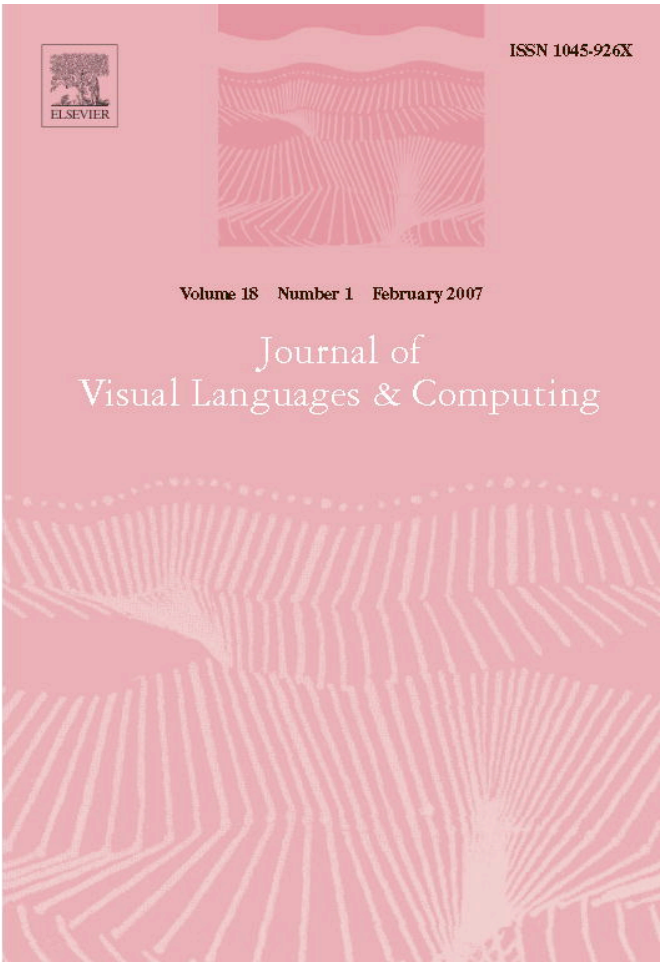

Available online at www.sciencedirect.com
ScienceDirect
Journal of Visual Languages and Computing
18 (2007) 1–21

Journal of
Visual Languages
& Computing

www.elsevier.com/locate/jvlc

TimeLine and visualization of multiple-data sets and the visualization querying challenge

David A. Aoyama, Jen-Ting T. Hsiao, Alfonso F. Cárdenas*, Raymond K. Pon

Computer Science Department, University of California, Los Angeles, USA

Received 11 February 2004; received in revised form 11 November 2005; accepted 11 November 2005

Abstract

Data in its raw form can potentially contain valuable information, but much of that value is lost if it cannot be presented to a user in a way that is useful and meaningful. Data visualization techniques offer a solution to this issue. Such methods are especially useful in spatial data domains such as medical scan data and geophysical data. However, to properly see trends in data or to relate data from multiple sources, multiple-data set visualization techniques must be used. In research with the time-line paradigm, we have integrated multiple streaming data sources into a single visual interface. Data visualization takes place on several levels, from the visualization of query results in a time-line fashion to using multiple visualization techniques to view, analyze, and compare the data from the results. A significant contribution of this research effort is the extension and combination of existing research efforts into the visualization of multiple-data sets to create new and more flexible techniques. We specifically address visualization issues regarding clarity, speed, and interactivity. The developed visualization tools have also led recently to the visualization querying paradigm and challenge highlighted herein.

Keywords: Data visualization; Visualization architecture; Visual querying; Multi-platform visualization; Remote processing

*Corresponding author.
E-mail address: cardenas@cs.ucla.edu (A.F. Cárdenas).

1045-926X/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2005.11.002

1. Introduction

There has been a great increase in the amount of data that is being collected and stored in computers. The increase in information comes from a great number of sources, including digitizing data stores that were previously only available in physical format, data that is captured through electronic tools, such as in radiological scans and satellite imagery, and data gathered by the proliferation of sensor networks. While these massive amounts of data have great potential in terms of analysis, there is a difference between raw data and the knowledge that can be extrapolated from them. Raw alphanumeric viewing of data often does not effectively convey the useful knowledge that the data contains. This is where data visualization comes into play.

There are several goals of data visualization. One is to create an environment that is representative of the data set in question. This gives context to raw representations of data, giving the user a more complete understanding of the data set's significance. This is ideal for spatial data in the medical and geophysical domains. To be analyzed properly, data collections in these fields must be visualized. Another goal of visualization is to aid the user in discovering trends in data by creating graphical depictions of data values. An example of this would be simple line graphs where value changes over time can more easily be seen. Data visualization can also make data analysis faster by focusing the user's attention to regions of interest automatically, possibly revealing properties of a data set not obvious from initial observation.

The field of data visualization as a research topic grew exponentially in the 1980s with the advent of powerful personal computers grew in popularity [1]. The increase in local processing power made it feasible for graphics representations of data to be generated in real-time at the user's desktop. This was the real breakthrough that made visualization techniques so powerful. Users could use these visualization tools to explore data and get customized views of interesting area on demand. With interactivity comes flexibility that makes the visualizations more meaningful.

As computing power has increased, visualization techniques have grown in sophistication. Computers have become capable of advanced real-time three-dimensional (3D) graphics, revolutionizing the types of visualizations possible. These increased capabilities have made it possible to explore the visualization of multiple-datasets simultaneously. This can be realized in either tracking changes in a single data set through time, thus visualizing the evolution of data, or through visualizing two or more datasets together; thus, allowing the detection of relationships between certain conditions.

While the visualization of a single data set can be fairly straightforward, considering multiple-data sets at once raises several issues. The challenge becomes creating visualizations that have clarity and understandability so that users can easily see what they need to see. Addressing these challenges, we have integrated various visualization techniques into the TimeLine project [2]. The focus of the paper is on bringing multiple data sources and streams together into a single visualization interface. Since the data types for the applications of interest to us currently encompass medical and geophysical data, the results of data queries are displayed to the user as graphical visualizations. We have made strides in incorporating multiple query results together to highlight changes in data as well as overlaying multiple results together for analysis.

In Section 2, we will first examine current visualization techniques, focusing on 3D techniques that are currently the most sophisticated available and have the ability to

convey the most information. Multiple-data set visualization applications and issues will be addressed. Current research efforts are discussed and related to our efforts, which are presented in Section 3, including the approaches used, the tools and technologies that are incorporated, and the integration of the visualizations into the TimeLine application. Implementation is discussed in Section 4. Finally, the challenge of the future visualization-querying paradigm will be considered in Section 5.

2. Visualization techniques/tools and multiple data set visualization

An understanding of basic data visualization techniques is necessary to fully appreciate many of the advances made in the visualization of multiple-data sets. In this section, current applications of multiple-data set visualization, issues, and descriptions of current 3D visualization techniques, including their relationship with our research into the visualization of multiple-data sets are discussed.

2.1. Applications

Two major domains for data visualization are the medical and geophysical data domains. They both lend themselves to visual representation because of their spatial nature. In addition, both domains often deal with temporal factors that also make them prime candidates for multiple-data set visualization for analysis of data evolution.

2.1.1. Medical

Major sources for medical imaging visualizations come from radiological scans. Major scans are CT and MRI scans that use techniques to non-invasively detect tissue type at a given 3D point within a patient's scanned area. In both scan types, the primary application of visualizations is to take the volumetric data sets obtained from a scan and reconstruct it for rendering. This gives physicians a tangible view of the data instead of having to view single 2D slices, which is more cumbersome and less intuitive. These 3D views allow for analysis of a patient's condition as if the patient were physically being examined.

However, both CT and MRI scans detect multiple types of body tissue. Examples of these types of tissue include bone structure, skin, vein and artery tissue, and soft tissue such as brain matter. It is most useful to users to be able to differentiate these different tissue types so that specific areas can be targeted and analyzed. Each type of tissue or identifiable scanned object can be considered a different data set. For example, this allows bones to be colored white while veins are colored red and arteries are colored blue [3,4]. This increases clarity, understandability, and usefulness of such visualizations.

In the medical domain, the evolution of patient condition is also essential to making diagnoses concerning a variety of conditions. These include bone fractures and cancer detection and progression [5]. By viewing the state of a condition in CT or MRI scans through time, physicians can better analyze the patient's current situation than if just the current scans is considered.

2.1.2. Geophysical

In an analysis of applications of the visualization of single geophysical datasets, we noted that the main sources of such data primarily comes from land-based sensors and satellite imagery. Land surface sensors, such as the CapeScience accessible airport weather

sensors [6], often capture a variety of different values. These include surface temperature, pressure, humidity, visibility, and wind speed and direction. In the case of satellite data, multiple bands of radiation are typically captured, each of which are useful for different applications. For example, with the MODIS satellite image data, 36 different bands of radiation are captured, grouped into 11 different categories: Land/Cloud/Aerosols Boundaries, Land/Cloud/Aerosols Properties, Ocean Color/Phytoplankton/Biogeochemistry, Atmospheric Water Vapor, Surface/Cloud Temperature, Atmospheric Temperature, Cirrus Clouds Water Vapor, Cloud Properties, Ozone, Surface/Cloud Temperature, and Cloud Top Altitude [7]. As is seen from the variety of different types of data that can be gathered from just a single satellite system, it is obvious that it can be useful to overlay several of these bands, so that the relationship of different properties can be seen.

In addition, geophysical sensors are often capturing data continuously, and satellites typically pass over a given region at least several times a week [7]. This gives the opportunity to examine the evolution of weather patterns or other physical phenomena through time. We have introduced the two-dimensional (2D) Image Stack architecture [8] that is ideal for the visualization of geophysical data in two dimensions. This project takes data from co-registered variables or datasets and provides a framework for representing and querying across these multiple variables.

2.1.3. Other domain applications

While data with spatial meaning, as is the case with medical and geophysical data, is ideal for visualization, analysis processes can also be applied to other sources of data. There has been research ongoing into the visualization of data mining results for online analytical processing (OLAP) operations in data warehousing systems. These systems take the conventional data cube view of data, which summarizes multiple dimensions of data, and makes them into visual, virtual entities. Systems of these types include the DIVE-ON system developed at the University of Alberta [9] and the Map Cube project from the University of Minnesota [10]. Such systems make it possible to use visualization to enable quicker analysis of data than would be possible through simple alphanumeric representation of OLAP results as is done with database management systems and specialized OLAP software. There are also efforts to represent astrophysical simulation data and search result data that frequently deal with multiple dimensions [11]. These systems benefit from multiple-data set visualization because aspects beyond three dimensions can be represented within single visualizations.

2.2. Challenges

There are many issues that must be addressed when dealing with the visualization of data. These are obstacles that must be overcome for visualizations to be effective. Issues range from logistical, back-end type problems that include speed and data set preparation to fundamental problems, such as visualization clarity.

Clarity is a major issue with any type of graphical presentations. Effort must be made to create easy to understand visuals that effectively convey the proper meaning to the user. Clarity can be achieved in the field of data visualization by using appropriate rendering methods, creating intuitive viewing environments (including layout and observer viewpoint), and highlighting important sections of visualizations. Appropriate rendering methods can depend on the type of data set being viewed. Highlighting important sections

in visuals is also helpful in improving clarity. This focuses the user's attention to potentially interesting sections of a data set. However, extrapolations of exactly what is considered interesting varies widely from domain to domain, and often cannot even be determined without the use of a domain expert. We have made strides toward creating clearer, more understandable visualizations through innovative combinations of data set manipulation and rendering techniques.

Speed and interactivity are also important issues that must be considered when dealing with visualizations as the rendering speed of an image directly affects the responsiveness of a visualization. There are two main factors that affect the speed of a rendering: processing time and network latency (if image processing is not done on the local machine). A major requirement to make computer-based data visualizations most useful is dynamicity of such renderings. This includes being able to change viewing parameters such as color, zoom, and rotation in real time. This customizability allows the user to use visualizations to explore data rather than just use them as static representations or summaries of data. We have created an environment in which the visualizations can be manipulated in 3D space in near real-time. We present a networked data acquisition and rendering architecture allows for a centralized rendering solution, which reduces client computer taxation for stable response regardless of client computer speed in Section 4.

Data set preparation is an important issue when choosing data sets for use in multiple visualization techniques. To make many of the data sets ready for visualization, a significant amount of pre-processing must take place. As mentioned before in the consideration of medical applications, it is often useful to separate or identify different tissue types or organs from each other before visualization. This brings the issue of image segmentation into the forefront. There is much research into developing image segmentation techniques that can be extended to 3D type medical scans. Popular segmentation algorithms include that of edge based and watershed methods [12] or region growing techniques [13]. While there are several viable segmentation algorithms, none of them are guaranteed to completely segment an image or a volume correctly. Almost all methods are time consuming and cumbersome to setup. In addition, to analyze if regions have been properly segmented, domain experts must be consulted. This is a big concern and issues involving segmentation must be solved before visualization can be effectively scaled. Co-registration is also a challenge in data set preparation as different sources of data are often not calibrated to one another and processing must be done to ensure that corresponding data sets are properly aligned [14,15]. While these types of operations can be done manually by a domain expert, co-registration techniques are most useful when they can be automatically done. In Section 4, we highlight how these issues are addressed.

2.3. Related works

In this section, we discuss existing visualization techniques, many of which we incorporate into a new visualization system, the TimeLine. We classify the following visualization techniques into two categories: multiple-data set visualization and 3D volumetric visualization.

2.3.1. Multiple-data set visualization

While visualization of individual datasets is useful in itself, such tools become much more powerful when multiple-datasets are involved. These techniques create new

opportunities for data analysis and interpretation. A subset of visualizing multiple-data sets is the area of multidimensional or multivariate data visualization. In these cases, a single data set is still considered, but it involves multiple dimensions for each data value. An example of this would be data cube-type operations [10] in which a single data value can vary with parameters such as temperature, humidity, and pressure.

To analyze relationships between multiple-data sets or to explore how data changes over time, multiple-data set visualization must be used. Multiple-data set visualization is useful for comparing or relating datasets to one another. This can be used for a single data type, as is the case for multidimensional data sets, or for multiple data types. This creates two categories of comparison visualizations. The first type is vertical in nature in that it involves comparing multiple modalities of data sets to each other. In these types of visualizations, the datasets are often obtained during the same temporal time frame. These are useful for finding trends and relationships that exist between different datasets. For example, in the geophysical domain, Los Angeles smog levels may have an effect on the land surface temperature of certain areas. In addition, the smog levels are no doubt related to the positioning of major freeways and the topology of the region. With vertical comparisons between these datasets, users can get a more complete picture of the current data set values. The other category of comparisons is horizontal in nature and involves examining the change and evolution of a single data set type over time. Within the visualization itself, there are multiple visible sets of data, but the source of the data is a single actual data set. An example of this type of situation would be the movement of precipitation over a given region over time. Horizontal natured comparisons would be able to follow and make observations about the evolution of a single data set.

2.3.1.1. Overlaying visualizations in single views. In single data set visualization, just a single entity is viewed and rendered. An extension to this system is that of overlaying multiple visualizations within a single view. This overlaying of multiple data is fairly trivial in the case of one and 2D representations. An easy example of this can be seen in one-dimensional x, y plots. Even when extended to two dimensions, areas can be represented through techniques such as semi-transparent filled areas or through the use of contours or other outlining techniques.

However, when moving into the third dimension, the difficulty in creating clear and useful visualizations becomes more of a challenge. This is partly because of the limitations of computer display systems as well as the way humans perceive vision. When creating 3D visualizations, the rendering system is trying to recreate objects in 3D space in a 2D canvas, the computer display. While there are research efforts into virtual reality-type systems, which can give depth to displays [9,10], this still does not solve all the problems dealing with the creation of functional representations of the datasets. The main problem of creating such visualizations is that virtual reality-type systems do not solve fully involves the creation of views that hide potentially important information from the user. This involves using various rendering techniques outlined above as well as using other novel methods for making visualizations clearer and more useful. There are efforts to make combined views of multiple regions in medical scan data [3,4] using multiple types of 3D volume rendering techniques as seen in Figs. 1 and 2.

Another data set visualization technique is multidimensional scaling (MDS), which uses similarity (or dissimilarity) matrices to visualize the relationships between datasets by plotting points whose Euclidean distances between each other correlate to the similarity

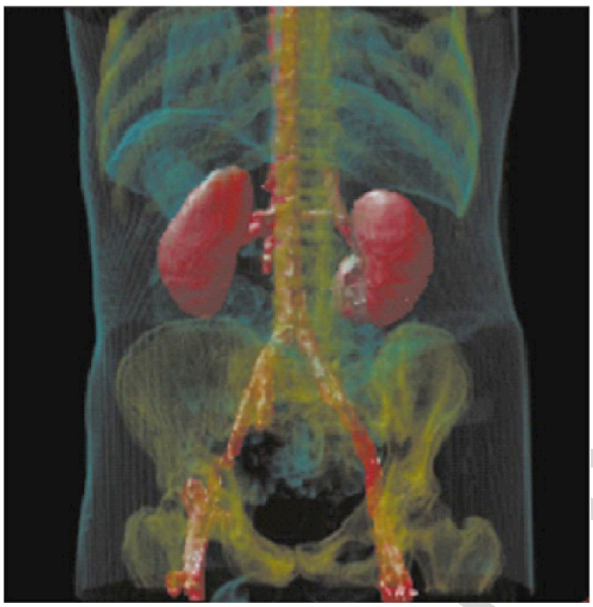

Fig. 1. Two-level torso rendering example [3].

Fig. 2. Two-level hand rendering example [3].

matrices [16,17]. This technique can be used for data clustering and dimensionality discovery and has been applied to many domains, including document classification [17,18] and bioinformatics [19]. Our work in comparing a data set at multiple points in time could be alternatively represented as an MDS plot to view the overall similarity of these multiple sets. However, this representation would most likely be more abstract and lose the correlation of geographic position of the data represented.

Other research uses projections of data set data onto 2D planes [20] and combines data set evolution techniques [15] with 3D visualization in the medical domain. In the TimeLine system, we combine these different techniques to create more intuitive visualizations as well as extend the techniques into the geophysical domain.

2.3.1.2. Multiple views. Another major approach to viewing multiple-data sets is that of multiple views. In this technique, several viewing windows are utilized so each window can possibly represent a different aspect or dimension of the data. Another basic use of such a system is to show different data sets in each of the views in the same orientation so that users can easily make comparisons between the datasets. This can be advantageous over using a single visualization view, since such comparisons can be difficult if the user must switch between views to fully examine two or more datasets. This is especially true in cases in which overlaying multiple-data sets in a single view can become too cluttered.

There are groups actively encouraging the use of multiple views [21] when appropriate. It is noted that, as in all cases with data visualization, the key issues to deal with are the utility that can be gained from using multiple views. All systems should add to the experience of the user and focus their attention rather than distract and clutter the workspace. The systems presented in this section strive toward this goal and through user testing [21–25], have shown that using multiple views can be beneficial. There has been research into creating easy environments in which multiple views for multiple-data set/dimensional visualization in addition to systems that automatically create multiple views to facilitate the accessibility of a natively complex visualization environment [22,25]. Additionally, research has also been made into user-friendly environments, which can be used to easily link multiple view windows together [24] to generate different views of highly dimensional data.

In a subsequent section we introduce our advances that go beyond the current state of the art highlighted above.

2.3.1.3. Glyph-based rendering. Another useful method for multidimensional data in particular is that of glyph-based rendering. Multidimensional data can be defined as data in which single points of data have multiple attributes or parameters that can be used to describe the point. For each of the rows that exist within a data set, there are a variety of attributes. Having so many dimensions causes difficulty in properly representing the data in visualization presentations because of the limits of 3D plots. Three dimensions can be assigned to traditional data points, but beyond purely positional variance, standard plots are unable to handle data of four dimensions or higher.

Glyphs are generally defined as icons in 2D or 3D space that either represent data points or are used to represent some context parameters for a given view. They address the representation of four-dimensional (4D)+ data sets by introducing other visual parameters to vary. Types of variation can be categorized into different types such as shape, size, orientation, position, color, texture, and transparency. These variations have been used to add higher dimensions to 3D glyphs to create visualizations capable of representing up to 9 dimensions [11,26,27]. An example of glyph-based visualization can be seen in Fig. 3. There is also research in combining glyphs with geophysical visualization within a single view [28].

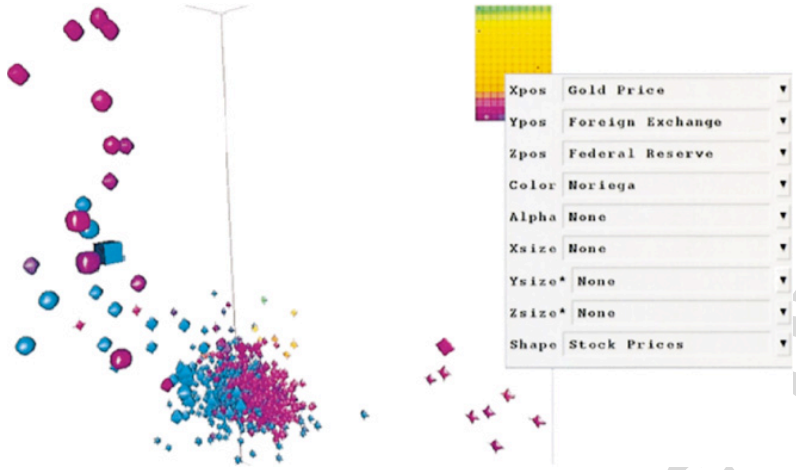

Fig. 3. Superquadric shape generation example [26].

2.3.2. 3D volumetric rendering and visualization techniques

3D rendering techniques have become an essential to the visualization of multiple-data sets together. Several methods of 3D data set rendering are used heavily in information visualization. It is important to be familiar with basic rendering techniques to understand the nature of the visualization techniques proposed. Some of the more popular techniques include: direct volume rendering (DVR), maximum intensity projection (MIP), isosurface (IS) rendering, and non-photo realistic rendering (NPR). We will highlight briefly the first three of these methods.

All three of these types of visualization techniques take as input a 3D array representing the volume to be rendered. DVR or voxel rendering represents each point in the input array as a rendered point in 3D space [3]. This method is useful in representing volumes that do not have definite predefined boundaries, such as clouds. MIP rendering is similar to DVR except that for each point in the rendered image, only the highest value in the input array is displayed [3]. This creates renderings that are often clearer and sharper than DVR. IS rendering extrapolates a defined border surface for the volume [3,15,20]. This rendering technique is the most common choice for rendering volumes that represent tangible objects with clearly defined borders such as bones in a medical scan.

3. TimeLine and multiple data stream visualization

Significant efforts have been into the development of the Multimedia Stream System TimeLine [2] to provide access to multiple temporal heterogeneous data sources, some streaming and some static. The method in which this is done is through a three level hierarchical interface to the data sources. This can be seen in the screenshot in Fig. 4 [2]. The left-hand side of the interface displays the data sources available to execute temporal queries on. Data sources are then selected and a query is run on those data sources. The result of this initial query is a visualization of the results in a time-line format in the bottom pane. Each of the data sources is represented by a single line in the time-line, with icons representing results from the query. The time-line can be zoomed in or out to get a more general or more detailed view of the results. The final level of querying takes place when the user clicks on a given query result and the actual contents of the query result are

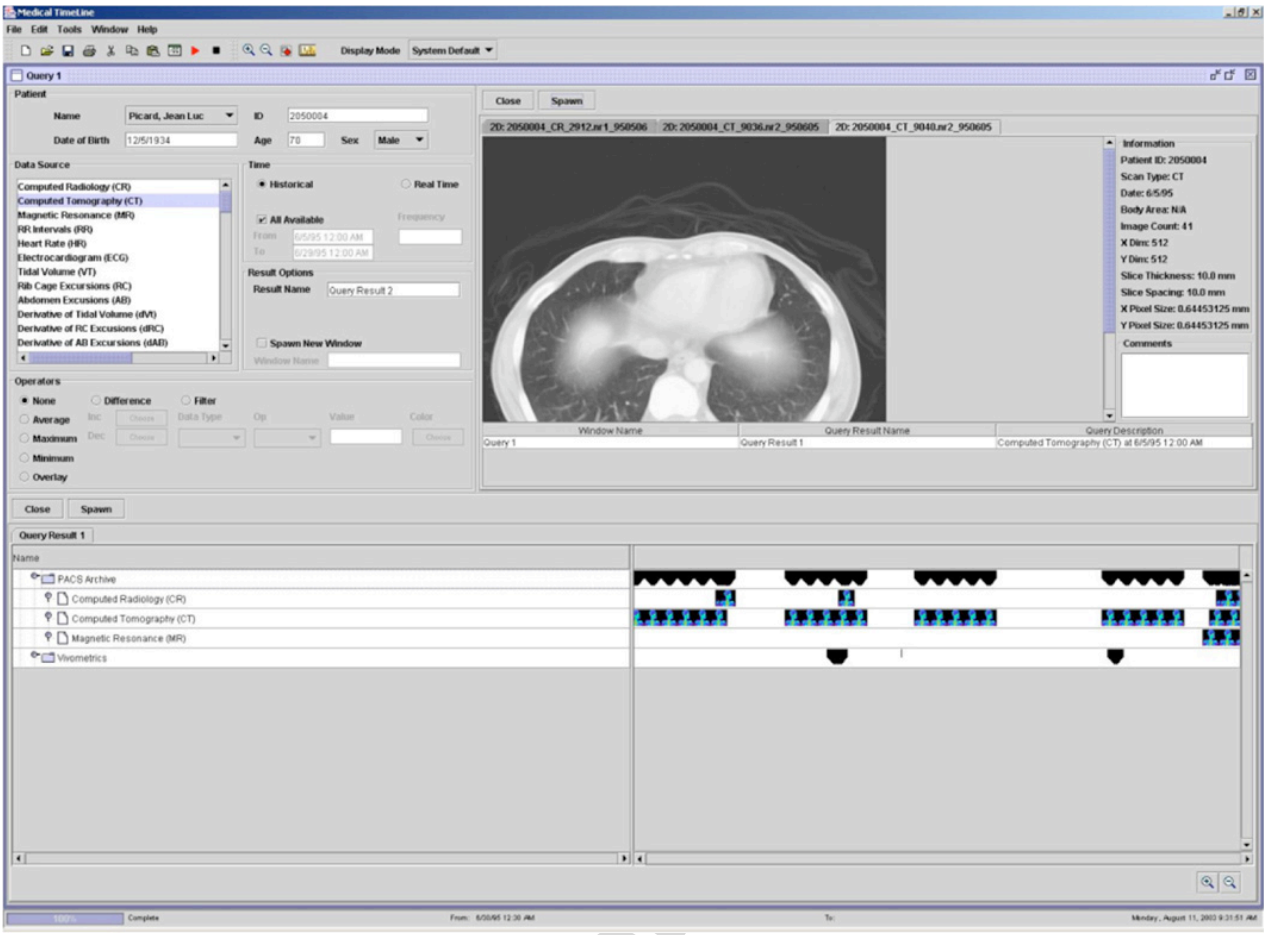

Fig. 4. TimeLine screenshot [2].

displayed in the results pane in the upper right corner. In the example image in Fig. 4, a CT scan is being viewed in 2D form.

The TimeLine system deals with the two major application domains of medical and geophysical data. Fig. 4 represents the medical TimeLine interface. The geophysical interface is similar, but deals with different data sources and features different data visualizations.

In this section, we will cover the major innovations that the TimeLine project represents in terms of multiple-data set visualization and information visualization in general. Implementation details will then be given including an examination of the tools used as well as the integration of visualization engines into the TimeLine application. Issues are then discussed along with future directions for development.

3.1. TimeLine user interaction and paradigm

The TimeLine user interface contains three major parts: the query module, the time-line module, and the visualization module. The query module allows the user to specify relevant streams regarding the subject and the type of the information that is desired. The time-line module gives the user an initial view of the available streams of data. The visualization module provides the user a medium in which he or she can build complex views of the desired data.

The query module consists of the location, data source, time, result options, and operator's section. Within the location section, the user can select from a set of predefined general area such as "Southern California," using a pull-down tab or manually define an area to explore by entering the longitude and the latitude of the bounding rectangle to the area. Although the following discussion of TimeLine is in the context of geographical applications, TimeLine applies to other domains as well such as medical domains in which we are active. Within the data source section, a menu is displayed where the user can select from a set of registered data sources to populate the time-line and view historical or real-time streaming information. A single data source can be selected using a single mouse click and multiple data sources can be selected easily. Within the time section, the user can specify the time property of the data. Historical data and real-time streaming data consist of the two classes available in the system. The start and end date and time can be specified for historical data and the retrieval frequency can be specified for real-time streaming data. Within the result options section, the user can assign a name to each of the queries and specify whether this query is to be spawned in a new window. This gives the user the ability to build new queries upon previously formed queries and some user interface management options. Within the operator's section, the user can apply operators to queries such as average, maximum, minimum, difference, and color filters. The user can also specify the retrieved slices of data to be overlaid on top of one another to build 3D visualizations.

The time-line module visualizes instances of data that are returned from the query specified by the user. When a query is executed by pressing the red triangle button at the top of the user interface, a new tab is created in the time-line module, displaying the retrieved data in a time-line format. Each selected data source is organized as leaf nodes using a tree structure, according to the type of data returned from each data source. The parent nodes in the tree group similar types of data sources together, and display the union of its children's data instances as its set of data instances. Thus, if a level of a tree is collapsed, the parent level, which is still shown, would give the user an idea of the amount of data its children hold. The time-line module also allows for zooming in and out to give the user a better view of the available data.

The visualization module displays 2D and 3D visualization for the selected instances of data. Data is selected using the query and time-line modules. After selecting the desired data, records are retrieved from the database or from a real-time streaming source, processed by the visualization module, and shown to the user utilizing various software packages. The user can rotate the 3D visualization by specifying a new angle with which to view from. To have a better view of the visualization, the user can spawn the visualization module into its new window, where additional zooming in or out can be called upon by clicking on the zoom tab at the top of the window. Information, such as window name, query result name, and query information for each slice of data that compose the visualization, is listed and shown. By doing so, the user can see a visualization of the data and the text description of the data in the same visualization module.

3.2. Innovations

There are two major innovations in the TimeLine project in terms of multiple-data set visualization. The first has to do with the nature of the entire system. Using ideas similar to the Snap-Together visualization project [24], which is subsequent to our work, the TimeLine interface has a three-level hierarchical interface for viewing data sources, viewing

query results, and finally viewing the contents of those results. Another subsequent project is the COPLINK project [29] that uses modular windows to create an interface to a police information collaboration system. Each panel in the application is linked to the others, and user activity progresses naturally from the higher-level data source panel to the lower level visualization panel. The transparent and intuitive visualization of data from these heterogeneous data sources is beyond many of the research efforts examined, which typically just use a single data source. In addition, the ability to include real-time streams further extends the capabilities of the TimeLine architecture. Although we have not fully implemented yet this real-time capability, near-real-time access has been implemented.

The second major innovation comes in the lowest level of data visualization in the application. The TimeLine provides a common platform through which a variety of different data visualization methods converge. The challenges described in Section 2.2, including clarity, speed, and interactivity, are addressed by the TimeLine architecture through both design and implementation. The specifics of how each of these issues are addressed are detailed in Section 4.6. In the main results rendering window, we use combinations of different techniques that have been previously researched. We combine ideas from the GRUVI project [15], which uses techniques to visualize the evolution of MRI data with those of the Vested Magic Mirrors project [20], which uses 2D projections to add clarity to visualizations in our geophysical representations of data. An example of this type of visualization generated by the system can be seen in Fig. 5.

In this Fig. 5 example, two sets of land surface temperature are plotted in 3D over a map of the region. On the bottom of the visualization, there is a 2D map of the United States indicating the geographic limits of the visualization. The x, y dimensions represent longitude and latitude, respectively. The z-axis represents the temperature in Kelvin degrees. Above the 2D map, lies a 3D plot representing two temperature readings from two different data sets (shown as white grid meshes surrounding the red and blue volumes in

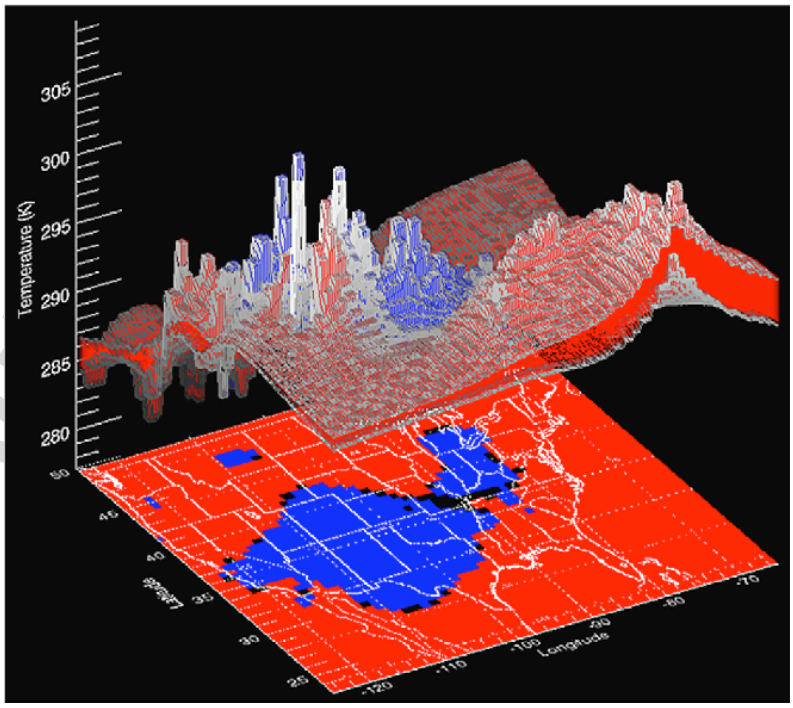

Fig. 5. Two land surface temperature plots (red = increase in temperature, blue = decrease in temperature).

3D space). The difference (positive change or negative change) between the two plots is filled in indicating the change between the two datasets. Regions in which temperature increases between data sets taken at different times appear as red DVR regions and decreases in temperature appear as blue regions. Furthermore, these regions are projected onto the map, so a clear indication is shown of exactly where this phenomenon occurs.

In addition, we have created unique visualization methods for temporal geophysical data that characterizes the evolution of data through use of value thresholding and data aggregation. This can be seen in Fig. 6.

In this example, land surface temperature (in blue) and humidity (in red) of a region are tracked over ten time steps and their evolution is plotted in three dimensions. The data sets are interpolated fields obtained from CapeScience [6] airport data from November 14, 2003 to November 21, 2003. The z-axis in this visualization represents different time slices. Each level of time indicates a different temporal data set. A threshold level is used to determine if a given point is filled in the volume. For temperature, only values above 298 K are displayed and humidity values above 80% are displayed on each level. This type of visualization is effective in detecting evolutionary changes in a single modality as well as comparing the behavior of data from two modalities. This is powerful because it provides for analysis of complex situations over both time and data type within a single view.

A final case shows an experiment in using multiple z axes to visualize multiple modalities of data. An example of this is seen in Fig. 7. This visualization is similar to that in Fig. 5, except in this case four datasets are represented in two different data types. Temperature is shown on the lower plot, similar to the example in Fig. 5. Pressure is shown on the upper plot. In both plots, the differences in the two data sets plotted are filled using DVR techniques. Red represents positive change and blue represents negative change.

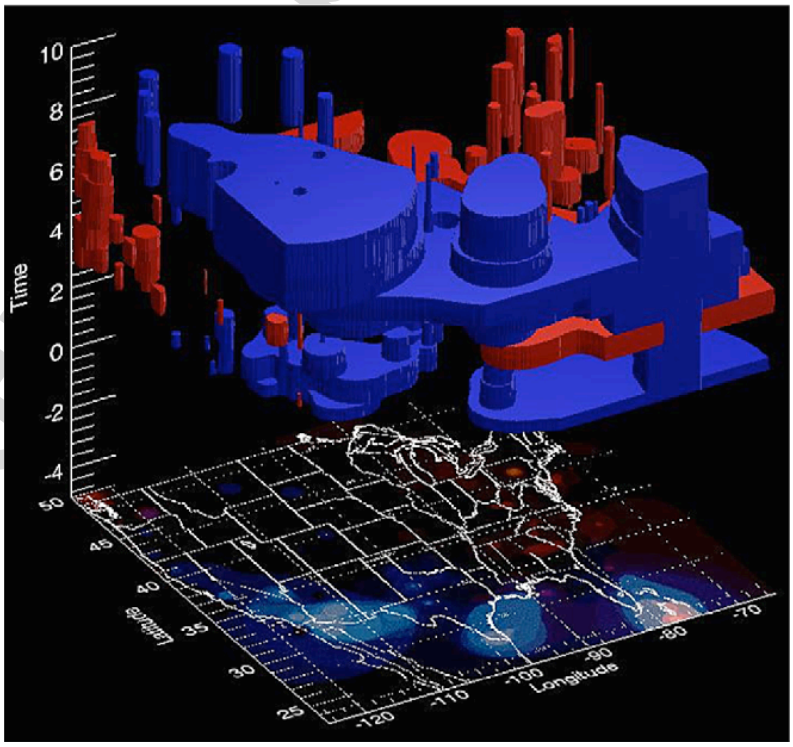

Fig. 6. Temperature/humidity evolution.

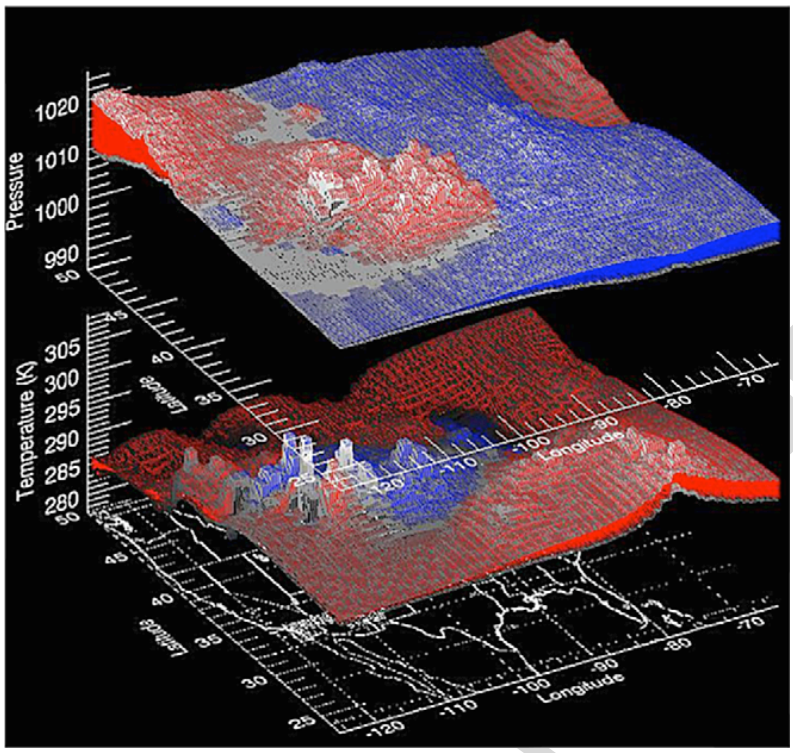

Fig. 7. Multi-modality plot example.

4. Performance and implementation challenges

We have used many different tools when building the lower level visualization engine for the TimeLine project. The main rendering tool for both 2D and 3D type plots is the interactive data language (IDL) package from Research Systems Inc. (RSI). Since the TimeLine application is written as a Java application/applet, to access the IDL environment, another RSI product is used to provide a Java interface to the system. This system is called IDL on the Net (ION). We make use of IDL's Java Bridge architecture to allow better integration of the IDL runtime environment and Java. The time-line module is implemented on top of the E-Gantt [30] open-source software package, which has many built-in chart-chart functions.

4.1. IDL

While the time-line presentation of query results and some of the simpler types of visualizations in the TimeLine application are done natively in Java, IDL is the heart of the advanced visualization engine which creates all of the 2D and 3D type plots of data. IDL is useful as a customized environment that is tailored-made for data set manipulation and processing. Many high-speed built-in functions exist for preprocessing datasets to make them suitable for visualization.

In addition, powerful options exist for the creation of a wide variety of visualizations in both the medical and geophysical domain. Included are the volume rendering techniques: IS, DVR, and MIP. There exists some functionality to support some basic image and volume segmentation, but these functions have yet to be explored. In geophysical

rendering, the system has functionality to natively handle the longitude/latitude coordinate system and provides functions to help with the interpolation of sparse data sets.

4.2. ION

IDL natively operates in its own operating environment, much like other analytical languages, such as MATLAB [31]. To integrate IDL into the Java-based TimeLine application, the ION tool is used to provide an interface to an ION server. The Java client opens an ION connection to the ION server, which creates a separate IDL runtime environment for each connection. Once connected, primitive variables can be transferred between the Java client and the ION session, and IDL commands can be executed. The results are sent back to the Java client in the form of a panel containing an image of the results of the commands given to IDL. This means that the power of the rendering system can be accessed and viewed through the Java TimeLine interface.

4.3. Java integration

IDL includes the Java Bridge module that allows IDL to create and manipulate Java objects within the IDL runtime environment. While native communication between Java applets must still go through the ION system, the Java Bridge allows for IDL to directly contact data sources that are encapsulated within Java objects. This creates a much more flexible environment in terms of network communication and data access. Because the datasets can be retrieved and processed completely without sending them to the user machine, reduction in network latency is great. This also increases data security for sensitive data sources, since the data never leaves the controlled network environment between the data source servers and the IDL/ION server.

4.4. Multiple data set/stream visualization integration into TimeLine

The overall infrastructure architecture for the TimeLine application is outlined in Fig. 8. Both the TimeLine application and the ION/IDL server have access to the data sources. However, since the IDL/ION server uses the Java Bridge to access the sources, the amount of data that is sent to the actual client machine is limited. This is the key in reducing the amount of network traffic that travels to the client, which is important because often the connection to the client can be slow or unreliable since it is operating over the Internet. The previously mentioned issues along with data security are also alleviated with the integration of the current system.

4.5. Implementation

The visualization components implemented in the TimeLine project have used the most current versions of all the software tools mentioned above. IDL and ION has been used as the backend-rendering engine that all the visualizations in the example figures were created in IDL and ION. The Java Bridge architecture is integrated into the infrastructure to increase efficiency and performance. Java classes encapsulating each of the visualization types have been created for integration into the TimeLine application. Since the entire

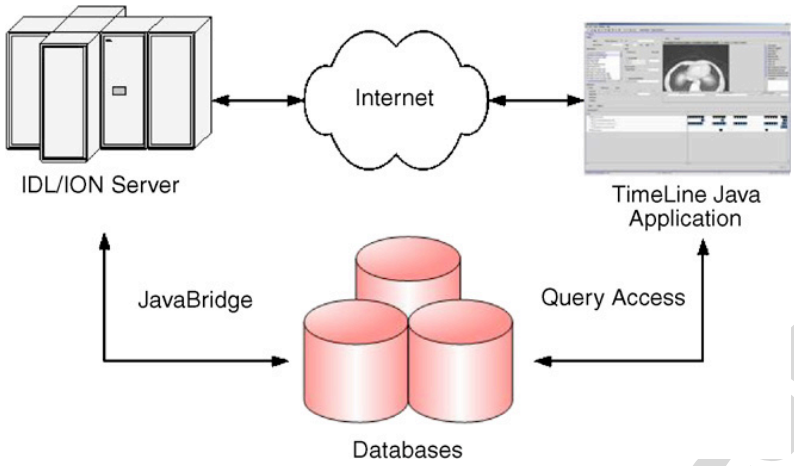

Fig. 8. TimeLine system architecture.

TimeLine interface is written in Java, it should be accessible from all Java enabled platforms.

4.6. Challenges addressed

There are several issues to be dealt with concerning the current visualization implementation clarity: speed, interactivity, and limitation of tools. These issues are directly derived from the issues present for visualization in general.

4.6.1. Clarity

The issue of clarity is addressed through the organization of the interface of the TimeLine application as well as the use of different visualization techniques. The TimeLine interface allows the user to view data from multiple heterogeneous data sources on many different levels at once. The user can see metadata about the data sources from a high level while selecting which sources to view, temporal information about the data from each source, and detailed visualized results from individual data elements. This architecture gives a complete view of data from the high-level metadata to fine points in the query results window. The visualization rendering techniques are used in the TimeLine to allow the user to clearly compare and analyze the relationship between different data elements. Using the methods in Section 3.2, we give the user different options in choosing the most appropriate type of visualization for the data set being viewed.

4.6.2. Speed and interactivity

When the visualization rendering system was implemented, instead of using the Java Bridge to transfer data to and from the IDL/ION server, entire datasets were first retrieved from the data sources by the TimeLine client, then were sent over the ION connection from the TimeLine client to the ION server. This incurred large network latency delays as the data sets sometimes range up to 30 MB. If a client were using a dial-up connection to the Internet, then the delay of transferring this volume in just one direction is over an hour. These kinds of delays are unrealistic for usability. We have remedied this problem by using the Java Bridge to prevent data sets from being sent to the client machine at all.

Another issue that must be dealt with concerning speed is that of rendering time. The IDL rendering speed of 3D IS surfaces are fast and efficient, most likely due to hardware acceleration. However, DVR and other voxel type volume rendering methods are much slower and incur delays of a few seconds for a single scene. This makes semi-real-time rotation and manipulation through IDL and ION not as usable than if the rendering delay were in the milliseconds.

Although ION provides a solution for accessing IDL through a Java interface, it does not provide the real-time interactivity that is available when working natively in IDL. Because all of the rendering is done on the ION server rather than locally on the client system, true real-time interaction with the visualizations is not possible. This includes the ability to rotate views and zoom in smoothly and continuously. The best the ION can do is provide near real-time interaction by providing control parameters, sending them to the ION server and getting the resulting image back. This makes the system somewhat cumbersome to use and not as natural as being able to use the mouse, for example, to rotate and scale a view.

4.6.3. Data set preparation

Because our research is focused mainly on interface and visualization, we have not independently developed facilities to do data preparation in terms of co-registration and segmentation, but instead use existing tools to handle such a task. Furthermore, the TimeLine architecture could be adapted to include such techniques when appropriate. We have also considered data set processing to reduce network transmission latency to remote visualization rendering locations.

5. Visualization querying paradigm and the challenge

The traditional database accessing and interaction paradigm has been such that the user has to specify the particular data sources and the particular location and time instances or period of interest via traditional GIS software packages [32,33] or query languages such as SQL, or the recent TimeLine paradigm introduced above, to access and visualize data for the user to analyze. Unfortunately, very often science questions are such that it is the location and/or the times themselves that are the unknown. In other words, the user knows what data behavior, trends and relationships are of interest and the objective is to find out if they occur and where and at what times. Thus, we envision and are pursuing a future visual and querying interaction paradigm that we call "visualization querying" to meet the challenging need of providing the user the means to first specify the type of data behavior and relationships, such as the one shown Fig. 4, and then have a system that automatically determines what queries to generate to scan the relevant data sources for the visualization specification. This paradigm is radically different and the reverse of the traditional "data source specification and querying first and then visualization for analysis and discovery."

Among the major challenges is to design and develop the overall means to specify the visualization desired, as opposed to the visualization being generated for specific data retrieved by queries. These visual means are part of our current research. Note that all visualization software products operate on specific data inputs; no visualization is performed without first providing the specific data sets.

In prior research leading to the Query visual language [34], we took a first step in this direction as we dealt with queries, such as "Find medical images of patients with tumors

similar in shape and size to the one shown on the screen at the location or vicinity shown in the lung,'' targeted at databases of medical tumor images. Here, the query targets tumors that may appear in the database of images but which have been already segmented to identify the location and characteristics of each tumor in each image (e.g., shape, area, centric, etc.); so the type of pictorial object of interest (the tumor) in the query predicate has been already ''pre-indexed'' in the database for fast access and retrieval. Other project advances in querying by image content also have this limit [35–40]. For example, starting with the pioneering QBIC at IBM that addresses more general image content features, such as ''Find images that have about 60% white color in the lower part and 40% blue on the top'' (like beach scenes), there is a specific set of features (color, shape, textures and location) for which the images in the database have been previously scanned and indexed to support queries.

Unfortunately, in the new scenario of visualization querying that we now introduce herein, we do not have the luxury of a limited range of types of objects (such as 2D tumors) nor of a clearly limited-size database with previously established ''indexes'' for fast access to the objects of interest. We have instead many arbitrary types of objects like the ones shown in each variable plane in Figs. 5–7, such that it is not possible to pre-segment and pre-index the database for the numerous and arbitrary image objects that the user may specify in a visualization querying paradigm.

If we want to find an object that may look like one of the variable planes in Figs. 5–7, then there will be essentially an infinite search space in the data sets (if there are no constraints on the coverage area of the queried temperature profile, geographical area to search in, nor any time intervals to search in) as there are no pre-segmented and pre-indexed such objects in the data sets. We propose to deal with this challenge in several ways: by soliciting from the user specific features of particular interest (such as range of maximum and minimum magnitude of cell intensity) to include in the query to try to narrow the search space (in a way similar to what we did greatly in Query), divide and conquer approaches, and approximate querying and searching. Many queries fortunately may be targeted by the user to specific geographic areas (such as the Caribbean area, Alaska Wildlife Refuge range, the Amazon basin in Peru) and/or more limited points in time (such as within the last two years) and this would certainly reduce the search space to become closer to the problem addressed in our Query advances [34].

Just as there are challenges of visual and pictorial ambiguity in visual languages that have appeared in the literature [34,38,40,41], there are also ambiguity problems in visualization languages. We are currently addressing these ambiguities and issues. The following are some of the challenges we have identified that would have to be overcome, with some of the presented approaches being considered. Suppose that Fig. 5 is a user's visualization input:

- Is the user interested only in the specific locations and time intervals appearing there for the indicated variables? What if this profile is found in a significant geographic part of this area but not the whole area?
- Is the magnitude of each type of data variable (e.g., temperature and pressure in Fig. 5) precisely the only magnitude of interest, or how close to it is it of interest also? What if a somewhat close temperature profile is found, like within 25% of the magnitude specified for most of the locations specified?

- What about if only one variable value distribution specified in the visualization is found to be close enough after database search, should there be a qualified answer to part of the visualization specification?

It is most likely that in this future paradigm the precise visualization specification may not be present, but that a close situation may exist. Thus, we wish the system to automatically detect and inform us what episodes are close to the visualization specification. This may become the main mode of user and system interaction in the proposed visualization-first paradigm. This is indeed a major research challenge for future work. A possible approach to explore is the so-called "cooperative query processing" or "query relaxation" approach [40,42–44], which in the traditional querying paradigm results in database content searching that is "close" to the exact fit specified in the query command. The user "relaxes" the values of attributes of interest in the search, that is, the system searches for values within a small interval the ideal attribute value. Multiple values may be relaxed simultaneously, but then the combinatory of the ranges of the several relaxed attributes to retrieve explodes.

Another major challenge is what we see as the inevitable tremendous amount of database searching that will be required for most visualization specifications. In essence, each visualization query is a major data mining type of query over vast data sources. We are considering methods to search spaces and speed up searching, such as divide and conquer approaches, use of parallel processors to simultaneously search different portions of the data sets, and obtaining from the user various key characteristics about the visualization specification, such as the bell-shape distribution of the data as shown in Fig. 5 or what the average of the temperature over the geographic domain specified should be. Undoubtedly, some visualization queries may exceed the potential benefit to cost ratio that can be afforded with current hardware performance, even if we solve the ambiguity issues above. We view this visualization paradigm as the next huge wave of database searching cycles, as a follow up to the current data mining cycles being consumed.

Visualization querying is a highly challenging area, and at this stage of research we are already identifying significant unknowns and there is no certainty yet of a solution.

6. Conclusion

There have been great strides in the areas of single-data set and multiple-data set visualization. Advances in graphical processing speed, increased network connectivity, and the quality of data sources have created great opportunities to provide users with more powerful environments to visualize data. Our research encompasses these recent advances, integrating cutting edge visualization techniques and coupling them closely to multiple data sources via TimeLine, bringing together methods proposed in many research efforts as well as creating novel techniques. Advances in multiple-data set visualization create new ways for researchers in numerous data domains to view and analyze data. More effective and quicker analyses will give new insights and opportunities to unlock the potential of the increasing amount of captured data. The long-range goal of using the vast interconnectivity of the Internet is to provide the opportunity for users anywhere to have access to the TimeLine system with highly accessible data sources and visualization capabilities.

We address major challenges in multiple-data set visualization including clarity, speed/interactivity, and data set preparation. Clarity is addressed through the TimeLine interface

and through use of new visualization techniques for relating multiple-data sets. Speed and interactivity is an implementation issue that is streamlined by the Java Bridge architecture in IDL to provide visualization at remote locations in light of restricted bandwidth. Data set preparation is an important step in visualizing data sets from different modalities in a single domain. While our research does not extend into algorithms such as co-registration and automatic segmentation, we recognize these are critical to scaling a multiple-data set visualization system to large volumes of data.

Visualization querying is a major challenge to meet requirements of many science queries. Visualization querying is the reverse of the traditional 1–2 steps of "first querying and then retrieving." In visualization querying, the user specifies the kind of output or visualization desired and then the system generates automatically the queries which are then applied to the database to help find the solutions similar to the one provided. We have outlined its challenges and generic approaches as we undertake advancing the state of the art to some day provide full-fledged visualization querying.

References

[1] P.C. Wong, R.D. Bergeron, 30 years of multidimensional multivariate visualization, in: Scientific Visualization, 1997, pp. 3–33.
[2] MIST, UCLA multimedia streams system project, 2003.
[3] H. Hauser, L. Mroz, G.I. Bischi, M.E. Groller, Two-level volume rendering, IEEE Transactions on Visualization and Computer Graphics 7 (2001) 242–252.
[4] H. Hauser, L. Mroz, G.-I. Bischi, M.E. Groller, Two-level volume rendering—fusing MIP and DVR, in: IEEE Visualization Conference 2000 (VIS'00), Salt Lake City, Utah, 2000, pp. 211–218.
[5] H.P. Meinzer, M. Thorn, M. Vetter, P. Hassenpflug, M. Hastenteufel, I. Wolf, Medical imaging: examples of clinical applications, in: ISPRS Journal of Photogrammetry and Remote Sensing 56 (2002) 311–325.
[6] CapeScience, CapeScience AirportWeather, 2003.
[7] B. Conboy, MODIS Web, 2003.
[8] A.F. Cárdenas, R.K. Pon, P.A. Michael, J.T. Hsiao, Image stack viewing and access, Journal of Visual Languages and Computing 14 (2003) 421–441.
[9] A. Ammoura, DIVE-ON: from databases to virtual reality, in: ACM Crossroads Student Magazine, 2003.
[10] S. Shekhar, C.T. Lu, X. Tan, S. Chawla, R. Vatsavai, Map cube: a visualization tool for spatial data warehouses, in: J. Han (Ed.), Geographic Data Mining and Knowledge Discovery, Taylor & Francis, London, 2001.
[11] L.A. Gee, M.A. Abidi, Segmentation of range images using morphological operations: review and examples, in: SPIE Conference on Intelligent Robots and Computer Vision XIV, Philadelphia, PA, 1995, pp. 734–746.
[12] R. Adams, L. Bischof, Seeded region growing, IEEE Transactions of Pattern Analysis and Machine Perception 16 (1994) 641–647.
[13] R. Woods, Automated Image Registration, 2003.
[14] M. Tory, T. Möller, M.S. Atkins, Visualization of time-varying MRI data for MS lesion analysis, SPIE International Symposium on Medical Imaging (MI '01) 4319 (2001) 590–598.
[15] A. Konig, H. Doleisch, E. Groller, Multiple Views and Magic Mirrors—fMRI Visualization of the Human Brain, Vienna University of Technology, Vienna, 1998.
[16] J.B. Kurskal, M. Wish, Multidimensional Scaling, vol. 07-011, Sage, Beverly Hills, London, 1978.
[17] M. Chalmers, P. Chitson, Bead: explorations in information visualization, in: Presented at 15th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, 1992.
[18] C. Chen, Visualization of knowledge structures, in: S.K. Chang (Ed.), Handbook of Software Engineering & Knowledge Engineering, World Scientific Publisher Co. Pte., Ltd., Singapore, 2002.
[19] M. Schroeder, D. Gilbert, J.V. Helden, P. Noy, Approaches to visualisation in bioinformatics: from dendrograms to Space Explorer, Information Sciences: An International Journal 139 (2001) 19–57.
[20] M.Q.W. Baldonado, A. Woodruff, A. Kuchinsky, Guidelines for using multiple views in information visualization, in: The International Working Conference on Advanced Visual Interfaces (AVI '00), ACM Press, Palermo, Italy, 2000, pp. 110–119.

[21] CVEV, Coordinated views in exploration, 2003.
[22] J.A. Brotherton, J.R. Bhalodia, G.D. Abowd, Automated capture, integration, and visualization of multiple media streams, in: IEEE International Conference on Multimedia Computing and Systems (ICMCS'98), Austin, Texas, 1998.
[23] C. North, B. Shneiderman, Snap-together Visualization: Can Users Construct and Operate Coordinated Views?, University of Maryland, USA, 2000.
[24] J.C. Roberts, On encouraging multiple views for visualization, in: IEEE Symposium on Information Visualization 1998 (InfoVis'98), United Kingdom, London, 1998, pp. 8–14.
[25] A. Zwa, Scientific visualization at UMBC, 2003.
[26] D.S. Ebert, R.M. Rohrer, C.D. Shaw, P. Panda, J.M. Kukla, J.C. Roberts, Procedural shape generation for multi-dimensional data visualization, in: Computers & Graphics, vol. 24, 2000, pp. 375–384.
[27] D.S. Ebert, J.M. Kukla, C.D. Shaw, A. Zwa, D.A. Roberts, Automatic shape interpolation for glyph-based information visualization, in: IEEE Visualization Conference 1997 (VIS'97), Phoenix, AZ, 1997.
[28] N. Andrienko, G. Andrienko, P. Gatalsky, Exploratory spatio-temporal visualization: an analytical review, Journal of Visual Languages and Computing 14 (2003) 503–541.
[29] H. Chen, D. Zeng, H. Atabakhsh, W. Wyzga, J. Schroeder, COPLINK managing law enforcement data and knowledge, Communications of the ACM 46 (2003) 28–34.
[30] E. Gantt, E-gantt Software, 2003.
[31] MathWorks, The MathWorks: MATLAB and Simulink for Technical Computing, 2003.
[32] ESRI, ESRI GIS Software, 2003.
[33] GIS.com, Your Internet Guide to GIS (Geographical Information Systems), 2003.
[34] J.D.N. Dionisio, A.F. Cárdenas, MQuery: a visual query language for multimedia, TimeLine and simulation data, Journal of Visual Languages and Computing 7 (1996) 377–401.
[35] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, G. Taubin, The QBIC project: querying images by content using color, texture, and shape, in: SPIE Conference on Storage and Retrieval for Image and Video Databases, vol. 1908, San Jose, California, 1993.
[36] W.Y. Ma, B.S. Manjunath, NETRA: a toolbox for navigating large image databases, in: IEEE International Conference on Image Processing (ICIP'97), Santa Barbara, California, 1997.
[37] B.S. Manjunath, NETRA: image access by color and texture and video content-based retrieval systems, 2003.
[38] T. Catarci, M.F. Costabile, et al., Visual query systems for databases: a survey, Journal of Visual Languages and Computing 8 (1997) 215–260.
[39] M. Daoudi, S. Matusiak, Visual image retrieval by multiscale description of user sketches, Journal of Visual Languages and Computing 11 (2000) 287–301.
[40] M.J. Egenhofer, Query processing in spatial-query-by-sketch, Journal of Visual Languages and Computing 8 (1997) 403–424.
[41] R.P. Futrelle, Ambiguity in visual language theory and its role in diagram parsing, in: IEEE Symposium on Visual Languages (VL'99), 1999.
[42] W.W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, C. Larson, CoBase: a scalable and extensible cooperative information system, Journal of Intelligence Information Systems 6 (1996) 223–259.
[43] W.W. Chu, H. Yang, G. Chow, A cooperative database system (CoBase) for query relaxation, in: Third International Conference on Artificial Intelligence Planning Systems, Edinburgh, Scotland, 1996.
[44] W.W. Chu, C.-C. Hsu, A.F. Cardenas, R.K. Taira, Knowledge-based image retrieval with spatial and temporal constructs, IEEE Transactions on Knowledge and Data Engineering 10 (1998) 872–888.