Towards a Novel OLAP Interface for Distributed Data Warehouses

Ayman Ammoura, Osmar Zaïane, and Randy Goebel

The University of Alberta, Department of Computing Science, Alberta Canada
{ayman, zaiane, goebel}@cs.ualberta.ca

Abstract. We present a framework for visualizing remote distributed data sources using a multi-user immersive virtual reality environment. DIVE-ON is a system prototype that consolidates distributed data sources into a multidimensional data model, transports user-specified views to a 3D immersive display, and presents various data attributes and mining results as virtual objects in true 3D interactive virtual reality. In this environment, the user navigates through data by walking or flying, and interacts with its objects simply by "reaching out" for them. To support large sets of data while maintaining an interactive frame rate we propose the VOLAP-tree. This data structure is well suited for indexing both levels of abstraction and decomposition of the virtual world. The DIVE-ON architecture emphasizes the development of two main independent units: the visualization application, and the centralized virtual data warehouse. Unlike traditional desktop decision support systems, virtual reality enables DIVE-ON to exploit natural human sensorimotor and spatial pattern recognition skills to gain insight into the significance of data.

1 Introduction

The recent rapid development of data mining is a response to the fact that technology enables data collection, classification and storage at a rate far exceeding that with which we can analyze it [10]. To better support the operations usually associated with data analysis and mining, researchers have developed the concept of a data warehouse [11] to model voluminous data in a way that promotes the transformation of information into knowledge. Since vision is by far the human's most predominant sense, many researchers have targeted visualization as the means by which data is presented for analysis [3, 7, 15, 16]. Our proposed system **DIVE-ON** (Datamining in an Immersed Virtual Environment Over a Network) takes visualization a step further by leveraging the human natural skills within an environment that simulates natural settings. Using the sensorimotor skills gained at childhood, one maneuvers through the natural world and acquires spatial knowledge almost unconsciously. To support such natural skills, we have constructed an Immersed Virtual Environment (**IVE**) that uses motion-trackers to acquire movement data, and then simulate the kinesthetic feedback through image transformation. This provides the user with a correlation between orientation and movement, to support a navigation interface that is transparent and

highly capable in examining spatial correlations [3, 15]. DIVE-ON combines advances in virtual reality (VR), computer graphics, data mining, and distributed data warehousing into one flexible system that can be used effectively with little or no training.

DIVE-ON constructs a virtual data warehouse from a set of distributed DBMS systems. Information needed during a visualization session is communicated between the visualization module and the virtual data warehouse as XML documents using CORBA or SOAP technologies. The data warehouse uses a global schema that describes the location of the information needed for building an N-dimensional data cube or any of its subsequently derived subsets. Once the immersed user specifies a particular view, the warehouse queries the individual sources, assembles the resultant cuboid as an XML document, and forwards it for visualization. Here we also present the **VOLAP-tree** (Visual OLAP tree), a special data structure designed to address the demands for real-time rendering and interactive OLAP operations through the recursive spatial decomposition of the materialized "OLAP regions."

Abstractly, DIVE-ON consists of three task-specific subsystems. Figure 1 shows the various layers comprising the complete system, from the data sources to the visualization environment. The first subsystem is the Virtual Data Warehouse (**VDW**) (see Figure 7), which is responsible for creating and managing the data warehouse over the distributed data sources. The second subsystem is the Visualization Control Unit (**VCU**), which is responsible for the creation and the management of the immersed virtual environment (IVE) to insure that the "reality" in virtual reality is not compromised (Figure 1 (3), details in Figure 5).

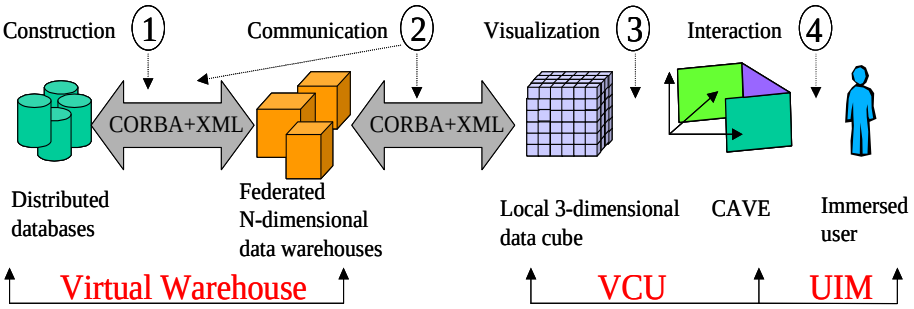

Fig. 1. The three components of the DIVE-ON system

The User Interface Manager (**UIM**) (Figure 1 (4), details in Figure 4) handles the direct application-control interaction as well as the automatic interaction that provides the kinesthetic feedback for navigation and aggregate manipulation. Inter and intra subsystem data exchange is provided by a set of specialized interfaces which implement specific protocols to guarantee extendibility and subsystem independence. This communication takes the form of client and server ap-

plications, using both Common Object Request Broker Architecture (CORBA) over TCP/IP and Simple Object Access Protocol (SOAP) over HTTP.

The rest of this paper is organized as follows. Our immersive display technology is presented, loosely corresponding to a CAVE. We include our motivation for using this environment, and virtual reality in general. We then present the software architecture of the Virtual Data Warehouse(**VDW**), and explain how the XML-based (XMDQL) queries are created and distributed amongst the various data sources.

2 Working In A CAVE

While information gathering and data warehouse management can be done from any location, the actual visualization experience takes advantage of the state-of-the-art virtual reality environment that is formally known as the **CAVE©** theater. CAVE is a recursive acronym (Cave Automatic Virtual Environment) [5], and refers to a visualization environment that utilizes tracking devices along with, up to six, large display screens. Our version places the user within three (9.5 X 9.5) feet walls (Figure2). Each of these walls is back-projected with a high-resolution projector that delivers the rendered graphics at 120 frames per second (Figure3). To simulate the way we perceive depth, the frame rate is divided into a left-eye channel and a right-eye channel (60 frames per second each). These two channels are synchronized with light weight shutter glasses that the user wears to create what is known as *stereoscopic* graphics.

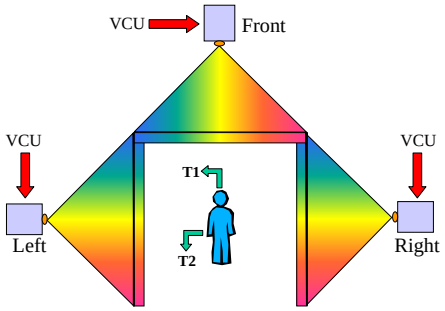

Fig. 2. A CAVE user within the three back-projected walls. T1, T2: The head and hand-held tracker data stream respectively (Real-time)

Using this type of environment for visualization over a desktop is justified by two psychophysical experiences; **immersion** and **presence** [3]. Regardless of how realistic the desktop graphics appear, the user is merely "looking at" a computer-determined point of view [15]. But within the walls of the CAVE the user is presented with an *egocentric* frame of reference which effectively immerses the user into VR. Standing at the center of the CAVE, the available field of view

is a user-centric 270 degree angle. Users can span this view just as they would in a natural setting, by simply turning their heads [5]. Presence is the sense of "being there" and is enhanced by simulating the *kinesthetic* feedback gained while walking through a scene [11]. The user's head location and orientation within the three walls are monitored via a light-weight head tracker. The tracking information is used to update the views using the magnitude and direction of the user's most recent motion vector. Presence is also enhanced by the use of a hand-held wand that is used to perform OLAP operations, probe the data objects, control the environment, and navigate the IVE.

Immersion and presence, when enhanced by peripheral vision and depth perception, are important factors that help improve situational awareness, context, spatial judgment, and navigation and locomotion [15]. As argued by Pauline Baker [3], these factors makes navigation within a 3D model world *practically natural*, and dramatically easier than trying to maneuver around three dimensions using 2D-based desktop controls. Since explorative visualization should be thought of as a task driven and not a data driven process, the next section illustrates how our virtual objects are created in light of what we seek to accomplish.

3 Spatially Encoding Data as Visual Cues

DIVE-ON creates a visualization environment on a conceptual level and, unlike most iconographic data visualization systems, DIVE-ON is not primarily concerned with quantitative measures. For example, the Immersed Virtual Environment (**IVE**) is not designed to tell the user that the total sale of a branch was X dollars; rather it is designed to convey the significance of this amount with respect to its context. Once an "interesting" locality has been identified, the user is capable of extracting the original data lineage.

The primary abstraction of DIVE-ON is based on graphical rendering of data cubes. Selected data are extracted from the VDW (Sec. 4.4) after which relevant attributes are encoded in graphical objects and then rendered in the virtual world. The VCU interprets the three-dimensional cube it receives from the VDW as a three variable function (Sec. 4.2). Each of the three data dimensions is associated with one of the three physical dimensions, namely X, Y, and Z. Since each entry in the data cube is a structure containing two measures M_1 and M_2, the VCU simply plots the two functions M_1(x, y, z) and M_2(x, y, z) in $\Re^3$.

We recognize that there are may alternatives for encoding the data cube measures as graphical objects. Our current prototype uses cube or sphere size and colour to provide the user with visual cues on measure contrasts. For example, if we are focused on the theme "dollars sold" (Figure 3), we assume the OLAP user is not primarily interested in the details that in year t the total sale of product p at store s was \$100,000.00. Instead, the VCU provides a context by associating these measures with visual cues that are bound to the virtual objects. In this case, the first cue we use is *size*, which is associated with the measure M_1 (dollars sold). After normalization, $M_1(x_t, y_p, z_s)$ is used to render a cube (or a sphere) of appropriate size, centered at position (x_t, y_p, z_s), for some t, p, and s within

the data range, as shown in Figure 3. A VR user "walking" among these virtual objects becomes almost instantly aware of the relative significance of each value, without the need for specific numeric data.

Our prototype uses an object's *colour* as a second visual cue, by normalizing a measure M_2, and mapping to a discrete 8-colour palette. For example, we can encode any abstract data mining "interestingness" measure from "red" to "blue." For example, at the lowest level of aggregation (high granularity), colour can represent the deviation from the mean along one of the dimensions. This is particularly useful for market fluctuation analysis. Similarly, if the user is viewing highly summarized data, colour can be a very effective way to locate anomalies at a lower level. For example, the M_2 value for a month object can represent the maximum M_2 of any of the days it aggregates. In Figure 3, each virtual object represents the total revenue for a given year. The colour "red" indicates that one particular month deviates significantly from the rest of the year. We expect the OLAP analyst will *reach* in virtual reality and "select" that object, in order to understand the deviation, and inquire about the exact figures for that year. Similarly, the user may be interested in understanding the stability which dominates a product category (a "blue" object).

(a) (b)

Fig. 3. A team of immersed users discussing the "dollars sold" data cube. (a) Using cubic objects (b) A user pointing the direction of flight within 3D-lit spheres

Figure 3 presents the IVE created by rendering cubes that employ the visual cues described above. In this case, the X-axis (left to right) represents the "product" dimension. The axis pointing in the direction perpendicular to the picture is the "time" dimension Y, while Z represents "location." (The floating 3D interaction menu is also visible.)

Our brief discussion presented cubes as the basic VR geometry, but DIVE-ON can also use spheres in the same way. While spheres can encode the same information as cubes with less occlusion [1], rendering spheres is computationally much more expensive. To create a 3D sphere the system must compute light

sources, normal vector calculations, material specification, and shade rendering. None of these calculations are required for cubes, since the polygon rendering is typically done by hardware.

4 System Design and Architecture

To flexibly support various VR devices, tools, and environments it was necessary to separate the creation from the application of the virtual world. Each of the three main DIVE-ON components is designed within a wrapper that defines the mean for information exchange. In this section the main components that make up the system are presented along with the the task specific design and implementation issues.

4.1 UIM: The User Interface Manager

The User Interface Manager (UIM) is the subsystem that is responsible for receiving, filtering, and channeling all available input streams. Input examples include the location and orientation of the tracking devices, and the button-status on the hand held tracker (Figure 4). This information must be updated at a sufficient rate to provide a natural smooth interaction with the environment. To provide the sense of immersion and presence, the VCU reads the head tracker motion data collected by the UIM (T1 in Figure 2), then transforms the stereo graphics to simulate that motion in a physical world. For example, if the user walks forward, the appropriate image is shifted backwards to create the illusion of "walking" through the data. The data stream emitted from the user's hand (T2 in Figure 2), is used to track the position of the 3D menu in the virtual world. These so-called "floating menus" represent the user's hand to six degrees of freedom (6-DOF) [9].

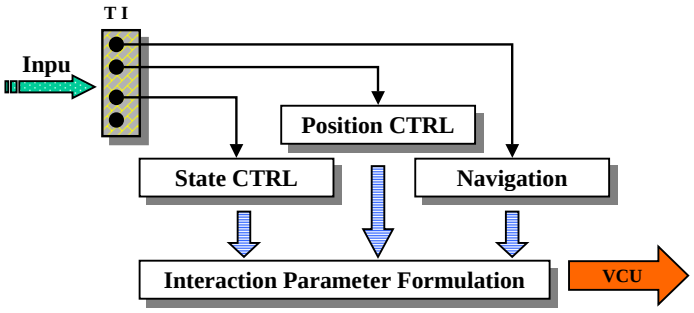

Fig. 4. User Interface Manager. The Tracker Interface (TI) receives the real-time tracker input stream and channels it according to type. The set of interaction parameters is then fed to the VCU

Using the floating menu system, the user is able to perform all application control commands, including various OLAP operations, in a natural manner. For instance, to perform a "roll-up" operation the user activated the menu system, selects "roll-up" and then, using the hand held tracker, points to the dimension to be rolled-up. The operations of "slice," "dice," and "drill-down" are implemented similarly (Figure 3). In a typical session, a client (VCU) first establishes connection to the VDW and, using warehouse queries, the user can then inquire about the number, size and attributes of each data dimensions available.

Viewpoint manipulation is implemented by navigation control. In views that involve dimensions with large domains, the user may request the activation of *flight mode*. In this mode, the user travels through the data by simply pointing in the appropriate direction. Flight speed is determined by how far the arm is extended away from the body. Finally, DIVE-ON also provides the user with the ability to inquire about the original data that is represented by a given virtual object. The hand-held tracker default mode is 3D *virtual pointer*. If an "interesting" data aggregate is encountered, the user can point and pop up an object-fixed window containing the particular aggregate lineage.

4.2 VCU: The Visualization Control Unit

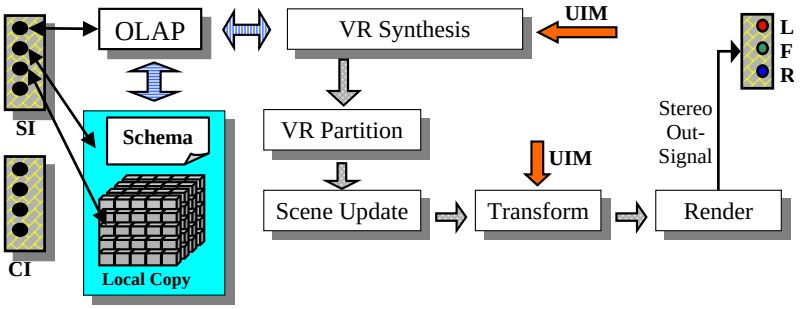

Fig. 5. The VCU Architecture. **SI** and **CI** are the SOAP and CORBA client Interfaces respectively. Output is channeled to Left, Front, and Right projection stereo signals

The VCU is the module responsible for generating and managing the IVE. This makes data visualization and exploration independent, so the specifics of the VDW should be of no concern to the VCU developer and vice versa. To implement this abstract view, the VCU and VDW are each constructed within a wrapper that isolates the only method of relaying messages between the two subsystems. The messages use a simple communication protocol which effectively hides the implementation details and allows the VCU and the DCC to be independent of one another. After the DCC completes the creation of the N-dimensional data cube it signals the VCU (via the DCC-Shell). Since we are generating a 3D virtual world, only three dimensions can be viewed at any given

time (four dimensions is possible by enabling animation using a function we call"animating the data through time" [1]). The three dimensions and associated measures selected by the user are extracted from the N-dimensional data cube, then a 3D cube is passed to the VCU for rendering. In light of the above discussion, what level of abstraction does that the 3D cube represent? If the cube received is already summarized, then for every "roll-up" that the user requests a new 3D cube must be requested. This imposes unnecessary strain on the network and degrades the system's interactivity. For this reason, the VCU builds a working 3D copy that materializes different levels of abstraction into imbedded OLAP regions that are indexed by the VOLAP-tree (Section 4.3).

4.3 VOLAP-Tree: A Spatial Decomposition Structure

Data is obtained from the VDW along with the concept schema and the concept hierarchy that describe the associated aggregation method. To accommodate partial ordering, a concept hierarchy is presented by a simple tree structure with the root being the attribute "ALL" at level 0. The information is then used by the VCU to construct a *working cube* consisting of a set of OLAP regions that provide all possible views (differing granularities) of the user specified dimensions. Each region is a contiguous chunk that can be identified by two vectors. To illustrate using a simplified example, consider Figure 6 which represents a 2D slice of the working cube. The view corresponding to region B (Province/Item) is identifiable by the vectors (X_2, Y_0) and (X_3, Y_1).

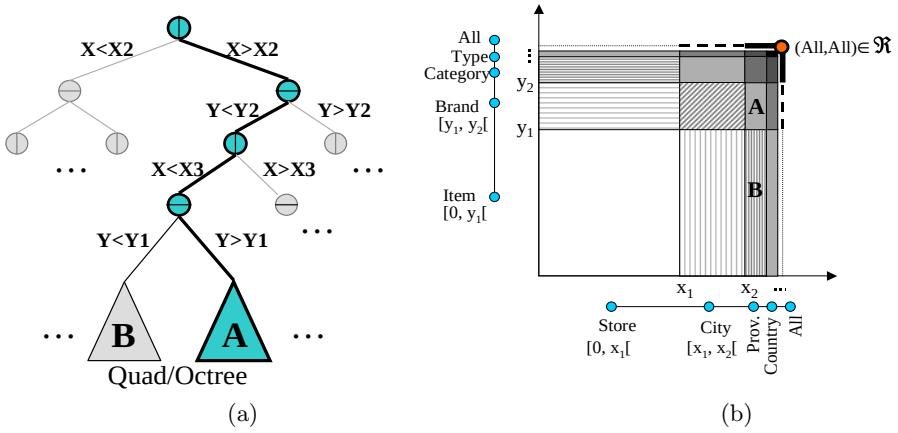

Fig. 6. :(a) The VOLAP-Tree. (b) Materialized working cube within the VCU

The simplest mapping between the user's location in virtual space and the location in the data space is to correlate discretized points in VR with the index

values of the working cube. One can imagine that this slice as the "floor" of the CAVE and when the user maneuvers through VR, the center of the rendered scene is updated to reflect the cell that the user is closest to. This design is closely related to the used metaphor of "walking through the data cube." The efficiency of this design stems from the fact that performing OLAP operation is simply equivalent to "transporting" the user to a different region within the working cube. As an example, consider the two associated concept hierarchies that are shown next the axis that maps them in VR (Figure 6 b). Domain attributes of the dimension "location" at the lowest level, "store," are assigned the index values between $[0,X_1[$ along the X axis. Similarly, the attributes defining "categories" in the "product" dimension are assigned the range $[Y_2, Y_3[$. Assuming that the current view is (Province/Item), region B, specifying a "roll-up" operation on the dimension "product" is equivalent to transporting the user into region A.

The VOLAP-tree is a hybrid that includes a 3D-tree (KD-tree) and a set of Octrees that is designed to quickly transport the user into different OLAP regions while maintaining an acceptable frame rate regardless of data size. Figure 6 (a) illustrates a 2D version of the tree. A 3D-Tree is implemented as an upper layer to index the OLAP regions within the working cube . The root of the VOLAP-tree is the root for the 3D-tree. At the leaf level, each 3D-tree leaf contains a pointer to the second layer, which is an Octree that recursively partitions that particular OLAP region into octants. Recursion continues until all octants at the leaf level do not contain more than a given number of data points. Within the VCU, the "VR Partition" module (Figure 5) uses the the user's location (UIM input) to determine the appropriate set of octree nodes to use for rendering. As the user's position changes through VR, so does the set of rendered octants. When the user performs an OLAP operation the VOLAP-tree is traversed and the new Octree root is located for rendering.

4.4 VDW: The Virtual Data Warehouse

The Virtual Data Warehouse (VDW) is abstract centralized data warehouse comprised of a set of distributed data sources and a shell (DCC-Shell) which is responsible for modeling and querying these sources (Figure 7). The DCC-Shell is maintains a pool of meta-data (*cube schema*) that represents a global multidimensional model of all dimensions and measures available from the distributed sources. When the VDW is initiated, the cube schema is constructed and copied to all data sources. To insure query consistency, the DCC-shell updates all copies when a data source has been updated. A resource allocation table within the DCC-shell maintains the location, data organization, and preferred communication method for each source. The cube schema and the resources data are XML documents for easy maintenance, extendibility, and flexibility.

A client has three available query classes. First is the *Warehouse query*, which provides the client with basic VDW structure including the dimensions of a data cube, the measures available, and the main theme of a cube. The *Cube schema query* provides the meta-data of one specific data cube in the VDW. This meta-data includes a depiction of all available dimensions, measures, and

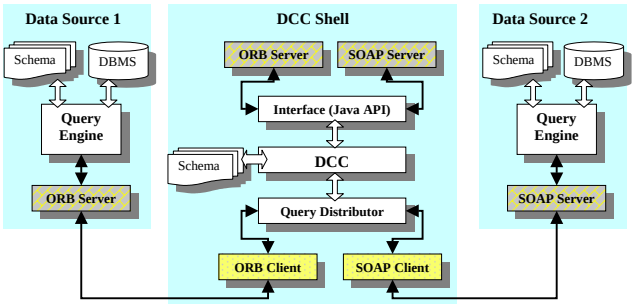

Fig. 7. The Virtual Data Warehouse (VDW) architecture

the concept hierarchy that further describes each dimension. Finally, the *Cube data query* is used to obtain an entire N-dimensional cube or any subset of it. This is particularly useful for applications such as the VCU, which handles only one 3D cube per visualization session.

4.5 XML Multidimensional Query Language (XMDQL)

Our query language choice is an XML-based query language, XMDQL, which we use to interact with the VDW in order to manage and access the available data. XMDQL allows the user to express multidimensional queries on the VDW. The concept of a special multidimensional query language was first proposed as an industry standard by Pilot software [14]. Their language MDSQL, however, does not take advantage of XML and its flexibility and interoperability in the context of federated data warehouses. In OLAP terminology, this type of query is equivalent to slicing and dicing the data cube. The result of an XMDQL query is a cell, a two-dimensional slice, or a multidimensional sub-cube.

DIVE-ON defines XMDQL as a query language that is formatted in XML to query the VDW; it also provides functionality similar to Microsoft's MDX (Multidimensional Expressions). To specify a cube, an XMDQL query must contain information about the four basic subjects: (1) The cube being queried, (2) dimensions projected in the result cube, (3) slices in each dimension and (4) some selection and filtering constraints. The basic form of the XMDQL is as follows:

```
<XMDQL>
   <SELECT>
     Project dimensions and slices
   </SELECT>
   <FROM>
       Which cube to query
   </FROM>
   <WHERE>
     Filtering constrains
   </WHERE>
```

```
</XMDQL>
```

4.6 Query Distribution and Execution

According to Figure 7, a VCU data query is first received by the DCC-Shell interface (through ORB/SOAP Server) and forwarded to the **Query Distributor** after the DCC has formed the appropriate XMDQL query. The Query Distributor analyzes the query, finds which data source contains the required data, and then distributes the query accordingly. Each Data Source executes the query separately, either by translating the XMDQL query into another OLAP query language and getting the result, or by directly accessing the original data source. Regardless of the execution method, each Data Source forms the results as a data cube that is returned to the DCC, which forms an N-dimensional data cube. For our visualization, the DCC then extracts the VCU-requested 3D data cube and sends it to the CAVE for rendering. To illustrate, the distribution information can be stored as following:

```
<Distribution dimension="Store">
  <Component path="N_America.USA"
             mart="DataSource1">USA sales data</Component>
  <Component path="N_America.Canada
             mart="DataSource1">Canada sales data</Component>
</Distribution>
```

5 Conclusion and Future Directions

We have presented a system prototype for visual data mining in an immersed virtual environment. Since the very early days of computing science with extremely limited technologies, scientists have been fascinated with virtual reality (VR). VR systems are capable of abstracting complex problems or scenarios by exploiting the human's natural skills including the visual system and spatial knowledge acquisition. The CAVE theater is a new technology that enables algorithms to interact with the human sensorimotor system. With DIVE-ON, we have focused this technology into a new direction, namely remote visual data mining.

So far we have exploited the human visual system to convey information pertaining to data. In the near future we also plan to experiment with *data sonification* techniques to add audible cues to the IVE. Since hearing is usually a background process, DIVE-ON will use the audible cues mainly to steer the user's foreground process, vision, into a direction that may need in-depth examination. Limiting the use of audible cues in this manner avoids the permeation of the IVE with sensory input that could lead to some undesired perceptual complexities. We also plan to investigate is the use of distortion views, also called fisheye or detail-in-context, in 3D graphics. Creating distortions in the 3D data cube, like creating a virtual magnetic field with a repelling force around interesting

data items, can solve some of the occlusion problems by emphasizing relevant data and putting details in context. However, creating a fisheye effect on local detail in a virtual reality environment without compressing the remainder of the data is not trivial. In addition we plan to merge the interaction operations for distorted views in 3D with OLAP operations, such as aggregating local details, specializing and generalizing on a local detail, etc.

References

1. Ammoura, A., Zaïane, O. R., and Ji, Y., "Immersed Visual Data Mining: Walking the Walk," Proc. 18th British National Conference on Databases (BNCOD'01), Oxford, July2001.
2. Agarwal S., Agrawal R., Deshpande P., Gupta A., Naughton J. F., Ramakrishnan R. and Sarawagi S., "On the Computation of Multidimensional Aggregates," Proc. of VLDB Conference, 1996, pp 506-521.
3. Baker, M. P., "Human Factors in Virtual Environments for the Visual Analysis of Scientific Data," NCSA Publications: National Centre for Supercomputer Applications.
4. Chaudhuri, S., and Umeshwar, D., "An Overview of Data Warehousing and OLAP Technology," Proc. ACM SIGMOD Record, March, 1997.
5. DeFanti, T. A., Cruz-Neira, C., and Sandin, D. J., "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," Proceedings of ACM SIGGRAPH, 1993, http://www.evl.uic.edu/EVL/VR/systems.shtml.
6. Extensible Markup Language (XML): http://www.w3.org/XML/
7. Foley J. and Ribarsky, B., "Next-Generation Data Visualization Tools." In Scientific Visualization Advances and Challenges, chapter 7, pp 103-127. Academic Press/IEEE Computer Society Press, San Diego, CA, 1994.
8. Gary, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., and Venkatrao, M., "Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub-Totals," Proc. of the Twelfth IEEE International Conference on Data Engineering, February, 1996, pp 152-159.
9. Green, M. and Shaw, C. develop MR-Toolkit at the University of Alberta: http://www.cs.ualberta.ca/~graphics/MRToolkit.html
10. Han J., and Kamber M., "Data Mining: Concepts and Techniques," Morgan Kaufmann Publishers, 2001.
11. Hand, C., "A Survey of 3D Interaction Techniques," Computer Graphics Forum, December, 1997, 16(5), pp 269-281.
12. Jaswal, V., "CAVEvis: Distributed Real-Time Visualization of Time-Varying Scalar and Vector Fields Using the CAVE Virtual Reality Theater," IEEE Visualization, 1997, pp 301-308.
13. Keim D. A., Kriegel H.-P.: VisDB: A System for Visualizing Large Databases , System Demonstration, Proc. ACM SIGMOD Int. Conf. on Management of Data, San Jose, CA, 1995.
14. Pilot Software: http://www.pilotsw.com/news/olap_white.htm
15. van Dam, A., Forsberg, A. S., Laidlaw, D. H., LaViola J. J., and Simpson, R. M., "Immersive VR for Scientific Visualization: A Progress Report,"Proc. IEEE Virtual Reality, March, 2000 (VR2000).
16. Ward, M. O., Keim D. A.: Screen Layout Methods for Multidimensional Visualization, Euro-American Workshop on Visualization of Information and Data, 1997.