
Head: 3.2:3.2:1.5
Frog: ~1:1:2.2
Aneurism: 1:1:1
Teapot: 1:1:1
Foot: 1:1:1

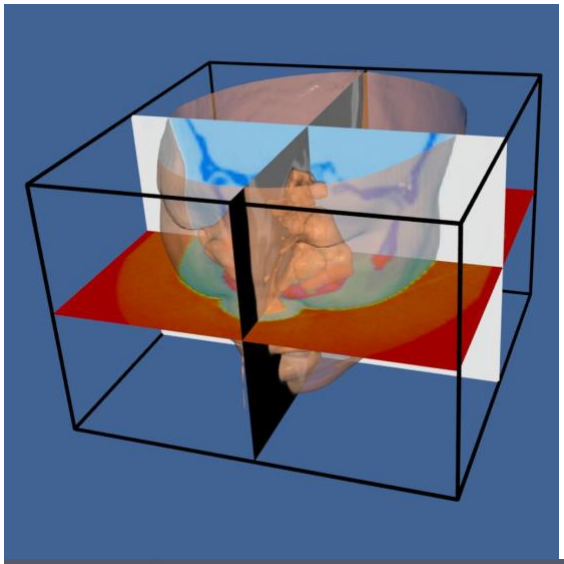
Vector datasets:
Carotid (76x49x45)
Test Data 1 (36x36x1)
Test Data 2 (357x357x1)

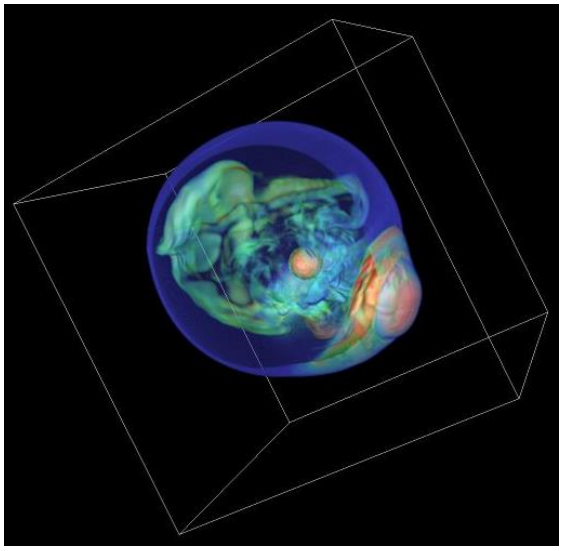
Other Sources of Volume Data:
Visible Human: http://www.nlm.nih.gov/research/visible/visible_human.html
Siggraph volume datasets collection: https://education.siggraph.org/resources/cgsource/instructional-materials/archives/volume-visualization-data-sets
Stanford: http://graphics.stanford.edu/data/voldata/

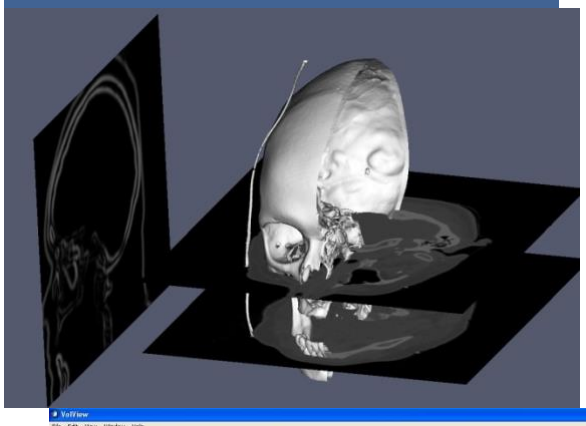
Good Luck!

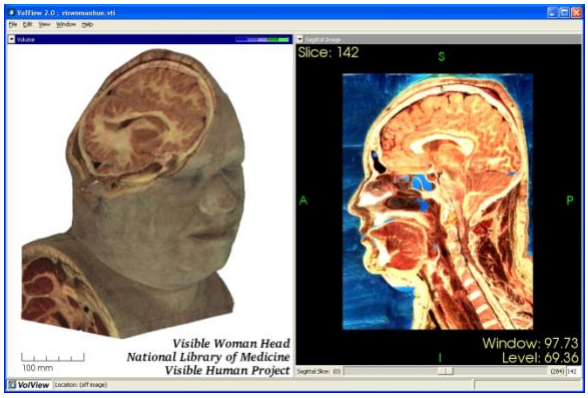
farhan@utm.my

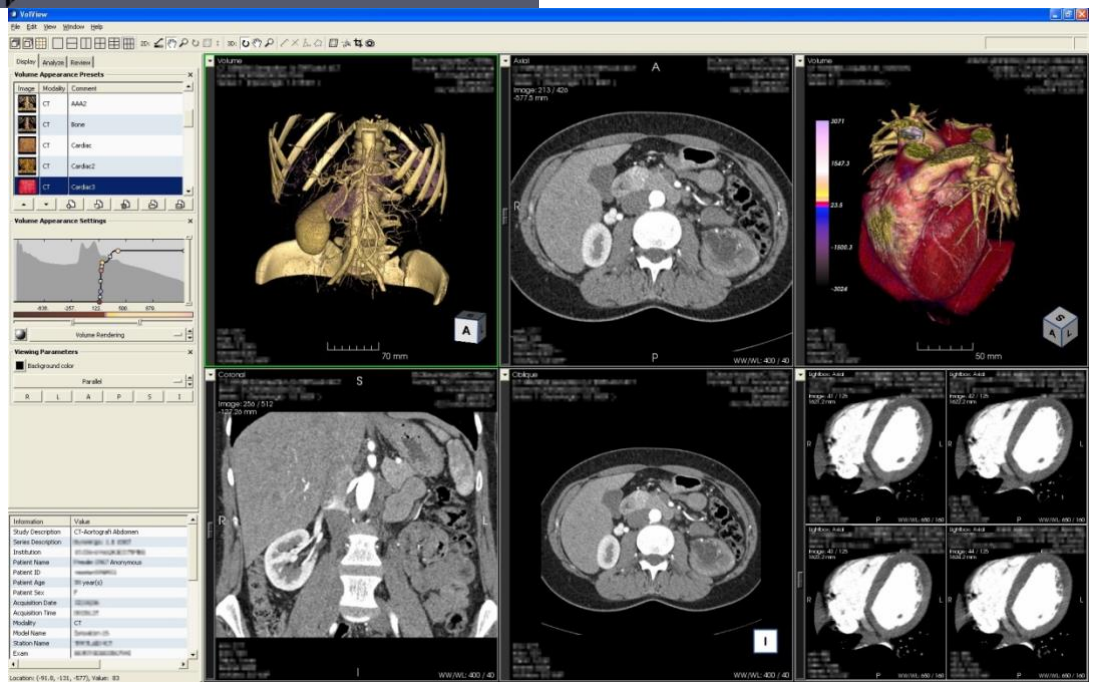
Personal Project Examples:





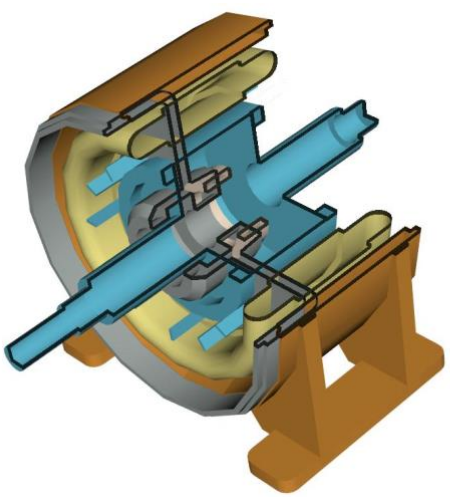


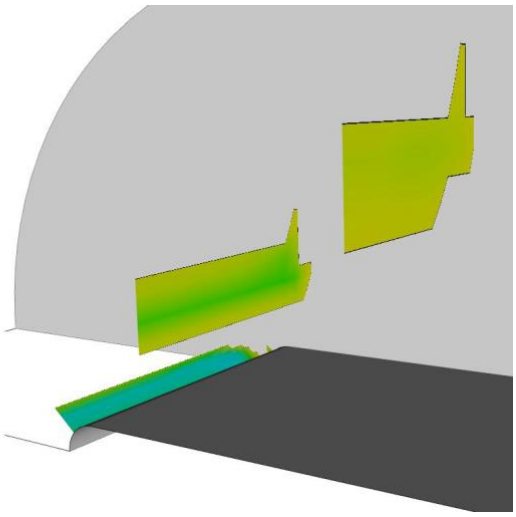


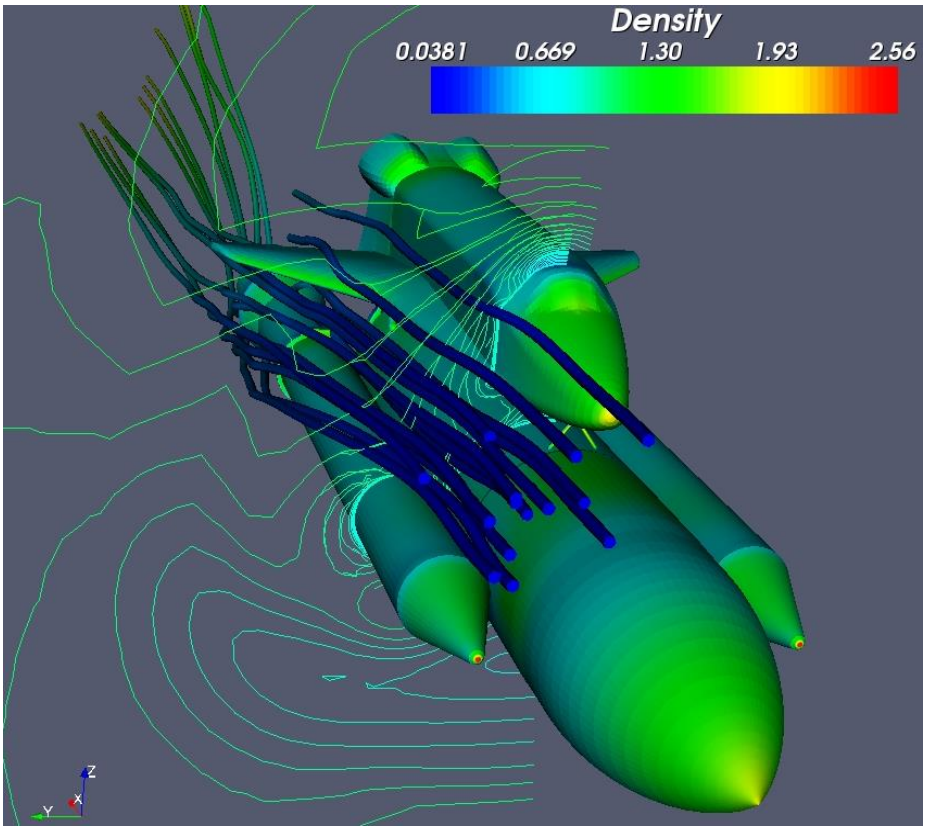


farhan@utm.my

Group Project Examples:







farhan@utm.my