

Introduction to scientific visualisation
Part 1

Petra Heil
Australian Antarctic Division
&
Antarctic Climate and Ecosystems CRC
University of Tasmania
Hobart, TAS 7001
Australia

October 12, 2006

This course is designed to introduce students of environmental science studies (e.g., oceanography, atmospheric sciences, environmental science) to *state of the art* computer-based graphic tools for scientific analysis as well as digital presentation of derived results. The course emphasis is on presenting available software tools and how to use and apply these in academic research where the focus is on computational fluid dynamics. Not all software packages currently available have been included in this course, instead the it is attempted to provide the student with an overview of the various types of software available and their potential benefit within fluid dynamics applications.

Course objectives are to:

- analyse and interpret large data sets derived either from experiments or computer simulations;
- visually summarise multi-dimensional properties (e.g., extrema, eigenvalues) of the data set;
- understand application and virtue of discussed software tools;
- prepare a well-structured project using the student's data set.

Syllabus

Part 1:

- 1. Background and introduction.
- 2. Graphical presentation of data.
- 3. Basic visualisation algorithms.
- 4. Volume visualisation.

Part 2:

- 5. Visualisation systems: a) Matlab.
- 6. Visualisation systems: b) IDL.

Part 3:

- 7. Visualisation systems: c) GMT.
- 8. Visualisation systems: d) Ferret.
- 9. Visualisation systems: e) Vis5d+.
- 10. Other application examples and useful data sources.

Chapter 1

Background and introduction

Visualisation can be grouped into scientific visualisation, program visualisation, and standard 2D visualisation. This course will focus on scientific visualisation, where datasets, often derived from numerical models, are displayed to assist their analysis. Information on program visualisation may be obtained from the Australian Partnership for Advanced Computing¹ [APAC], for example the parallel debugger Totalview² (a product of Etnus).

1.1 Historical background - Theory

In the last decade scientific advances have enabled extended access to large observational data sets (e.g., from satellite-based sensors). The recent improvement of computational power has led to availability of extensive data sets from the output of numerical simulations. Scientists, especially in computational fluid dynamics, now have to deal with the analysis of extremely large and multi-dimensional data sets. Their task is to identify, extract and analyse physically, chemically or biologically meaningful signals, which are contained within the spectrum presented by the data. The development of tools to visualise these large data sets has advanced at a lesser rate than the computational power. The user is still required to provide the visualisation software with knowledge of the parameter range, temporal or spatial bisecting etc. Close-up visualisation of large data sets are still not common.

New techniques for the advanced visualisation of large data sets need to be devised. These need to include the visual presentation, procedural representation and data mapping, and should be cost-effective and time efficient. While multi-processor

¹<http://nf.anu.edu.au/facilities/>

²<http://nf.anu.edu.au/facilities/software/totalview.php>

machines are available these new visualisation techniques need to be scale-able to work over a range of computing power and bandwidth, including single-processor computers, such as labtops.

It is also important to consider that the impact of any visualisation is based on how it is perceived by the viewer. "Data visualisation is a joint function of computer graphics and perception" [M. Green, 1999].

Chapter 2

Graphical presentation of data

Currently a wide range of research into data visualisation is carried out to:

- Derive the *signal to noise* ratio of the input data via a procedural representation consisting of implicit models.
- Detect meaningful features in large, detailed spatial data using topological operators and separatrix curves and surfaces.
- Adapt the procedural representation to the appropriate level of detail by using multi-resolution techniques based on multigrid methods.
- Provide data specific knowledge as metadata to explore these extremely large datasets at the feature level.
- Visualise the data directly from the procedural representation, using and extending numerous existing visualisation techniques (e.g. isosurfaces, contouring, data slicing, volume rendering, line-integral convolution)
- Track the level of accuracy of the procedural representation by taking account of errors through digitising, modelling, reconstruction, and visualisation.

One such project is *National Partnership for Advanced Computation Infrastructure's* (NPACI) Scalable visualisation¹, which is a tool set for the investigation of very large multi-dimensional data sets, and which is designed to process data sets that are too large to fit into memory and swap space of the accessible computing system (laptop to multi-processor machine). The NPACI's software allows this by supporting data paging on demand in and out of the memory. NPACI states that the software is configured to be thread-safe and to perform well on multi-processor computers.

¹<http://www.npaci.edu/>

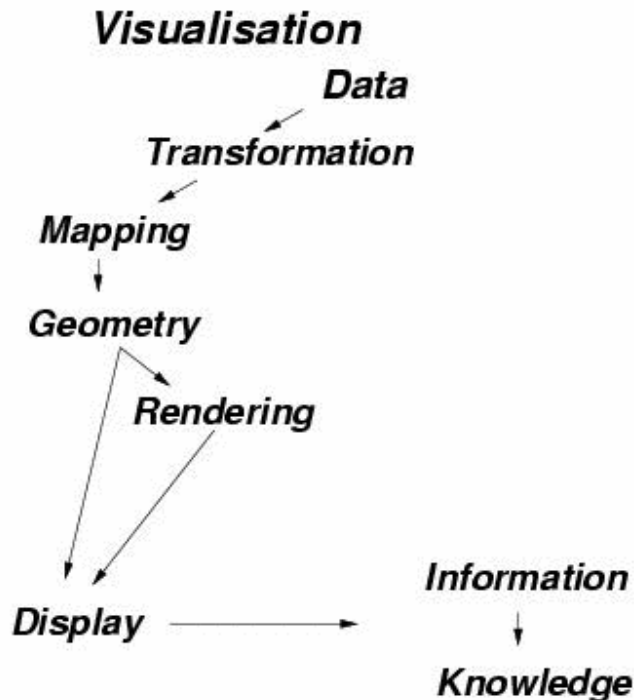


Figure 2.1: Schematic diagram of the processes involved in computational visualisation.

2.1 Definition of visualisation

Visualisation is an important tool to support the scientific interpretation and the presentation of numerical data resulting from computational simulations or measurements. There are several goals to scientific data visualisation: It is used to display data or parts of multi-dimensional data in a way to allow the researcher to explore the data set and to identify patterns or changes of scientific information contained in the data. The researcher might employ different visualisation techniques to uncover a signal within the data before applying other techniques to analyse (e.g., *Fourier* transformation) the signal. Data visualisation is also used to present data and derived conclusions to an audience. This often involves a reduction of the data to highlight aspects of the often large data set.

2.1.1 The visualisation process

The actual process of visualising data may be broken down into the following steps:

Data transformation

- Geometric transformation
E.g., scaling.
- Attribute transformation
E.g., vector analysis of the data set.
- Topological transformation
E.g., transformation of polygonal-type data into grid data.

Data mapping

Before a data set can be displayed the data points need to be remapped to a grid. A number of algorithms are available:

- The *scalar algorithm* works on scalar data by generating isosurfaces, contour lines etc.
 - Contour lines:
2D contouring produces contour lines, 3D contouring produces isosurfaces.
 - Colour mapping:
Association of colour with different scalar values.
- The *vector algorithm* works on vector data, e.g. in a display of arrows to express a flow field.
- *Tensor algorithms* work on tensor data.
- *Model algorithms* output a textured, topological or geometric data set of the original data.

2.1.2 Tasks of data visualisation

Data visualisation consists of a string of necessary tasks:

- Deciding on a visualisation system.
- Data input into the visualisation system.
- Choose a visualisation style for the representation of the data.
- Select an appropriate output format (e.g., image file, hard copy, movie file, or rendering).

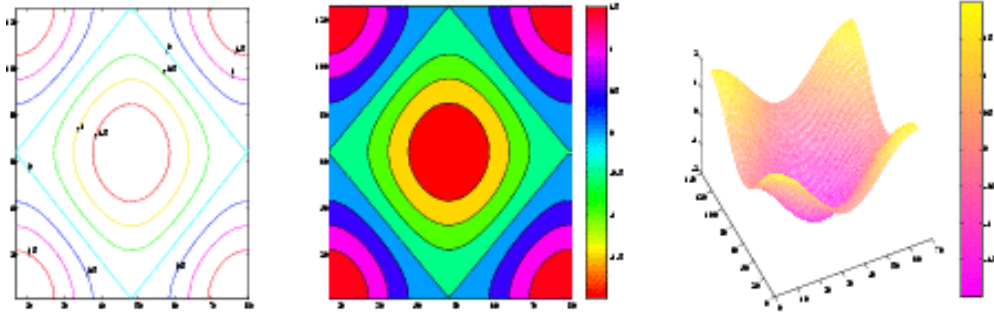


Figure 2.2: Methods of visualising three-dimensional data: Isolines (contour lines; left), filled contours (middle), three-dimensional view (right).

2.2 Visualisation spaces

Depending on the dimension of the computational domain and the dimension of the visualisation space a data may be displayed in various ways [Schroeder et al., 1998]. The following visualisation techniques are commonly used:

Dimension of the visualisation domain

	Dimension of the computational domain			
	1D	2D	3D	multi D
0D		Dot Lines	Dot surfaces	
0D	Points	Particle points	Particle traces	
0D		Scatter plots	Scatter plots	
1D	Lines	Vector plots	Vector plots	
1D	Curves	Contour plots	3D vector diagrams	
2D		Elevation plots	Tiled maps	Attribute mapping
2D		Contour plots	Texture maps	
2D		Images		
3D			Volume rendering	Attribute mapping
3D			Visual Modelling	

In fluid dynamics, the visualisation of multi-dimensional parameter fields both scalar and vector fields is important. The three-dimensional visualisation is based on regular or curvilinear grids. Vector fields can be visualised by a number of

techniques including particle tracing, interactive probing, and selective visualisation. Data fields derived from climate models are generally displayed as isolines (contours) of a physical or chemical parameter within a two-dimensional sections of the grid (??, ??). Similarly vector quantities are visualised as streamlines or arrows. With increasing computer power interactive multi-dimensional visualisation packages (??, ??) have become available to allow unlimited exploration of multi-dimensional data fields.

Reference:

Schroeder, W., K.M. Martin and W.E Lorensen, The Visualization Toolkit, 2nd Ed., Prentice Hall Inc., 645pp., ISBN 0-13-954694-4, 1998.

2.3 Rendering

There are two algorithms for data rendering:

- *Image order*
 - Rendering takes place by ray tracing.
 - Volume rendering is carried out via ray tracing or ray casting, and generally applied to translucent surfaces, clouds or haze. This is done without any intermediate conversion of the multidimensional data onto the two-dimensional display surface.
- *Object order*
 - Rendering takes place by object.

2.4 Light and colours in visualisation

Sun light is made up of a spectrum of wave lengths in the visible band of the wave spectrum. Colour is perceptive to each viewer. Artificial lighting, including computer monitors and HTML colour schemes, use models to recreate the natural colours.

The light source:

All through this course a simple light model, with the source of light placed at infinity, is applied.

2.4.1 The RGB scheme

There are three different colour receptors in the human retina, implying that only three numerical colour components are necessary to describe any naturally occurring colour. Based on the primary colours (red, green and blue) the following *additive* colours are available in the RGB scheme:

red & green & blue	=	white
red & blue	=	magneta
green & blue	=	yellow
red & green	=	orange

2.4.2 The HSV scheme

The HSV scheme is an additive colour scheme based on the Hue (attributes of the colour), Saturation (contribution of white) and Value (intensity or brightness). The HSV colour model is based on a cylindrical coordinate system, where the HSV scheme occupies a cone. Hue is measured as an angle around the vertical axis, where red resides at 0°, green at 120°, yellow at 240°. Saturation is a ratio between 0 (focusses on the Value axis) to 1 (on the triangular sides of the cone). Value presents the cone height, and ranges from 0.0 (black) to 1.0 (white).

2.4.3 The CMY scheme

In the CMY or CMYK scheme a *subtractive* mix of Cyan, Magneta and Yellow (and black) is used to generate colours. This colour scheme is used for printing, painting and other traditional arts. The colour wheel is made of three colour classes:

Primary colours	Secondary colours	Tertiary colours
cyan	indigo	blue
magneta	red	purple
yellow	green	rose
		lime
		orange
		aqua

2.4.4 Visual perception

Visualisation relies heavily on perception. The way any colour scheme is perceived varies with:

- the size of an object;
- the relative distance between neighbouring objects;
- the ambient light;
- the background colour scheme;
- the vision strength of the viewer.

The luminance equation [Pavlidis, 1982] describes how the ambient intensity (and hence the visual contrast) may be calculated from the diffuse colour using the RGB colour scheme:

$$L = 0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B$$

Reference:

Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Maryland, 416 pp., 1982.

2.5 Definition of visualisation tools

Defining the coordinate systems

It is necessary to clearly define the coordinate systems generally used in computational visualisation, and to carefully distinguish between them.

- Model coordinate system:
Surface on which the model is defined.
- Real world coordinate system:
Space in which the real processes take place.
- View coordinate system:
2-dimensional slice taken from a viewpoint within the model coordinate system.

- Display coordinate system:
For pixel values on the display unit.

Defining the graphical tools

- Point
- Line
- Polyline
- Polygon

Chapter 3

Basic visualisation algorithms

Next we will discuss effective means to analyse large data sets. For this, one may employ geometric techniques to extract objects (such as curves, surfaces, solids) from their data.

3.1 Feature extraction

"Features are defined as geometric and topological patterns of interest in a part model and which represent high level entities useful in part analysis."

[Henderson and Anderson, 1984]

Automatic extraction, abstracted into algorithms, is provided by feature extraction. The feature-based techniques present a precise evaluation and comparison of the data [Silver and Zabusky, 1993]. With it any object or pattern of interest may be identified and displayed as a feature within the data, while excessive, and potentially confusing data are excluded. The employed algorithms will extract the feature from the data, and characterise it by its quantitative attributes. Feature extraction reduces the storage requirements (1 - 7% of the solution size [Silver and Zabusky, 1993]). The chief disadvantage of feature extraction is that solutions outside the extracted domain are not available for analysis once the original data set has been abandoned.

Identifiable features:

- **feature size** (derived from surface, perimeter, area or volume; function of pixel count)
- **feature shape** (derived from border characteristics using shape measures, Fourier descriptors, invariant moments or edge steepness)

- **feature colour** (derived from colour description and optical properties)
- **feature texture** (derived from appearance differential to neighbouring cells using statistical geometric features or covariance matrixes)
- **statistical distribution properties**

To complete the scientific analysis, feature extraction must be followed by feature tracking, occurrence identification and visualisation.

References:

Henderson, M.R. and D.C. Anderson, Computer recognition and extraction of form features: A CAD/CAM link, *Computers in Industry*, **5**, 329–339, 1984.

Silver, D., and N.J. Zabusky, Quantifying visualizations for reduced modeling in nonlinear science: Extracting structures from data sets, *J. Visual Communication and Image Representation*, **4**(1), 46–61, 1993.

3.2 Maximum Cross-Correlation [MCC] Method

The MCC method employs sequential data slices (e.g., optical imagery) to compare the evolution of features between the two slices. The method requires the identification of a *template* window in one data slice and a larger *search* window in the second data slice. The size of the former window needs to fulfil requirements for statistical confidence, the size of the latter window needs to be large enough to accommodate the maximum translation speed of the analysed feature. The *template* window is moved within the *search* window and cross-correlation function for each incident are calculated. The indent with the maximum cross correlation is taken as the most probable feature translation [Emery and Thompson, 1997].

Reference:

Emery, W.J., and R.E. Thompson, Data analysis methods in physical oceanography, Pergamon Press, 634pp., 1997.

Chapter 4

Review of tools

4.1 Data types

Binary: Machine readable form

"byte *"	:	8 - bit integer, 0 = _i 255	"unsigned character"
"short *"	:	16-bit integer	
"long *"	:	32-bit integer	
*	:	unsigned binary integers	

ASCII: Programmer readable form

(American national Standard Code for Information Interchange)

"integer"	:	0, 1, 2, 3, ..., N (N: hardware dependent)
"floating point"	:	0.0, 1.0, 123.4, 0.1234E+3 etc.

HDF¹: Hierarchical Data Format is a multi-object file format for the transfer of graphical and floating-point data between computers that was created at the National Center for Supercomputer Applications (NCSA).

netCDF²: network Common Data Format is another similar to HDF sponsored by the University Corporation for Atmospheric Research (UCAR).

4.2 Fortran program for ASCII to HDF conversion

Information on HDF³ is available through NCSA⁴.

To create a 64 binary HDF file (example.hdf) from an ASCII file (example.ascii) type at the Unix command line:

```
> f90 m_sds_hdf.f -o m_sds_hdf.x -ldf  
> m_sds_hdf.x
```

³<http://hdf.ncsa.uiuc.edu>

⁴<http://hdf.ncsa.uiuc.edu/doc.html>

```

      program m_sds_hdf
c
c  creates an HDF file from an ASCII integer file
c  for more information on subroutines and HDF
c  contact the NCSA at University of Illinois at
c    address:  NCSA Documentation Orders
c              152 Computing Applications Bldg.
c              605 East Springfield Ave.
c              Champaign, IL 61820
c              Phone: (217) 244-0072
c
c    Anonymous ftp file server name: ftp.ncsa.uiuc.edu
c
      integer image(64,64,62)
      real val(64,64,62)
      integer shape(3),ret
      integer DFSDsetdims
      integer DFSDputdata
c
      open(5,file='fan_64.ascii',status='old',err=88)
c
      shape(1)=64
      shape(2)=64
      shape(3)=62
c
c  read in integer values from input file
c
      ix=shape(1)
      iy=shape(2)
      iz=shape(3)
c
      read(5,10)((image(i,j,k),i=1,ix),j=1,iy),k=1,iz)
      10 format(20(1x,i3))
c
c  change mode from integer array (image)
c      to floating point array (val)
c
      do 20 k=1,iz
      do 20 j=1,iy
      do 20 i=1,ix
      val(i,j,k)=image(i,j,k)
      20 continue
c
c  write val to an HDF file
c
      ret=DFSDsetdims(3,shape)
      ret=DFSDputdata('fan_64_sds.hdf',3,shape,val)
c
      if (ret .ne. 0)then
      write(*,*)'Error Writing HDF File'
      endif
c
      88 stop
      end

```

Copyright Notice

All information and data (including graphics) provided by the University and its staff on this Web server are, unless otherwise noted, copyright by the University of Tasmania, Australia. Unlimited distribution of University copyright material is permitted, but only if textual and graphic content is not altered and the source is acknowledged. The copyright in other material found on this server may be owned by other people, and you should get permission from them before distributing their material. Copyright 2004, University of Tasmania.