Distributed Architectures for Environmental Visualisation Systems

BAUDOUIN RAOULT, BRIAN NORRIS, JENS DAABECK[1]

RICARDO CARTAXO MODESTO DE SOUZA, GILBERTO CÂMARA[2]

[1]European Centre for Medium-Range Weather Forecasts - ECMWF

Shinfield Park, Reading, Berkshire RG2 9AX, England

{baudouin, norris, daabeck}@ecmwf.int

[2]Instituto Nacional de Pesquisas Espaciais - INPE

Av. dos Astronautas, 1758, São José dos Campos (SP), Brazil 12227-001

{cartaxo, gilberto}@dpi.inpe.br

Abstract. This paper presents an architecture for a distributed environment, targeted at supporting the development of systems for retrieval, manipulation and visualisation of large environmental data sets. The proposal has been used as a basis for the development of METVIEW, a system for meteorological and climatological data, used operationally at the European Centre for Medium-Range Weather Forecasts and Brazil's Centro de Previsão do Tempo e Estudos Climáticos (CPTEC).

Keywords. Visualisation systems, environmental applications, distributed graphical architectures

1 Introduction

This work presents a distributed architecture for environmental visualisation systems. Our proposal is based on the idea of *service-oriented architectures*, which allow for the combination of different services (such as data access, data manipulation and visualisation) on a single environment.

Most current visualisation systems, such as AVS and KHOROS (Rasure and Williams, 1991), are based on the "data-flow metaphor". This metaphor is closely linked to the idea of a *visualisation pipeline* (pre-processing, feature selection, geometric modelling and rendering). These systems provide a large set of individual specialised modules, which can be combined by the user to perform a complex operation, by means of a visual programming interface. These systems also enable the user to add new functions, by means of extension mechanisms. For examples of extensions to KHOROS, see Barrera, Banon and Lotufo (1993).

Although very flexible and modular, this class of systems has some disadvantages: the *data access* facilities are typically limited to file handling (data management has to be provided by the user) and the sheer number of functions creates a *semantic gap* for the user without a Computer Graphics background (e.g. what does "display pixmap" mean?).

Our proposed architecture presents two important differences in comparison with existing scientific visualisation systems such as KHOROS, AVS and Data Explorer:

- Integration with data base management systems for data access and management.
- A communications protocol which simplifies process distribution on a network.

In practice, the architecture described herein has been used as a basis for the development of METVIEW, a system for retrieval, manipulation and visualisation of meteorological and climatological data. METVIEW is a joint development of the of the European Centre for Medium-Range Weather Forecasts (ECMWF) and Brazil's Centre for Weather Prediction and Climate Studies (CPTEC), with participation from the French Weather Service (Météo-France).

The paper is divided into two parts. Initially, our conceptual design of distributed environmental

visualisation systems is outlined, and we later show its use in the implementation of METVIEW.

2 Conceptual Design

In this section, we present the general ideas behind our conception of distributed graphics systems for environmental visualisation.

2.1 General Considerations

Distributed architectures are becoming an important alternative to client-server and file-based architectures for graphical systems.

The growing power of individual workstations has blurred traditional distinctions between client and server machines. Therefore, sharing processes on a network is a more efficient way of using the potential of current hardware platforms than designing large, single-process systems. For that reason, this type of systems are also called *"service-oriented architectures"*, by way of contrast of with *client-server* environments.

In a *client-server* architecture, the process rôles are clearly specified: the client processes receive user commands and execute applications, which request data over the network from a server process, by means of SQL queries or NFS calls.

In *service-oriented* environments this distinction between clients and servers is removed: all processes are *service providers.* The idea is each service providers is specialised on a specific task, and that the tasks may be distributed over different machines.

2.2 System Design

The architecture's heart is the interoperability layer (also called *"request broker"*), that provides communication between services. For environmental visualisation applications, the system should also provide:

- An iconic based user interface, which appropriate metaphors.
- A visualisation module, which implements a fixed rendering pipeline, since the application domain is known beforehand.
- Image and Graphics animation modules.
- Data Access module, which communicates with the data base management system.
- Data Manipulation module, by means of a macro language.

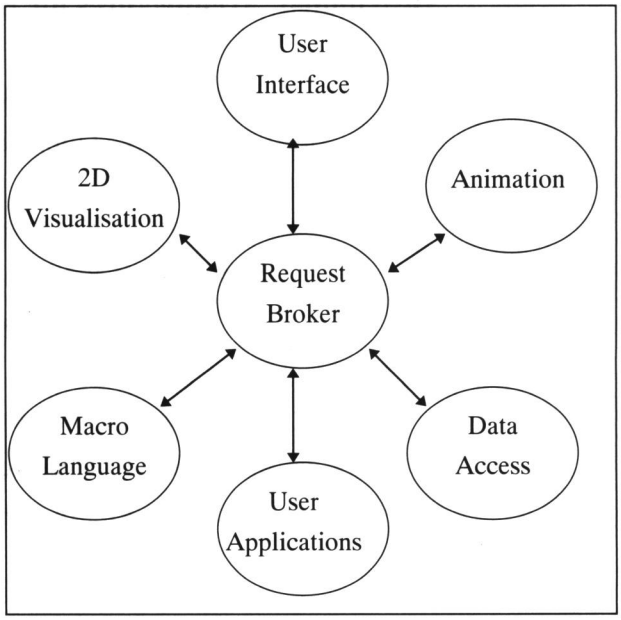

Figure 1 - Distributed Architecture for Visualisation

Our design also allows for extensions: by means of a simple programming interface, the users can add new functions to the system.

2.3 Request Broker

A request broker implements an *object model,* a representation that rises above the limits over a single program or language, and is responsible for:

- Matching client requests and available services.
- Translation and transfer of parameters and objects between different machines and processes.

The most important decisions on the implementation of a request broker are the *object model* and the *communication protocol* used. Current technological alternatives include competing proposals such as CORBA (Common Object Request Broker Architecture, proposed by a consortium that includes IBM, Sun, HP and Digital) and OLE (Object Linking and Embedding, championed by Microsoft).

These proposals are incompatible with each other. Furthermore, even though specifications such as CORBA define a distributed standard, some of its implementations (like IBM's SOM) operate in a single address space (Campagnoni, 1994).

An even more serious question is the "original sin problem": standards such as SOM and OLE are being

proposed as a means of supporting the idea of "compound documents", typical of office automation applications, and there are doubts about their scalability for complex and large environmental data sets. There is a significant difference between encapsulating a 2Kb drawing from a CAD program into a text editor, and transferring a 200 Mb satellite image from an image processing application into a map generation procedure.

Therefore, we propose that the request broker be implemented on a simple and straightforward manner. We chose to implement the "request broker" (RB) using a simple and user-configurable protocol and to write it in ANSI C language, using standard RPC calls to ensure maximum portability.

Our RB has no "internal intelligence". At initialisation time, the process reads a configuration file, which indicates:

- the services available;
- the commands associated to each service;
- how to run the modules associated with each service.

The communication between processes is purely asynchronous and all processes are clients and servers simultaneously. Since the modules communicate using the TCP/IP protocol, they can be run on different machines.

The object model, described below, is based on the idea of "self-describing data sets". The RB process does not perform any decoding of objects, but is limited to passing parameters and data among the processes.

2.4 Distributed Object Model

Objects used in environmental visualisation applications consist of three main groups: point observations, 2D and 3D files (scalar and vector fields). The fields include satellite imagery and numerical weather prediction results.

Our object model is based on the idea of "self-describing data": each data object exchanged between the applications contains information which enables the receiving service to decode its contents without having to resort to any additional file.

We feel that self-describing data sets simplify the design and implementation of the request broker, since the additional space needed is small, relative to the data size.

There are currently various proposals of formats for self-describing scientific data, which include Net/CDF (Rew and Davis, 1990). Net/CDF is a more complete proposal, which consists both of a *data format* and a *decoding library*, freely available.

In the meteorological field, the World Meteorological Organisation (WMO) has defined the GRIB (2D fields and images) and BUFR (observation) formats., which were used in the implementation of METVIEW.

2.5 Communications Protocol

The typical user of an environmental visualisation system follows a direct processing pipeline: *data retrieval* based on a query request, *data transformation* by mathematical functions and *data presentation* (2D, 3D or animation).

In line with our ideas of a clear design, we propose a simple, yet powerful protocol, based on a language with the following abstract syntax:

```
COMMAND,
   PARAMETER1 = VALUE,
   PARAMETER2 = VALUE1/VALUE2,
   PARAMETER3= VAL1/TO/VAL2/BY/VAL3
```

We consider that this protocol is general enough to portray the batch-oriented nature of visualisation requests.

For example, a data retrieval command may be written as:

```
RETRIEVE,
    PARAM = IMAGE,
    TYPE = METEOSAT5,
    BAND = IR1,
    DATE = 860125,
    TIME = 14Z.
```

This command requests an METEOSAT satellite image (first infrared band), obtained at 14:00 Utc in 25 January 1986.

2.6 Data Access

One key issue which is usually not considered in most scientific visualisation systems is the problem of data access: data is supposed to be available at the user's workstation. Whilst this approach may be acceptable in some cases (such as biochemical applications), it is not compatible with the large and complex data sets typical of environmental centres.

Data for environmental applications typically consists of numerical weather forecasts and climate

prediction (archived as grids or spectral coefficients), satellite imagery and observational data.

Centres which handle environmental data are accumulating data at staggering rates. ECMWF MARS archive currently holds 15 years of meteorological data (10+ Tera bytes), on various media including automatic cartridge robots.

Such massive quantities of data cannot be handled effectively by traditional relational DBMS, operating on client-server mode. We also note that SQL queries are not supported in most current mass-storage systems. In many environments, it is not even feasible to query the availability of data or to browse its contents.

Our architectural design envisages a multi-level hierarchy, based on the notion that most environmental visualisation applications are "read-only": data is read from a large archive and manipulated before graphical presentation. When a data request is issued, the system will search through a hierarchy of data repositories:

- user files on a local workstation (typically 1~10 Gb).
- files in one or more network servers, managed by an RDBMS (optionally, the files may be stored as long fields within the RDBMS) (typical values: 10~100Gb).
- files on data servers, based on massive data handling technology such as CFS (Common File System) or UniTree (1~1000Tb).

2.7 Data Visualisation

In dealing with environmental data, 2D visualisation and animation are still widely used, but 3D (Hibbard et alli., 1994) visualisation techniques are being more and more in demand by the user community.

The main problem in building 3D visualisation modules for environmental data is displaying the temporal evolution of such phenomena. In other words, there is no practical use in presenting environmental data as static 3D pictures. Fast interactive animation, showing more than one variable simultaneously (the so-called "5D paradigm") is mandatory for 3D visualisation.

2.8 Extensibility Issues

It is extremely important, in any graphical system, to allow the user to provide extension and therefore add functionality to the system.

In our proposal, the user should develop his application following a well-defined protocol. This protocol is a set of procedures which establish communication between the new application and the request broker. These procedures include:

- registration of the new module to the request broker;
- a "service" callback function, which associate a command name that the module must execute with the actual application.
- a "reply" callback function, which is called when the module is completed.

The user also needs to create a file which contains information for the request broker (which service is provided and how to run it). This file should also include the definition of the parameters used by the application, what are their valid values, their default values, a set of rules for performing simple consistency checks, and a meaningful icon which represents the application.

This file is then included in the main *system configuration file*, and the main user interface will create the various items required, such as menus and input fields.

3 Implementation of METVIEW

In what follows, we describe the implementation of the proposed architecture in the METVIEW system. METVIEW is an interactive meteorological visualisation application, which runs on UNIX workstations, using TCP/IP sockets as its communication protocol, and its user interface is Motif-based.

The METVIEW graphics visualisation is based on the ECMWF MAGICS package (O'Sullivan, 1993) and the supported data formats are WMO GRIB format for fields and images, and WMO BUFR format for observations.

3.1 User Interface

An important requirement for user interfaces for visualisation system is the need for powerful abstractions. For a general discussion on the issue, see Freitas and Wagner (1994).

The general philosophy of the METVIEW interface is that the user creates *objects* and performs operations on them. These objects are instances of the various classes of services available on the system.

We consider a typical visualisation request to consist of a combination of :

- a *data retrieval* definition (consisting of one or more fields, observations or images);

- a *data transformation* definition(consisting of a mathematical formula to be applied to the data set).
- *a visual definition* (which describes how the data unit is presented);
- a *plot window* definition (indicating geographical area and cartographic projection).

As outlined in Figure 2, the interface uses the "drag-and-drop" metaphor: by dragging the *data retrieval object* together with a *data transformation* definition and a *visual definition* into a *plot window*, the user instructs METVIEW to perform a series of actions, including data retrieval, data manipulation and visualisation. A general view of METVIEW's interface is shown in Figure 3 and a typical plot window is portrayed in Figure 4.

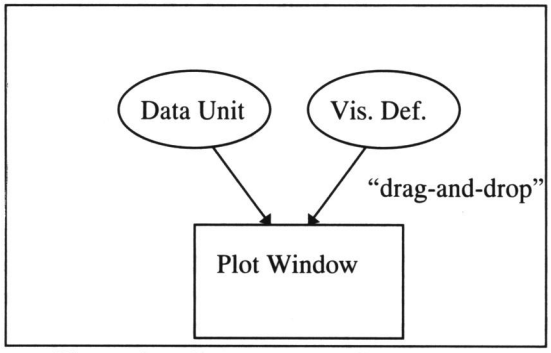

Figure 2 - "Drag and drop" metaphor

Each object definition can be modified, by means of a context-sensitive editor which is associated with all icons in the interface.

3.2 Applications

The current METVIEW version provides the following basic modules:

- *User Interface.*
- *Data retrieval* from a hierarchical archive (or from a local file).
- *Data manipulation* by mathematical formulas.
- Execution of a program written in METVIEW's *macro language.*
- *Visualisation and animation* of observations, images and fields (combined).

In addition, the following meteorological applications are available:

- *Cross section*: vertical cross-sectional plot of fields (figure 4 bottom left).
- *Metgram*: displays the evolution of a forecast at selected points in time.
- *Tephigram*: plot soundings from observations or from fields at selected points.
- *Vertical profile*: graph for the vertical profile of upper air fields.
- *Average plot*: vertical cross section of averaged upper air parameters (zonal or meridional).
- *Colour wind*: displays wind fields coloured by a temperature field.
- *Relative humidity*: computes relative humidity from temperature and specific humidity.
- *Total rain*: computes total precipitation (convective precipitation added to large scale precipitation).

3.3 Data Manipulation

Data Manipulation requirements vary greatly from system to system, and may include 3/4D modelling. A core minimum which is assumed to be typical of most meteorological, environmental and remote sensing centres is the ability to perform mathematical and thematic operations on fields and images.

METVIEW provides two facilities for data manipulation:

- the possibility of performing a mathematical operation on data being retrieved (the COMPUTE command);
- a macro language, with facilities for both interactive and batch interfaces.

When a COMPUTE request is issued, it contains a formula expressed as a text field. The formula should contain a list of items such as:

```
COMPUTE,
    FORMULA     = 'CP+LSP',
    FIELDSET    = precip
```

This formula indicates the calculation of a total precipitation field, which is computed based on the sum of a convective precipitation and a large scale precipitation fields. The data manipulation module understands arithmetic, trigonometric and exponential expressions. It will perform implicit loops when the formula applies to more than one field.

4 Project Status

METVIEW version 1.2, described in this paper, has been delivered and is operational both at ECMWF and INPE. For further references to METVIEW, please see Daabeck, Norris and Raoult (1994).

METVIEW 1.2 consists of various software components:

- Request broker: 1,000 lines of code (LOC) in C.
- Services, including *user interface, meteorological applications, visualisation, utilities and macro language interpreter*: 62,000 LOC in C++.
- Class libraries, used by services: 36,000 LOC in C++
- Low-level C library for interprocess communication routines, language processing, database and file access, free-form manipulation: 20,000 LOC in C.
- Libraries for graphics (MAGICS) and data handling (GRIB and BUFR): 80,000 LOC in FORTRAN.
- C++ interface to MAGICS (5,600 LOC in C++).

METVIEW is being developed at ECMWF and INPE with contributions from Bruno David, David Dent, Arne Jørgensen, Vesa Karhila, Elisa Nishimura, Brian Norris, Patrick O'Sullivan and Baudouin Raoult from ECMWF, Leonardo Bins, Fernando Mitsuo Ii, Ricardo Cartaxo and Lúbia Vinhas from INPE and Sylvie Lamy-Thépaut from Meteo-France. Jens Daabeck (ECMWF) and Gilberto Câmara (INPE) are the joint project managers.

The concept of the distributed architecture described in this paper and its implementation in METVIEW has been primarily developed by Baudouin Raoult.

References

Barrera, J.; Banon, G.J.F. and Lotufo, R.A. (1994). "A Mathematical Morphology Toolbox for the KHOROS System". *Proceedings of International Symposium on Optics, Imaging and Instrumentation - Image Algebra and Morphological Image Processing*, San Diego, July 1994.

Campagnoni, F.R. (1994). "IBM's System Object Model". *Dr. Dobb's Special Report on Interoperable Objects Revolution*, Winter 1994/95, pages 24-28.

Daabeck, J.; Norris, B.; Raoult, B. (1994). "METVIEW: Interactive Access, Manipulation and Visualisation of Meteorological Data on UNIX Workstations". *ECMWF Newsletter*, number 68, pages 9-28, Winter 1994/95.

Freitas, C. M.S.; Wagner, F.R. (1994). "Tool-Oriented Exploratory Visual Analysis". *Proceedings of SIBGRAPI '94*, pages 197-203. (In Portuguese).

Hibbard, W.; Paul, B.; Santek, D.A.; Dyer, C.E.; Battaiola, A.; Voidrot-Martinez, M.-F. "Interactive Visualisation of Earth and Space Science Computations". *IEEE Computer*, vol. 27 (7), pages 65-72, July 1994.

O'Sullivan, P. (1993) "MAGICS - the ECMWF Graphics Package". *ECMWF Newsletter*, number 62, June 1993.

Raoult, B. (1994). "Workstation MARS: Access to ECMWF's Data Base". *IV Workshop on Meteorological Operational Systems, Reading, England, November de 1993. Proceedings, ECMWF, 1994.*

Rasure, J. And Williams (1991). "An Integrated Visual Language and Software Development Environment". *Visual Languages and Computing*, vol. 2, pages 217-246, 1991.

Rew, R.; Davis, G. (1990). "NetCDF: An Interface for Scientific Data Acess". IEEE Computer Graphics and Applications, vol.10 (4), pages 76-82, July 1990.

Acknowledgements

This project has been partially supported by Brazil's Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), through the RHAE and ProTem/CC initiatives.

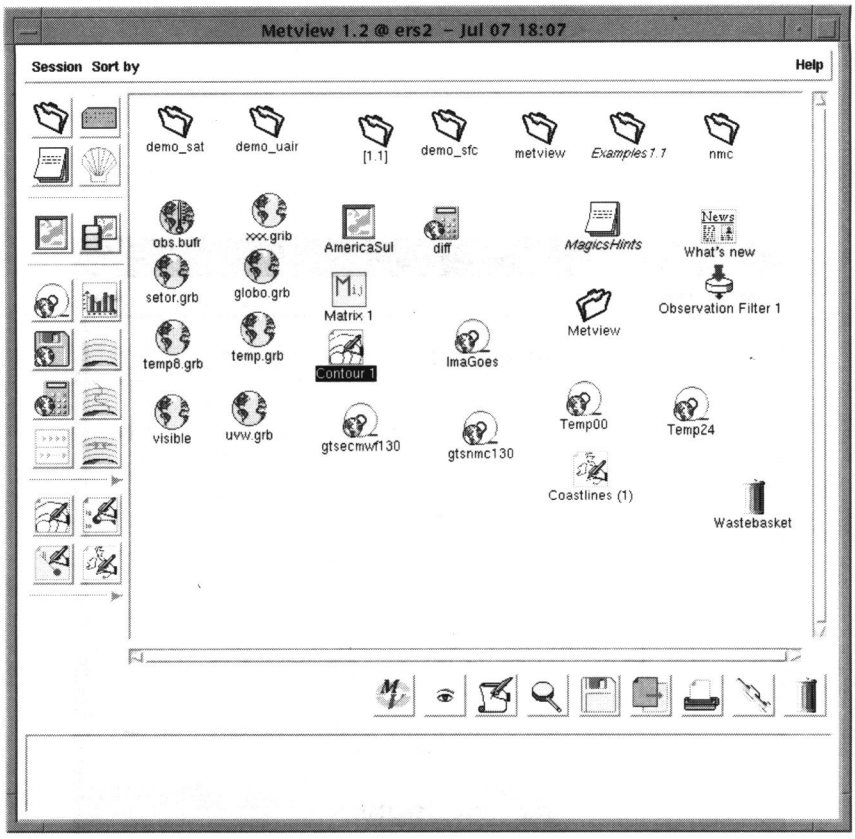

Figure 3 - Example of METVIEW's User Interface

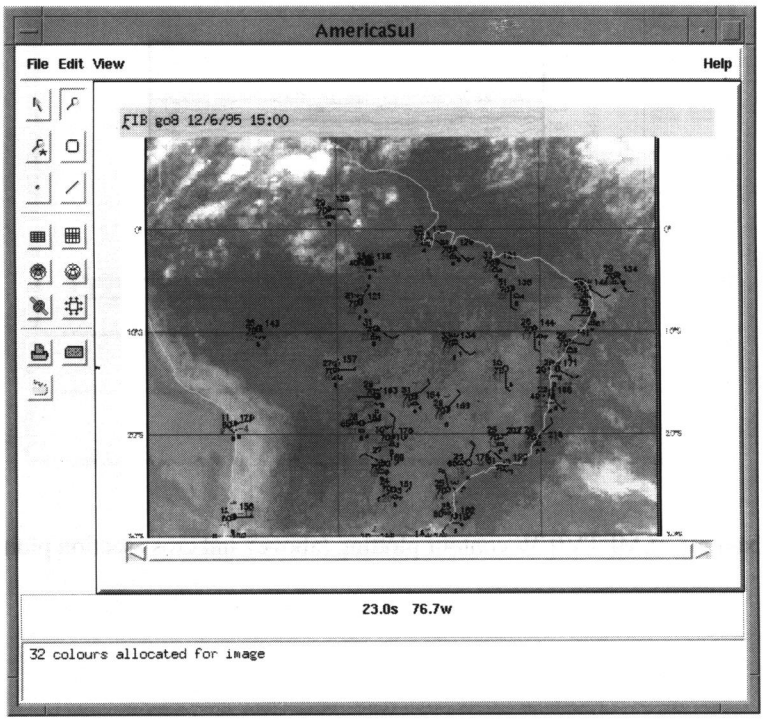

Figure 4 - Example of a METVIEW plot of image and observations

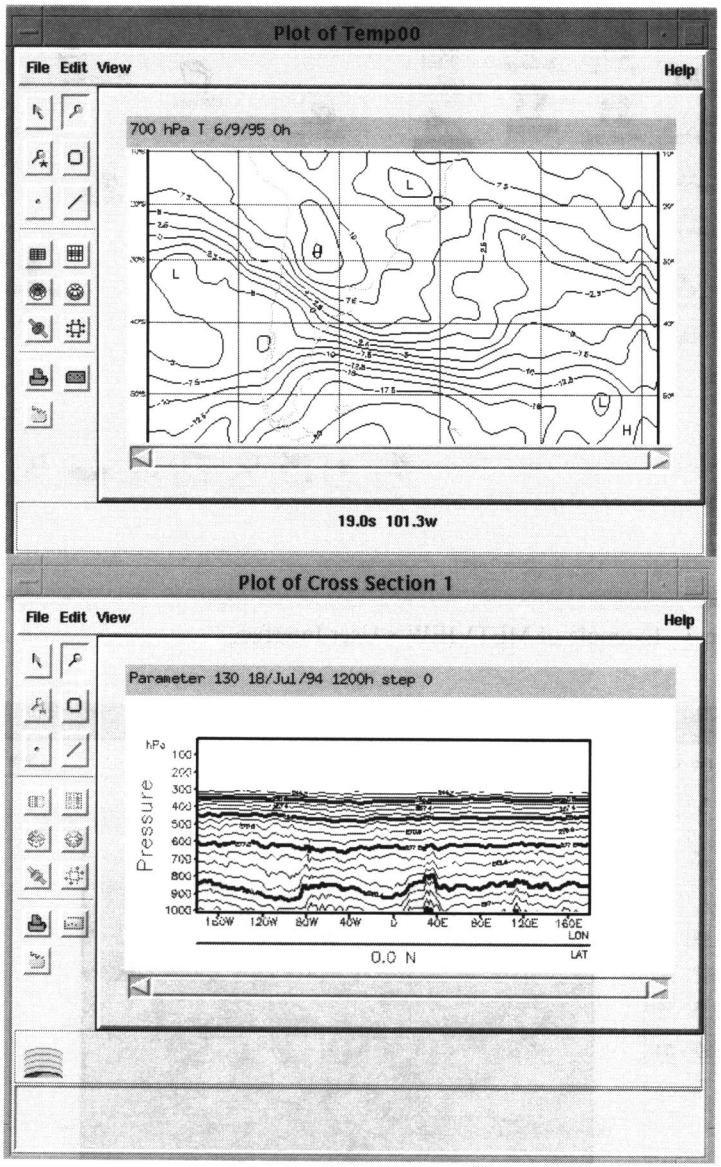

Figure 5 - Example of METVIEW contour plotting (above) and cross section plotting (below)