Tools for Meteorological Data Visualization

ISABEL HARB MANSSOUR
CARLA MARIA DAL SASSO FREITAS
DALCIDIO MORAES CLAUDIO

UFRGS–Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência da Computação - CPGCC
Caixa Postal 15064, 91501-970 Porto Alegre, RS, Brasil
manssour, carla, dalcidio@inf.ufrgs.br

Abstract. Weather forecast centers usually deal with a great volume of complex multivariate data, which usually have to be interpreted within short time. Tools that use scientific visualization techniques can be a great help providing support for both daily forecasting and meteorological research. This work presents a collection of tools designed primarily to aid the daily tasks of meteorologists from the 8^{th} Meteorological District, in the South of Brazil. The tasks of meteorologists and the classes of application data were observed to define tools requirements. System architecture and implementation follow a tool-oriented approach and the object-oriented programming paradigm, respectively. Current implementation provides mapping tools, that generate contour plots, icons maps, and a variety of graphs; recording tools, that allow to save and load images generated by the system; and a query tool, to read variables' values at selected meteorological stations.

1 Introduction

There is a great interest in understanding meteorological phenomena, because of their economic and social importance. These phenomena influence everybody and can seriously affect people and the economy, destroying lives, buildings and plantations. Weather forecast, prediction of the climate and atmospheric phenomena have always been considered as huge scientific computing applications, and soon turned out to be applications with many visualization requirements.

Operational and research centers in weather forecast usually work with a great volume of complex multivariate data, having to interpret them within short time. A graphic system that helps the meteorologists conduct both daily forecasting and meteorological research is very important, and, perhaps, essential, considering the impact of weather conditions on some regions.

The 8^{th} Meteorological District, located in Porto Alegre, collects data from 32 stations distributed in three states in the South of Brazil (Rio Grande do Sul, Santa Catarina and Paraná). These data, called surface observation data, or SYNOP data for short, result from the observation of variables as temperature, atmospheric pressure, wind velocity, and type of clouds, in each one of the 32 stations. Some of these data are acquired by means of instruments while others result from human visual observation. Once received by the 8^{th} Meteorological District (8^{th} DISME), SYNOP data are stored in a local database, and used for drawing (by hand) pressure and temperature contour maps for presentation only. They are also sent to the National Institute of Meteorology, in Brasilia. The 8^{th} DISME receives also satellite images, and other informations from the National Meteorological Center (NMC) and from the National Institute of Meteorology, which together with SYNOP data are used for local weather forecasting [5].

In order to support the tasks done by the meteorologists of the 8^{th} DISME, we developed a set of visualization and query tools, integrating them as a system called VisualMet. Despite the different kind of information received by the 8^{th} DISME, VisualMet deals only with surface observation (SYNOP) data. Integration of other data in the system will be considered later on.

As proposed in a methodology described in an earlier work [3], the tasks of the users (the meteorologists, in this application environment), and the classes of application data were observed to define system requirements concerning visual representations and related visualization techniques, and querying needs. System architecture and implementation follow a tool-oriented approach[4] and the object-oriented programming paradigm, respectively.

This paper presents the tools implemented in VisualMet, providing a brief description of the modeling and functional issues of the system in sections 2 to 4, and focusing on tools implementation in sections 5. Section 6 emphasizes some features of VisualMet, pointing to the related work found in the literature, while some conclusions are drawn in section 7.

2 Data Modeling

As stated above the system was designed using the tool-oriented approach proposed by Freitas and Wagner [4]. This approach uses the object-oriented modeling paradigm representing scientific visualization systems as collections of objects, where some of them are entities and others are tools. Classes of entities are used to model data representing the phenomena under analysis; classes of tools model the processes applied to data either for processing or visualization purposes. The data model diagram from OMT (Object-Modeling Technique) [7] was used for the modeling of entities and tools.

When received by the 8^{th} DISME, SYNOP data are stored as text files. Each file is identified by a name that encode the date and time of the data gathering, thus representing a sample, i.e., a collection of measures such as temperature, atmospheric pressure, air humidity, etc. for all the 32 stations. The 8^{th} DISME receives data three times a day. A separate file maintains information about the location (latitude and longitude) of each station.

In VisualMet, the entities are the samples of SYNOP data, which are modeled in the following way. The **SYNOP entity** is a class that models the samples. Each **SYNOP entity** is an aggregation of class **Sample**, which in turn is an aggregation of a class **Station**. This class has as attributes the position of the station (latitude and longitude), the station code, and one integer array, where the collected data are stored. Instances of the SYNOP entity are presented to the user as a collection named "Entities Base". Actually, the samples are still stored as the original SYNOP files. Once activated, the "Entities Base" is filled up with instances of the class Sample, which are loaded with the data stored in these files. We choose to model in this way because of the structure of the actual data, that must be kept in text files. For further details about the modeling of classes SYNOP, Sample and Station, refer to another work [6].

3 Tools Modeling

Tools are used to process the data in order to derive new data, generate images, or return data values as a result of queries. The tools are modeled as classes and are also presented to the user as a collection named "Tools Base".

Visual representations of SYNOP data are obtained through two classes: the **Map** and the **Graph** classes. Another tool, the Record tool, allows saving and opening image files generated by Map and Graph. The Query tool has the purpose of showing all the information about a selected meteorological station.

The **Map** class implements the exhibition of contour maps and icon maps, that are its subclasses. The **Contour Map** subclass is responsible for the contour lines generation, and the **Icon Map** subclass is responsible for the exhibition of the icons, on each station position, in agreement with the variable selected by the user.

Graph class allows the plotting of bidimensional and tridimensional graphs. The **2D Graph** subclass implements the necessary operations to generate three different kinds of 2D graphs. The first one is a simple bar graph that shows the values of a selected variable in each station in a specific sample, i.e., specific date/time. The second kind of graph is similar to the first one, except by the fact that it presents the values of the variable for three samples. By means of this graph one can observe the selected variable during one day (three samples) in each station. Finally, the third kind of graph is a line graph showing values of a variable per time in a selected station. The **3D Graph** subclass has functionality to generate two different kinds of 3D graphs, the bar graph and the surface graph. The 3D bar graph can be seen as a composition of a group of 2D bar graphs, showing values of variable per sample per time. The surface graph is used to plot the surface formed by a selected variable in latitude x longitude x variable.

The **Record** class does not have subclasses, because of its simplicity. It provides methods for saving and opening image files generated by other tools in the system. The Query tool are not modeled as a class, being part of the Interface class, described in the next section.

4 System's Interface

Besides the entities and tools classes, there is another important class, the **Interface** class, that implements the interaction facilities and provides the communication among entities and tools. The **Interface** class has three subclasses, that basically implement the three main windows existing in the system: Entities Base, Tools Base, and Main Window, as shown in figure 1.

The Interface class manages all the system through an events queue. The events queue stores all the user actions; each action is then identified, and triggers the appropriate function. This function calls the other classes, passing the suitable parameters. In this manner, the operations of all the classes are totally independent of the system interface.

Since the **Interface** class centralizes all the calls to the other classes, from the moment that a window is opened, until the final image is done, all the classes are independent from the data, as well as from the interface. Depending on the selected tool, for example, the methods to read the data, open a window, and generate an image are called by the **Interface** class. In this way, the selected tool receives just the necessary parameters and data to accomplish its function. By this way we implement the mapping between data and specific visual representations [4]. The **Interface** class knows the data structure that must be passed to each tool in the system.

The **Main Window** subclass has only the functions

of showing the help pages and exiting the system. The **Entities Base** subclass manages the list of selected entities and the Entities Base window, drawing the Samples' icons, and setting the selected ones. And, finally, the **Tools Base** subclass is responsible for drawing the Tools' icons, initializes the necessary parameters depending on the selected tool, and opens the window that lets the user to enter the parameters for the tool he/she has just selected.

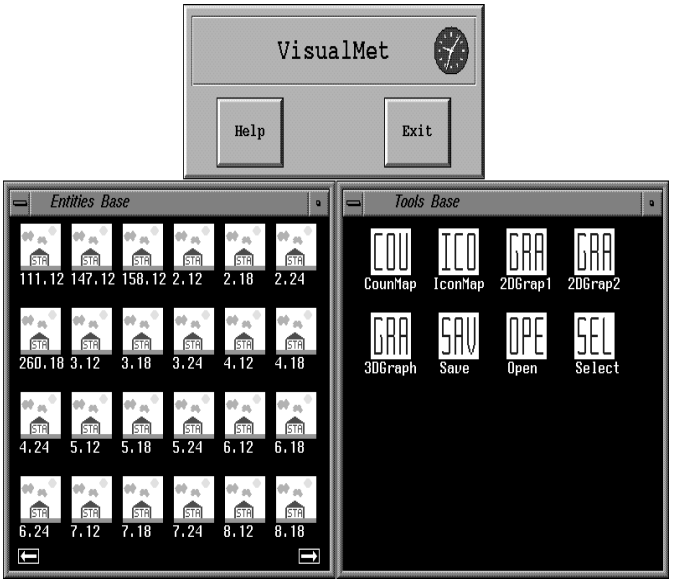

Figure 1: System interface

As to the operation of the system, the user activates a tool by first selecting (using the "point-and-click" technique) one or more entities in the Entities Base, and then pointing to a tool in the Tools Base. Contour and icon maps, query of variables' values in stations, and 2D and 3D graphs are the tools already available.

5 Tools Implementation

The system is being implemented in a Iris Indigo Silicon Graphics workstation, using the programming language C++, the graphical library GL [12] (we plan to use OpenGL, as soon as its availability in our Institution), and the Forms library [9], which is a powerful and simple package for building interfaces.

This section is further divided in six subsections, one for each implemented tool. Other tools that are being designed are briefly commented in the last section.

5.1 Contour Map

The **Contour Map** class generates images (that we call maps) with contour lines obtained through the interpolation of one of the following scalar variables: temperature, maximum or minimum temperature, dew point temperature, atmospheric pressure at station level, pressure at sea level, precipitation and air humidity. When this tool is activated the user has to specify the variable and the desired interval between lines. The contour map is then shown in another window. Figure 2 shows an example of a contour map. As can be observed on this figure the upper left corner of the window shows the latitude and longitude values, according to the cursor position in the window, and the lower left corner shows the value of the displayed variable for each contour line, that are presented in distinct colors. The title bar of the window has the identification of the variable that is represented on the map and the name of the selected entity (that is an instance of Sample class).

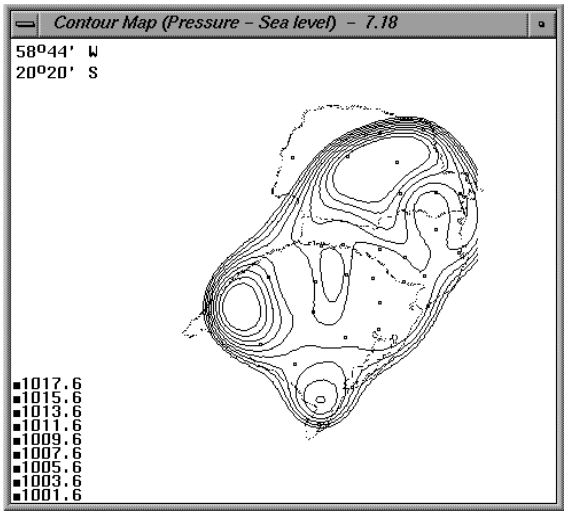

Figure 2: Contour map for atmospheric pressure

SYNOP data can be treated as scattered multivariate data, sampled over a bidimensional domain. The study of several methods used to process these data in order to obtain the scalar field that they represent, has lead us to choose the multiquadric method described by Foley [14, 15]. The obtained interpolant is then used to create a regular grid of data, which is inputted to a plotting algorithm that effectively draws the contour lines. This second algorithm is based on that developed by Dayhoff [8]. Details of both algorithms were reported elsewhere [6].

Contour maps generated by our system (i.e., using the multiquadric method followed by the Dayhoff-based algorithm) were compared with contour lines generated by the *ListContourPlot* function of the *Mathematica* software [13]. Since Mathematica's *ListContourPlot* function is only an alternative to the Dayhoff-based algorithm, the comparison was made by inputting to Mathematica several grids of data generated using the multiquadric method. The contour lines obtained with Mathematica were almost

identical to those generated by the Dayhoff-based algorithm. A small variation occurred because of the definition of the intervals between the contour lines.

An interesting problem found in the implementation of the multiquadric interpolation algorithm is the definition of a value (a constant named R) in the interpolation expression [1]. The interpolant used by this method is shown below:

$$M(x, y) = \sum_{i=1}^{N} a_i \sqrt{(x - x_i)^2 + (y - y_i)^2 + R^2},$$

where $R^2 > 0$ and a_i satisfies the NxN linear system of equations $M(x_k, y_k)$, for k=1,...,N. The R constant strongly influences the final result producing different contour lines in accordance with its value. Experiments done by the author of this method [1, 11] indicated that it depends on the number of data points, the distribution of the (x_k, y_k), and the function values f_k.

The possible solution to this problem was to allow the user to define the R value. But this is very difficult to users that are not used to interpolation methods.

Comparison of actual maps made by meteorologists with our interpolated maps was carried out using the same samples, and several variables. We then conclude that the best value for R is different in accordance with the variable. For pressure at station level and pressure at sea level variables, the value used for R is an approximation of the average of the distance squared between the points:

$$DM^2 = \frac{(x_{max} - x_{min})(y_{max} - y_{min})}{N},$$

where N is the number of points, x_{max} is the maximum value for x_k and the other variables are similar defined. The value used for R for the other variables is DM^2 multiplied by 0.4. By using these R values, some contour lines may be different, but the regions of lower and higher values (temperature, for example) are correctly located for all variables.

5.2 Icon Map

An icon map is an image of the geographical distribution of the 32 stations, with icons at each station representing the values of variables. The variables used in this tool are: wind direction and wind velocity; past weather and present weather, that means if it rained, if the sun was shining, etc.; clouds features, that describes the kind of cloud observed at each station; total clouds, that indicate the quantity of clouds; and atmospheric pressure tendency.

When the user selects the Icon Map tool, he/she is prompted to specify the variable that must be shown. The icons employed in the image belong to a set commonly used by meteorologists of the 8^{th} DISME. Figure 3 shows an example of an icon map with the attribute concerning clouds plotted. The icon for weather conditions regarding clouds is a circle: a full circle corresponds to completely cloudy skies while an empty one corresponds to clear skies. As in the contour map, in the upper left corner of this window the user can see the values of latitude and longitude of the cursor position in the icon map.

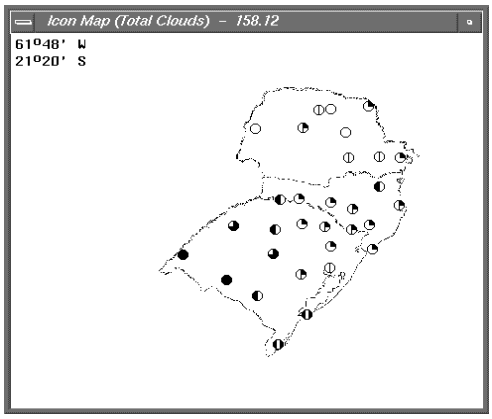

Figure 3: Icon map showing weather conditions regarding clouds

5.3 Query

The Query tool does not have an icon in the Tools Base because it's activated when the user positions the mouse on an icon in an icon map, or on a station icon in a contour map, and presses the left button. Then, a window with all attributes of the selected station shows up (figure 4). If there is no data collected for a station in the current sample, a message indicating the lack of attributes is presented.

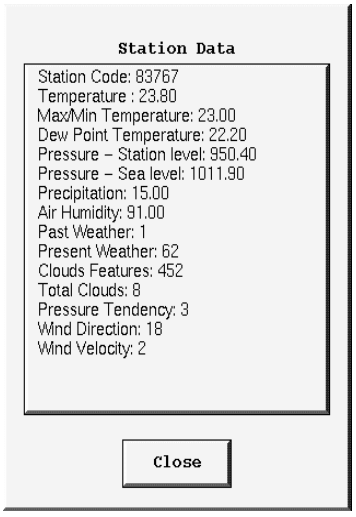

Figure 4: Results obtained by the use of the query tool

5.4 Record

The Record tool can only be used to store images generated by the system, using an internal format. The images (maps or graphs) can be stored at any moment. The user has only to select the window that shows the desired image (pressing the right button of the mouse in that window), and then the icon of the Save tool in the Tools Base. The parameters for this tool are the name of the directory and the name of the file.

To visualize an image previously stored by the system, the user has to select the Open tool in the Tools Base, and enter the appropriate parameters. The image is shown in a separate window. It's important to say that the Query tool can not be used in this window, because the sampled data is not loaded in the memory again. Only the previous generated image is being shown.

5.5 Bidimensional Graph

The 2D Graph tool provides three different types of graph for the following variables: temperature, maximum or minimum temperatures, dew point temperature, pressure at station level, pressure at sea level, precipitation, air humidity, wind velocity and wind direction.

The first type is a bar graph. The user can select an entity and a variable. The Y axis corresponds to the values of the selected variable, and the X axis corresponds to the stations' codes in the selected sample. With this kind of graph it's possible to observe the behavior of one variable all over the stations at the selected sample, which means at a specific date/time. The second type of graph is similar to the first one, but it allows the user to select three entities, thus plotting data from an entire day, for example. The difference from the first one is that three bars are shown for each station in the X axis (see figure 5). The bars are distinguished by colors, each one corresponding to a sample. The meaning of each color is shown on the upper part of the window, while all the stations' codes are listed on the right side of it.

The third kind of graph is a line graph. It allows the user to select a group of samples, one station and one variable, thus showing the variation of the selected variable at a particular station in time. The Y axis corresponds, again, to the value of the selected variable, and the X axis corresponds to the several date/times for the selected station (figure 6).

5.6 Tridimensional Graph

There are two kinds of 3D graphs available in the 3D Graph tool: bar graph and surface graph. The 3D bar graph enables the observation of a variable in the ways allowed by both the 2D bar graph and 2D type three graph. That means, the 3D bar graph is a group of 2D bar graphs shown along a third dimension that is the time. The user has to select a group of samples and a variable. The X axis corresponds to time (date/time of each sample), the Y axis corresponds to the selected variable, and the Z axis carries on the stations' codes. By means of this graph the user can analyze the behavior of one variable, for all stations, in a period of time.

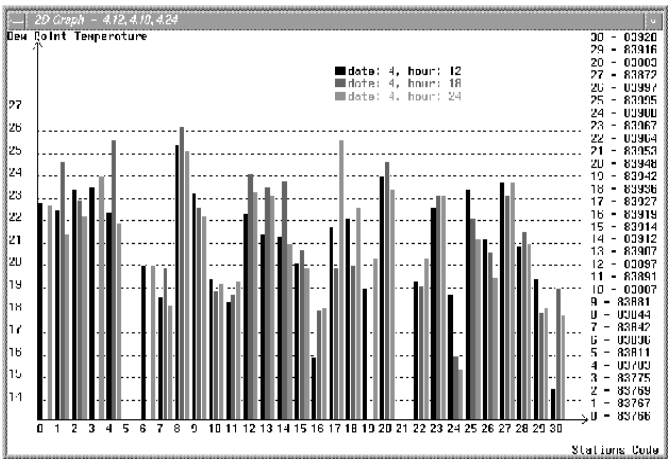

Figure 5: Bar graph of dew point temperature (three samples)

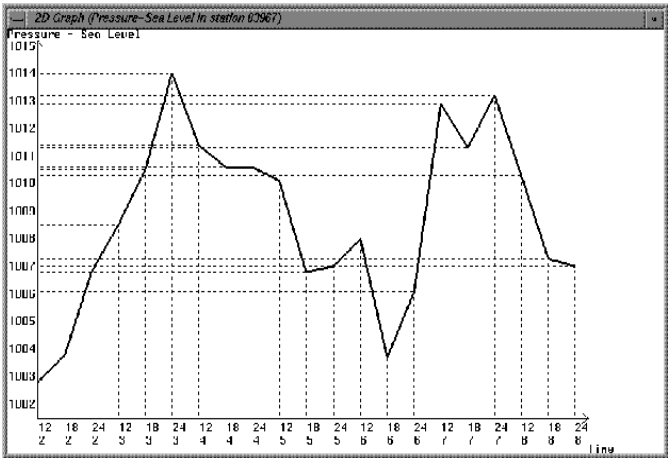

Figure 6: Line graph of pressure in one station for 21 date/time

In the surface graph, the X and Z axis correspond, respectively, to latitude and longitude, and the Y axis corresponds to the selected variable. The stations' positions are plotted and the values of the variable are interpolated to generate the surface. With this kind of graph it's easy to observe the behavior of a variable, for several stations, in a specific sample. The interpolation method is the multiquadric one, and, so, this graph is a 3D view of the regular grid obtained as the first step in the contour map tool.

6 Discussion and Related Work

The main goal that lead us to develop VisualMet was to exercise the tool-oriented approach (and related methodologies) [4,3] in a real environment. Visualization and exploration of meteorological data as a support for the daily work of meteorologists in the 8^{th} DISME came out as a challenging task due to the volume and nature of data processed by them without up-to-date computing resources.

Functional and interaction facilities have strong influence in the acceptance of any system. Different systems present different solutions to this problem. METVIEW [2] uses a distributed architecture based on the idea of service-oriented architectures, which allow for the combination of different services (such as data access, data manipulation and visualization) on a single environment. In VIS-5D [19], however, the user interacts with a fixed rendering pipeline, that supports atmosphere and ocean studies. To keep its interface simple, as stated in a recent work [20], VIS-5D denies many rendering choices to its users.

Interaction within VisualMet is done by a simple point-and-click technique in two collections of objects: the samples that the meteorologists already know, and the tools. Use of the tools are straightforward: the user is prompted for the input of any necessary information, immediately after the selection of the tool.

In a data exploration and visualization environment, and this is our case, the complexity and large volume of data impose the development of interaction facilities on top of visualization techniques. The query tool over a contour or icon map is a unique feature, that allows immediate access to original data. In spite of having different visual representations, meteorologists can query data in a homogeneous way, without changing the interaction style or switching to another tool.

As to the images that can be generated, literature is plenty of examples of what images can depict meteorological data. Bidimensional images consist, basically, of cartographic maps, where contour lines or false-colors represent scalar fields, and vector icons related to wind information can be depicted. Icons and glyphs can also be used to encode more than one variable.

Three-dimensional images enhance the representation of variables and phenomena with realism. A typical example often used nowadays is the stereo pair. Papathomas [16] implemented this method for clouds visualization. Alternative techniques used for the enhancing three-dimensional images of meteorological data and phenomena are transparency and textures.

The display of contour lines, obtained through scalar variable interpolation, can also be done in a three-dimensional domain, as illustrated by Papathomas [17], that presents an image where the contour lines are representing the dew point temperature, the pressure is represented by the shade of surfaces, and the colors depict the temperature.

A variety of three-dimensional graphs, that somehow influenced our 3D Graph tool, are presented by Hibbard and Santek [19, 20]. A surface can be defined and exhibited in a wire frame form or with texture and transparency to represent scalar data. Vector variables are exhibited as lines where orientation and thickness represent trajectory and magnitude, respectively.

The selection of visual representations for SYNOP data was made based on a methodology [3], as stated earlier, and also, on exchanging information with the meteorologists. The notion of a visualization technique as a mapping between data and visual representations is present in VisualMet. Also, this mapping is actually independent of the way data is stored. The **Interface** class, upon the activation of the mapping (by the user's selection of a tool) sets up the parameters and delivers the selected data to the mapping tool (i.e., the visualization technique). This is also a unique feature of our architecture. Well-known visualization systems as AVS [21], Khoros [22], and Iris Explorer [23] are based on the dataflow paradigm where visualization modules must know the structure of the data. This just does mean, however, that they implement the mapping concept in another way.

Some of the existent systems for meteorological data visualization work with data generated by simulations, that usually consist of three and four dimensional grids [10, 18, 19]. Another ones support a standard data format called GRIB data [2], or different sorts of data, such as satellite images, results of numerical models, and data gathered by remote-sensing instruments [18]. General visualization systems (Iris Explorer [23], for example) requires the conversion of user data to internal formats. Another ones (EnSight [24], for example) are tailored for engineering applications, despite being advertised as general purpose systems.

Data in our real environment (8^{th} DISME) comes in a variety of formats. Notwithstanding the current implementation of VisualMet deals only with SYNOP data, as it was designed using the tool-oriented approach and implemented following the object-oriented paradigm, it can be easily extended to cope all the formats of data received by the 8^{th} DISME.

7 Conclusions

General purpose visualization systems such as those based on the dataflow paradigm [21, 23] can not be used by meteorologists without programming skills because they often require the development of one or more modules, specially for data conversion. Moreover, we understand the process of scientific data analysis as the application of many tools, just ordered by the purposes of the scientist

in exploring his/her data. Our approach to this kind of application has been to develop systems tailored as sets of tools that can be applied on data entities. For sure we might have developed this approach on top of the dataflow paradigm, but it wouldn't be as efficient and elegant as programming it in a pure object oriented environment. This reflects also in the performance of the system. The most processor-demanding tool, the contour map, takes only 2-3 seconds to interpolate data and show the resulting image.

The participation of the meteorologists in the design and evaluation phases was essential and the initial set of tools that we provided them can now be easily extended.

The first extension is related to creating new entities based on the existing ones. For example, a sample has information about all the 32 stations and any tool that deals with samples, process all the 32 stations. A selection tool can be provided in order to allow the user to reduce the number of stations in the set he/she will work with. The new sample created by selecting a number of stations would appear in the Entities Base, and could be used as all the original ones.

Another tool that we are designing is an Animation tool that can be used to exhibit the dynamic change of contour maps over time. Thus, the meteorologist can see the changing behavior of temperature or atmospheric pressure, for example. Planned future work is also the porting of this implementation to a PC-platform, that is most commonly found in the 8^{th} DISME.

8 References

[1] A. E. Tarwater. "A Parameter Study of Hardy's Multiquadric Method for Scattered Data Interpolation". *Technical Report UCLR-53670*, Lawrence Livermore National Lab., Livermore, Calif. (1985).

[2] B. Raoult, B. Norris, J. Daabeck, R. Cartaxo, G. Câmara. "Distributed Architectures for Environmental Visualization Systems". In: *VIII Brazilian Symposium on Computer Graphics and Image Processing*, São Carlos, Brazil, October 1995. Proceedings, pp. 249-256.

[3] C.M.D.S. Freitas and F.R.Wagner. "A Methodology for Selecting Visual Representations in Scientific and Simulation Applications". In: *VI Brazilian Symposium on Computer Graphics and Image Processing*, Recife, Brazil, October 1993. Proceedings, pp. 89-98.

[4] C.M.D.S. Freitas and F.R.Wagner. "The Tool-Oriented Approach for Visual Exploratory Analysis". In: *VII Brazilian Symposium on Computer Graphics and Image Processing*, Curitiba, Brazil, October 1994. Proceedings, pp. 197-203.

[5] I. H. Manssour; C. M. D. S. Freitas and D. M. Claudio. Um Sistema para Visualização e Exploração de Dados Meteorológicos. In: *VIII Brazilian Symposium on Computer Graphics and Image Processing*, São Carlos, SP, October 1995. Proceedings, p. 321-322.

[6] I. H. Manssour; C. M. D. S. Freitas; D. M. Claudio and F. R. Wagner. Visualizing and Exploring Meteorological Data using a Tool-Oriented Approach. In: INTERNATIONAL CONFERENCE ON VISUALIZATION AND MODELLING, December 1995, Leeds, UK. **Proceedings...**.

[7] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen. "Object-Oriented Modeling and Design". *Prentice-Hall, Inc.*, 1991.

[8] M. O. Dayhoff. "A Contour-Map Program for X-Ray Crystallography". *Communications of the ACM*, Vol. 6, No. 10, 1963, pp. 620-622.

[9] M. H. Overmars. "Forms Library - A Graphical User Interface Toolkit for Silicon Graphics Workstations". Utrecht University, 1992.

[10] N. Max, R. Crawfis and D. Williams. "Visualization for Climate Modeling". *IEEE Computer Graphics & Applications*, Vol. 13, No. 4, July 1993, pp. 34-40.

[11] R. E. Carlson. "Multi-Stage Algorithms for Solving Scattered Data Interpolation Problems". *Unpublished presentation at $4th$ Texas Symp. on Approximation Theory*, Texas A&M Univ., College Station, Tex. (1983).

[12] Silicon Graphics. "Graphics Library Programming Guide". Mountain View, CA: Silicon Graphics, 1991.

[13] S. Wolfram. *Mathematica - A System for Doing Mathematics by Computer*. 2nd ed. Addison Wesley, 1993, 961 p.

[14] T. A. Foley. "Interpolation and Approximation of 3-D and 4-D Scattered data". *Comput. Math. Applic.*, Vol. 13, No. 8, 1987, pp. 711-740. [TA 87]

[15] T. A. Foley and D. A. Lane. "Visualization of Irregular Multivariate Data". *Visualization'90*, San-Francisco, October 1990. Proceedings, pp. 247-254.

[16] T. V. Papathomas, J. A. Schiavone and B. Julesz. "Stereo Animation for Very Large Data Bases: Case Study - Meteorology". *IEEE Computer Graphics & Applications*, Vol. 7, No. 9, Sep. 1987, pp. 18-27.

[17] T. V. Papathomas, J. A. Schiavone and B. Julesz. "Applications of Computer Graphics to the Visualization of Meteorological Data". *Computer Graphics*, Vol. 22, No. 4, Aug. 1988, pp. 327-334.

[18] W. L. Hibbard and D. Santek. "Visualizing Large Data Sets in the Earth Sciences". *IEEE Computer*, Vol. 22, No. 8, Aug. 1990, pp. 53-57.

[19] W. L. Hibbard and D. Santek. "The VIS-5D System for Easy Interactive Visualization". *Visualization'90*, San-Francisco, October 1990. Proceedings, pp. 28-35.

[20] W. L. Hibbard, B. E. Paul, D. E. Santek, C. R. Dyer, A. L. Battaiola and M. Voidrot-Martinez. "Interactive Visualization of Earth and Space Science Computations". *IEEE Computer*, Vol. 27, No. 7, July 1994, pp. 65-72.

[21] C. Upson, T. Faulhaber, D. Kammins, D. Laidlaw, D. Schlegert, J. Vroom, R. Gurwitz and A. Van Dam. "The Application Visualization System: a computational environment for scientific visualization". *IEEE Computer Graphics and Applictaions*, Vol. 9, No. 7, July. 1989, pp. 30-42.

[22] J. Rasure and C. Williams. "An integrated data flow visual language and software development environment". *Journal of Visual Languages and Computing*, Vol. 2, No. 3, September 1991, pp. 217-246

[23] Silicon Graphics. *Getting Started with IRIS Explorer*^{TM}. Mountain View, CA: Silicon Graphics, 1991.

[24] CEI - Computational Engineering International, Inc. *EnSight User Manual for Version 5.5*. North Carolina, USA: CEI, 1995.

Acknowledgments

Special thanks are due to the 8^{th} Meteorological District, which has provided all the necessary information for the design and implementation of VisualMet. We also gratefully acknowledge Marcelo Gomes de Oliveira, for the programming of the Graph tools. This work has been partially supported by CNPq (Conselho Nacional de Desenvolvimento Cientifico e Tecnologico) and CESUP-UFRGS (Centro Nacional de Supercomputacao, Universidade Federal do Rio Grande do Sul).