ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scientific Visualization

Summer Semester 2006

Dr. Ronald Peikert, Filip Sadlo

Practical Exercise 1 - Paraview, Flow Topology

Handed out: May 19 2006
Hand in: June 16 2006
Author: Filip Sadlo

1. Introduction

The practical exercises are based on the Visualization ToolKit *VTK*. This is a system consisting of modules with input ports, parameters and output ports. The output port of a module can be connected to the input port of an other module. This way, the data generated by the first module is passed to the second module for processing. A module can have several input and output ports. An output port can be connected to several input ports, whereas input ports have often only a single connection attached. The resulting data flow network topology is generally a directed graph.
This concept of visual programming is very useful for prototyping but also for processing of arbitrary data. The modules can be seen as high-level functions and connecting them is equivalent of writing code that calls them sequentially. This brakes down the distinction between the programmer and the user.
Depending on the data and the application, a hard-wired software is often not sufficient. Usually this would require that the programmer modifies the software. But in such a modular system the user can easily do arbitrary changes to the software without programming, by inserting additional filters, adding additional views for inspection, and so on.
Unlike other comparable visualization systems, VTK does neither offer a graphical interface nor a graphical representation of the network. *Paraview* is a graphical front-end to VTK that offers a graphical interface but (actually) no graphical representation of the network. However, it turns out that the graphical representation of the network and the possibility to connect modules visually is not necessary at least for small networks.

The first part of the exercise is to get familiar with Paraview and to realize a simple visualization by using existing modules. The second part of the exercise is to program a VTK module and apply it in Paraview to the Isabel hurricane dataset. The exercise is done preferably on the Linux version of Paraview. If you want to do it on the Windows version, only limited support can be offered, especially regarding compiling and debugging of modules. However, do not hesitate to contact sadlo@inf.ethz.ch in case of any problems.

2. Paraview 4 Points

As an introduction, we want to visualize the Isabel hurricane dataset using existing modules.

a) Start up Paraview:
Under Linux, type: /afs/ethz.ch/users/p/peikert/paraview/inst/bin/paraview

b) Read the Isabel dataset:
File -> Open Data
-> /afs/ethz.ch/users/s/sadlof/scivis_SS06/exercise1/isabel_crop_tet.vtu -> Open
Hit the green "Accept" button (this makes the module execute)
Adapt the display of the dataset:
Change to the "Display" tab of the reader module
Try out the different choices for "Representation" and finally select "Outline"
Information about the data channels can be found at:
http://www.vets.ucar.edu/vg/isabeldata/readme.html

c) Generate streamlines using the StreamTracer filter module:
Filter -> Stream Tracer
Click the "Set Point Position to Center of Bounds" button
Hit the green "Accept" button
Go to the "Display" tab of the Stream Tracer Module
Try out the different choices for "Color by"
(Vorticity is the curl ($\nabla \times \boldsymbol{u}$) of the velocity $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}(u(\boldsymbol{x}), v(\boldsymbol{x}), w(\boldsymbol{x}))$. It is therefore often used in the examination of vortical flow.)

d) Navigate through the data:
Rotation: Left mouse button
Translation: Middle mouse button
Scaling: Right mouse button
Adapt view: 
Center: 

e) Now we will add an isosurface of velocity magnitude:
In the "Selection Window", select (make yellow) the reader module (isabel_ ...), because we want to compute an isosurface of the data, not of the streamlines.

e1) Because the data does not contain velocity magnitude, we have to compute it first:
This is done by the "Calculator" filter
Filter -> Calculator
In "Result Array Name", insert e.g. "velocity_magnitude"
Inside "Calculator" insert "mag(velocity)"
Hit the green "Accept" button
Go to the "Display" tab and select "Outline" representation

e2) Now we can compute the isosurface:
Make sure the Calculator is selected in the "Selection Window"
Filter -> Contour
Select a value inside "Add Value" and hit the "Add" button
Hit the green "Accept" button

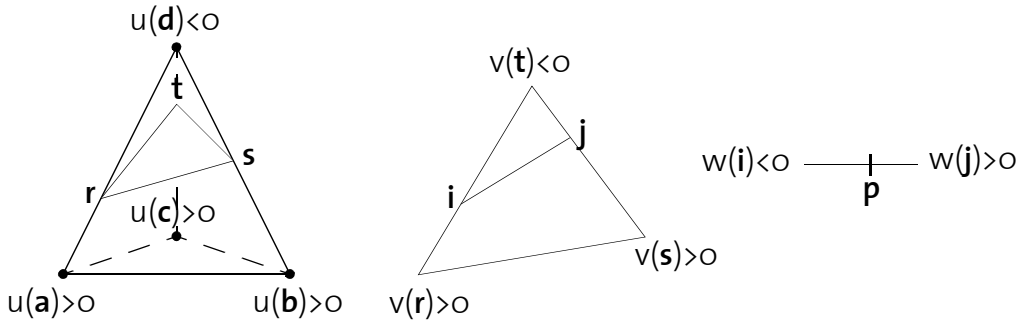
f) Now do further parametrization of the modules, add other modules, and include e.g. the pressure data channel, in order to achieve a visualization that allows interpretations (or at least observations) of the present hurricane mechanisms.
Save at least one adequate view as an image:
File -> Save View Image, preferably in PNG format

g) Write an image description containing information what the geometries represent, what the colors mean (including the meaning of low and high value), etc.
Add your explanations, observations (and interpretations).

Send the image(s) together with image description to sadlo@inf.ethz.ch.

3. Flow Topology Module 6 Points

Flow topology is primarily concerned with the detection and analysis of critical points. These are (isolated) points where the velocity vector is zero. The reader is referred to the (future) lecture slides and to the paper of Asimov for additional information, but it is not necessary for this exercise.
In order to simplify the search for the position of the zero vector, we restrict the method to tetrahedral grids with linear interpolation. Despite numerical concerns, the Isabel hurricane dataset has been converted from hexahedra to tetrahedra.
The proposed approach for detecting the critical points follows the idea of the marching tetrahedra algorithm. First, we look for the zero region of the first velocity component u. This is a planar triangle or quadrangle face inside the tetrahedron in the non-degenerate case (see figure below). We want to restrict the method to triangle faces at least for a first approach. Then we look for the region where u and v are zero. This is a line on the previously found triangle face. Finally we look for the region where all three components are zero. This is a point on the previously found line.



Scheme:

- The dataset is traversed cell by cell
- For optimization, it is tested if the cell contains a zero before searching the zero. This can be achieved by looking at the sign of the velocity components at the four nodes of the tetrahedron. If there can be no zero, the current cell is skipped.
- The signs of u, evaluated in the optimization test, can be used to determine which edges of the tetrahedron contain a zero. The zero positions **r**, **s**, and **t** on the edges can be derived from the equation for linear interpolation between the nodal values.
- The resulting triangle (**r**, **s**, **t**) is treated similarly to the tetrahedron. Instead of using the values at the four nodes of the tetrahedron, we have to interpolate the v component of velocity at the three corners of the triangle first.
- Finally, the procedure is applied to the w component of velocity on the resulting line (**i**, **j**), leading to the desired position **p** of the zero vector.

Implementation

The module will have one input port for accessing the unstructured field and one output port to output the critical points as geometry. The course web page contains a framework module for the exercise. Extract the directory to your project directory.

As a first step, we want to compile the framework module without modifications and test it inside Paraview. First, we have to generate the necessary makefiles:

- change to the extracted FlowTopo directory
- /afs/ethz.ch/users/s/sadlof/scivis_SS06/cmake-2.2/bin/ccmake . (include the dot)
- press 'c'
- an error message says that 'ParaView_DIR' has to be set
- press 'e'
- go to the 'ParaView_DIR' entry
- press enter
- insert: /afs/ethz.ch/users/s/sadlof/scivis_SS06/paraview-2.4.3-x86
- press enter
- press 't', for more options
- go to 'CMAKE_BUILD_TYPE' and set it to 'Debug'
- press two times 'c'
- press 'g' in order to generate the Makefiles

The makefiles have to be generated only once. All subsequent compilations only require a simple 'make' command. The 'make clean' command (removing compilation results) is also supported.

After compilation, the module can be loaded into Paraview as follows:

- Start up Paraview and load the Isabel dataset
- File -> Import Package
- Go to your project directory and open the file FlowTopo.xml
- Filter -> FlowTopo
- "Accept"

Add your flow topology implementation to the file vtkFlowTopo.cxx. The entry point to the code is the RequestData method. There are some 'code snippets' that show how to access the data.

Documentation

VTK Doxygen documentation is available at http://www.vtk.org/doc/release/5.0/html/
The VTK modules can be used as examples. The VTK module source code is located at:
/afs/ethz.ch/users/p/peikert/paraview/paraview-2.4.3/VTK/Graphics

Paraview documentation is not freely available. We have one book. Basically, the actual exercise should not require additional module parameters. If you want to add parameters to the module and hence need the documentation of Paraview, either use the interface files already existing inside Paraview at
/afs/ethz.ch/users/p/peikert/paraview/paraview-2.4.3/Servers/ServerManager/Resources/filters.xml
and
/afs/ethz.ch/users/p/peikert/paraview/paraview-2.4.3/GUI/Client/Resources/Filters.xml
as examples, or send a request to sadlo@inf.ethz.ch.
You would have to add appropriate entries to the files PVLocal.xml.in, PVLocal.pvsm.in, and to vtkFlowTopo.h.

Debugging

- start paraview
- ddd /afs/ethz.ch/users/p/peikert/paraview/inst/lib/paraview-2.4/paraview-real
- inside Paraview, load the FlowTopo module, but do not execute it (do not "Accept")
- inside the graphical debugger ddd:
- File -> attach to paraview-real
- File -> Open Source
- wait
- File -> Open Source -> Load Shared Library Symbols
- wait
- select the source file vtkFlowTopo.cxx
- add any break-points etc.
- continue debugging (click continue)

Apply your flow topology module to the Isabel dataset. Send your source code and an image showing the result to sadlo@inf.ethz.ch.

Additional Notes

We experienced slow startup and compilation times. This may be caused by the access to the AFS file system. If you have the opportunity, you could try to install the necessary files on a different file system. Ask sadlo@inf.ethz.ch in case of problems.