An Immersed Virtual Environment for Data Warehouse Visualization: Challenges and Implementation*

Osmar R. Zaïane Ayman Ammoura
Database Research Laboratory, Department of Computing Science
University of Alberta
Edmonton, AB, Canada T6G 2H1
E-mail: {zaiane, ayman}@cs.ualberta.ca

Abstract

A system, DIVE-ON, was developed for visualizing and interacting with data from distributed data warehouses in an immersed virtual reality environment called a CAVE. The system provides navigation operations, OLAP manipulations, and data selection and filtering procedures while virtually, and literally, moving through the data. DIVE-ON has three major components: a data collection and communication component, a visualization component and a user interaction component. This paper presents the challenges and implementation choices selected for this system concerning data representation, human computer interaction, and information rendering.

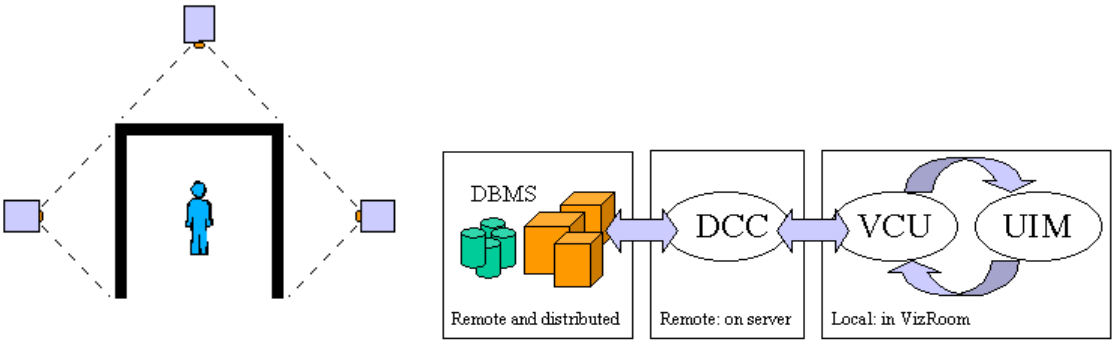
Keywords: Data Mining, Data Warehousing, Data Cube, Data Visualization, OLAP, CORBA, Virtual Reality, Human Computer Interaction, XML, Distributed Applications and Systems.

1 Introduction

Computers have revolutionized a wide range of activities, such as business, banking, publishing, engineering, medicine, etc. All these activities rely upon information and interpretation of information. Data visualization is thus an essential and intricate part of these activities. Advances in computer graphics technologies allow complex systems to be modeled with simple abstractions, making them easier for humans to analyze visually. In the last decade, many tools have been introduced in the field of data mining that enable the creation and manipulation of multidimensional data. The ability to aggregate data across N dimensions[10] opens the door for true 3D data visualization applications [16, 13, 12]. From credit card application processing to international sales analysis, the ability to cluster data and present special anomalies visually has significantly impacted the way data is handled and processed[9]. The combination of a human's ability to analyze visual information, and advances in computer hardware and software provides us with novel ways to visualize multiple large data sets. The University of Alberta is home to an exceptional visualization device, a large walk-in three-walled immersive display (Figure 1a), called VizRoom or CAVE. A smaller three surface display version, called the cavelet, intended to be portable, is also available in the laboratory. These devices are currently unique in Canada and provide the users with the opportunity to visualize and "travel" in three dimensions over collections of data and/or visualized objects.

In this paper a system that builds a data cube of three data dimensions and transports it to the facilities at the University of Alberta for rendering in a 3D immersed virtual reality environment is presented. The application implemented allows a user to interactively select three data dimensions

*Research is supported in part by the Natural Sciences and Engineering Research Council of Canada



(a) (b)

Figure 1: (a) The CAVE with 3 screen-walls and 3 syncronized projectors. (b) The three major components of DIVE-ON.

at a time from a multidimensional data warehouse and visualize aggregated measures at different conceptual levels (i.e. OLAP slice & dice operations). Figure 2 shows an example of interaction with the 3D visual data exploration system.

The DIVE-ON system (Datamining in an Immersed Virtual Environment Over a Network) takes advantage of the immersive display to put on view, to render and interact in a virtual reality environment, with data from distributed data collections. The system can be abstracted in terms of three task-specific subsystems that are integrated to provide all of the required services. The first subsystem is the Data Cube Constructor (DCC), which builds the data warehouse over the DBMS and satisfies incoming information requests. The second subsystem is the Visualization Control Unit (VCU), which provides the IVE (Immersed Virtual Environment). The usability of the entire system is the task of the third subsystem, the User Interface Manager (UIM). Physically speaking, the VCU and the UIM exist locally in the graphics research facility at the University of Alberta (VizRoom) while the DCC resides on a server regardless of its location. A communication layer provides the link between the subsystems as an interface between the VCU and the DCC and/or the DCC and the federated data warehouses. This communication layer could be devised as a CORBA interface over TCP/IP or XML-RPC (Remote Procedure Calls) over HTTP. Currently, only the CORBA interface was implemented. Figure 1b shows the connection between the DIVE-ON subsystems.

The Data Cube Constructor is responsible for extracting structural summary information from the source DBMS or distributed data marts. Its duties include the construction of a data warehouse in the form of n-dimensional data cube and the management of schema and concept hierarchy information. The Visualization Control Unit is responsible for all aspects of IVE creation, update and control. These include displaying data in the immersed virtual reality environment and creating an experience that is as realistic as possible through the use of special data structures and human-computer interaction (HCI) techniques.

With recent advances in computer graphics technology, graphical tools are now available to business to interpret the vast quantity of data generated by their operations. Many software products have introduced visualization packages that handle data mining results. Some of those include, MineSet [16], DBMiner[11], and the IBM Data Explorer[9], etc. All of these tools are designed to be used with traditional I/O devices such as CRT mice and keyboards. This is limiting because a natural visual representation of multidimensional data mining results implies a multidimensional environment. With traditional visualization toolkits, n-dimensional interaction is difficult due to the limitations of the two-dimensional nature of the I/O devices and the sizes of the conventional screens. To make up for the lack of these extra dimensions, other interaction features such as buttons, sliders

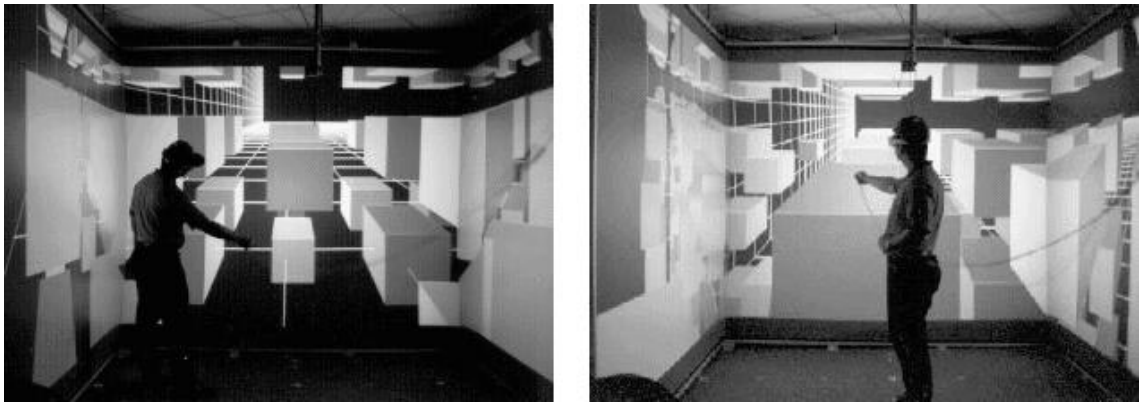


Figure 2: Interaction with DIVE-ON system.

and check boxes need to be provided in order to navigate through the data. These are not necessarily intuitive to learn or to use.

The system we developed provides means for data exploration in a three-dimensional layout. Whereas data could be multidimensional, the system allows the exploration of three selected dimensions at a time. While the gathering and building of information can be done from any location, the actual visualization experience takes advantage of the most sophisticated visualization facility in Canada, at the University of Alberta. This facility, the VizRoom, is powered by an SGI Onyx2 InfiniteReality Rack system with a 4-processor specialized graphics engine. The actual IVE is a physical room, 2.75 x 2.75 m (9.5 x 9.5 feet), with three walls constructed from projection screens 2.3 meters high (8 feet). Three projectors are used to back-project the images onto the three walls at the rate of 120 frames per second (see Figure 1a and Figure 2). This rate is divided in half to produce two sets of identical images, one for each of the right and left eye. These stereo images are synchronized with special shutter glasses that provide a true 3D effect. The result is an environment that provides total immersion in a 3D interactive stereo graphic. The user wears a hat with a tracker to identify the user's location in the room, and holds a second tracker with three buttons allowing active interaction. As will be seen in Section 4, this interaction is divided into two classes: active and spontaneous interaction techniques.

The remainder of the paper is organized as follows: Section 2 presents the data cube constructor component and gives a glimpse on the communication layer. Section 3 introduces the visualization control unit. The interaction and navigation with DIVE-ON is presented in Section 4. Finally, Section 5 conludes and highlights some on-going research issues.

2 Data Cube Constructor (DCC) and Communication Layer

As illustrated in Figure 1b, the visualization and interaction components are local to the VizRoom facilities, while the data to be visualized can be in any remote location. This design is meant to allow flexibility with regard to where the data could be and how it is organized. The DCC is thus situated on a server connected with a collection of data sources. These sources can be databases, flat files, data marts, or complete data warehouses. The DCC serves as an intermediary between the federated data sources and the immersed virtual environment. It collects the specified data from the different data sources and provides the VCU with a centralized view of a collective data warehouse. The connection with the data sources could be in real time, if the data warehouse is not materialized in the DCC, or off-line if a multidimensional data cube is generated on the server. In any case, simulated or materialized, the multidimensional data cube serves as input for the visualization unit.

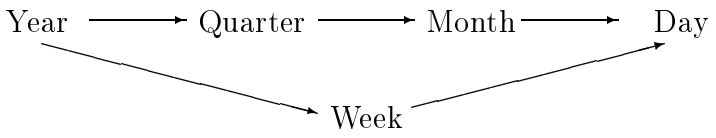


Figure 3: Time Dimension Partial Order.

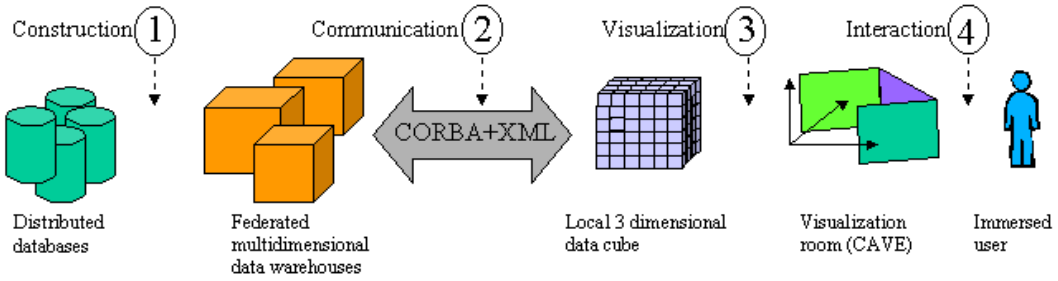


Figure 4: DIVE-ON Architecture.

Figure 4 shows the general architecture of the DIVE-ON system and the different phases between the data collection and the data visualization. In the current implementation of the DIVE-ON system, the first phase, "construction", is not implemented. The system assumes the existence of a MOLAP data warehouse that is constructed on top of the data sources under investigation. The DCC is just a middleware between the warehouse and the VCU answering specific requests such as getting the dimensions, aggregating measures, etc. It interfaces directly with MOLAP engine. Under investigation is a new DCC that would mediate between different distributed data sources via TCP/IP and HTTP to create and maintain its own data warehouse image on the sever. The present version of the DCC receives four types of requests from the VCU as described in Table 1. An initial request asks for the multidimensional logical data model. After the user selects the 3 dimensions and measures to be displayed, a second request asks for the information from the DCC. This same request is used to change any of the three current dimensions or measures. Another request asks for the concept hierarchy of a given dimension. Concept hierarchies are essential for any OLAP operation for they provide the schema according to which the drill-down, and roll-up operations are defined[6]. For illustration, we present two types of concept hierarchies, partial and total ordering. The first type can be viewed as a set S over which the relation R defines partial ordering[1].

In Figure 3 S = {Year, Month, Week, Day, Quarter} and the relation is termed partial since there is no strict ordering between the two attributes month and week. The second type of concept hierarchies can be illustrated by removing the attribute week to produce a total order relationship[1]. The DIVE-ON system implements the drill-down and roll-up operations that are means of data granularity control, which is defined by the concept hierarchy. In the DCC, this information is represented in the form of Meta data using XML. It should be clear that the attributes involved in the concept hierarchy do not necessarily correspond to actual attributes in the DBMS; they are created only as means by which a certain concept about the data can be conveyed.

2.1 The Communication Layer

The DCC unit can be thought of as a wrapper around a data warehouse, or data in general. The wrapper is interrogated for relevant data that is transmitted to another tier, the visualization unit, which stores only data pertaining to the three current dimensions. Since both units are not necessarily located together, they communicate over the Internet with a TCP/IP connection using

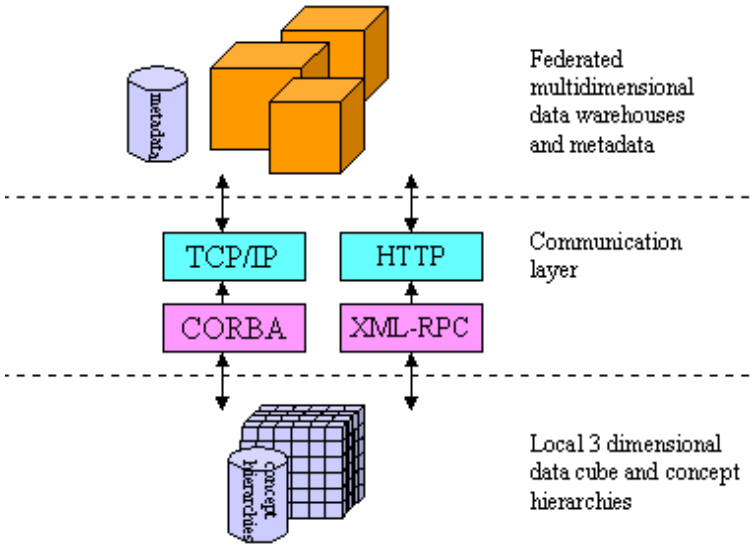


Figure 5: DIVE-ON Communication Layer.

Common Object ReQuest Broker Architecture (CORBA) or HTTP using XML-RPC (or SOAP, Simple Object Access Protocol). This multi-tier architecture provides:

1. Data Independence: DIVE-ON is not coupled with the data source, data application or data type. The VCU could be used for different categories of data; visualizing aggregated measures from a data warehouse or visualizing data mining results structured in a multidimensional space.

2. Simplicity: In order to visualize data with the DIVE-ON system, the data need not move from its original source. The DCC is installed on the data side as a server and it automatically identifies itself and communicates with the VCU (client) with messages in the form of object transmission.

3. Flexibility: The system could work with any data source, from any database vendor, as long as an ORB sever (Object Request Broker) in the case of CORBA, or an RPC daemon, in the case of XML-RPC, is installed with the DCC. When the data source is to be moved onto a different platform, the VCU is kept intact as long as the modified DCC maintain the same protocol.

4. Efficiency: Having separated the DCC from the VCU, the information is passed only as needed and requested by the user. This keeps the graphics system free from any overhead resulting from the computations necessitated by the DCC.

2.2 The CORBA Interface

As mentioned previously and depicted in Figure 5, the communication layer could be implemented with CORBA, XML-RPC, or both. Using XML-RPC as an interface is staightforward. All that is needed is an HTTP server that plays the role of an RPC deamon and a client that records the name of the server. The communication is made by means of XML documents. Despite the simplicity of XML-RPC[18], we opted to first implement the CORBA interface because of its more wide spread

	Request: VCU → DCC	Reply: DCC → VCU
1	Initial warehouse structure	Set of dimensions and measures available
2	$(D_0, D_1, D_2, \{M_i\})$	3D cube for dimensions D_0, D_1 and D_2, with measures $\{M_i\}$.
3	D_i	Concept hierarchy of dimension D_i
4	Status check	Structure that describes the current warehouse status
5	ϕ	Warehouse has changed. Need for update.

Table 1: Requests and responses between DCC and VCU.

use. XML-RPC or SOAP interface would be implemented by the time the conference proceedings are published.

The CORBA Interface can be separated into two parts, the server and the client[3]. These two parts are tightly coupled with the DCC and VCU respectively. In actuality, CORBA interfaces are so tied with these two modules that physically they are the same. These components will nevertheless be logically seen as separate as their sets of tasks are beyond those of both the VCU and the DCC. The information transmitted between the DCC server and the VCU client, in particular the concept hierarchies, are passed in XML format. This allows interoperability with other possible clients and applications.

2.2.1 Server-Side Server Communication layer

This section introduces the implementation strategy used in developing the Server-Side of the Communication Layer. We refer to it as the DCC-Servant. Because only one data source can be visualized at a time, the DCC-Servant only needs to maintain a single CORBA object at any given time. In general only one data source will be present at a physical location, so exactly one servant will be instantiated for each CORBA object (in this case a DCC module). However, in the event that more data sources are accessed, the DCC-Servant could maintain more CORBA objects. Currently, all server objects are transient. In other words, if the system is shut down, state changes are forgotten and object references held become invalid. The DCC operations provided by DCC-Servant are limited to a read-only service.

To extract data from the data warehouse, an API (Application Programing Interface) is needed to provide access to the proprietary database encapsulated by the DCC. The system was designed such that changing from one database system to another would not require rebuilding the entire system. A minimal API has been developed to use in this system. The operations provided by the API allow the DCC-Servant to obtain an array of measure data and a set of schema hierarchies. Facilities to set the current level of abstraction have also been added. The DCC operations that are available to the client are represented in the DCC-Servant interface (Table 1). They are easily mapped to RPC on SOAP.

Once implementations for all the basic DCC-Servant operations have been provided, a main server function is needed to start the Servant. This main function is responsible for initializing the ORB at run time, obtaining a root Object Adapter, creating a servant manager and activating the manager. An Object Adapter is used to provide the communication facilities to the CORBA objects. They provide objects with a way of interfacing with the ORB. Once this step is complete, the DCC-Servant needs to be coupled with a DCC module to connect it to the database. This gives a complete picture of the server-side part of the DIVE-ON system.

2.2.2 Client-Side Communication layer

The Client-Side Communication layer is referred to as the VCU-Client. It is much simpler than the DCC-Servant. Part of this is due to the fact that CORBA provides a template class of objects

known as "_var" types. A _var object is created for each interface defined in the Interface Description Language specification.

The client must obtain a reference to a remote object before being able to use any of the DCC-Servant operations. This is done by first initializing the local ORB. The ORB's services are then used to obtain a reference to the remote object. Currently, this is implemented in DIVE-ON by passing a stringified reference to the client. The client can then use ORB services to convert this stringified reference to an actual reference to a remote object. This reference must then be narrowed to a reference to an object of the desired type. Once a narrowed object reference to the DCC-Servant is obtained, the client can request operations in the same way as methods are called on local objects. CORBA hides all the details of the object's location and implementation. This means that requests to objects in different namespaces or the same namespace will be semantically identical.

The Object Management Group (OMG) has provided a specification for a set of standardized services that must be provided with ORB implementations. These three services are the OMG Naming Service, the OMG Event Service and the OMG Trader Service. The Naming Service allows CORBA programs to obtain object references without the need for them to be explicitly passed to the calling client program. Object references are located through symbolic names. The OMG Trading Service allows clients to locate objects with the help of a program that acts as a trader. This allows the client to perform a dynamic lookup of a service based on the description of the service it needs. The OMG Event Service allows CORBA programs to use distributed callbacks. CORBA programs that do not use the Event Services are generally coupled and communicate synchronously. The OMG Event Service allows for decoupled communication between remote objects. Future versions of the VCU-Client will use the OMG Naming Service to provide location transparency to objects used by the system. This will enable the client to find the DCC-Servant without the need for a stringified reference.

3 The Visualization Control Unit (VCU)

The main functionality of the VCU is to provide an Immersed Virtual Environment (IVE) for data visualization and exploration. To accomplish this task, the Human-computer Interaction (HCI) becomes very important. For the following section of the paper, the discussion is focussed towards computer graphics and is concentrated on issues within this area. First, a brief introduction for the graphical representation of data is due. The next section deals with human factor and shows how it all transparently fit together to immerse the user with their data.

3.1 Virtual Reality and Data Visualization

Before proceeding into the details of the VCU, this section introduces virtual reality and provides the rationale that motivated its use within this application class. VizRoom is CAVE product that was developed by the Electronic Visualization Laboratory at the University of Illinois at Chicago in 1992[7]. The name, CAVE, is the recursive acronym (Cave Automatic Virtual Environment) that was selected for this virtual reality theatre. Unlike most video-arcade type VR applications, the users of the CAVE technology are free from the cumbersome weight of the head mounted display system that involves wearing a helmet. In addition to the awkwardness of wearing a helmet within this environment, the hygienic factors were of more concern to some users. Since the DIVE-ON system is a CAVE application the user wears lightweight shutter glasses and walks around inside the IVE as they interact with virtual objects. The trackers that are used are of negligible weight. Using the CAVE type of VR, the experience is shared by multiple users. This feature is of extreme importance when the application is data exploration because the views can be shared and discussed inside the CAVE where all users can be immersed. All users can interact in real time with one another by pointing and moving around the objects as if they were all in the IVE, a feature that is unique to the CAVE.

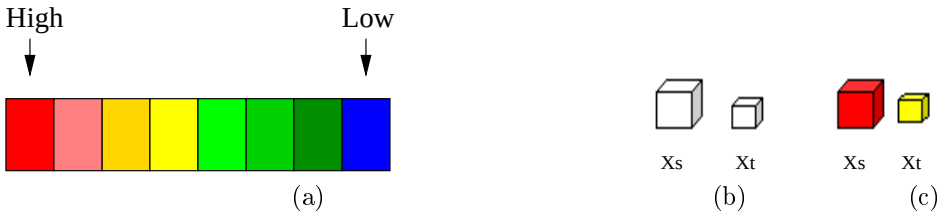


Figure 6: (a) Colour Cue Scale. (b and c) cubes with and without colour.

3.2 Visual Cues and Measure

How can graphics be used effectively to represent three-dimensional data sets along with the measures that specify the correlation among these data items? The idea is fairly simple and involves the identification of effective visual cues and associating these cues with meaning. There are currently two basic visual cues that are used in the DIVE-ON system namely, size and colour. We plan to add sound and motion as well. The virtual world created is three-dimensional by definition. Formally, the IVE can be viewed as the overlapping of the two graphs of the functions $M_1(X,Y,Z)$ and $M_2(X,Y,Z)$ where M_1 and M_2 are the two measures of interest. The three axes X,Y and Z are used to present three of the data dimensions as described by the metadata. The point $M_i(x_i,y_j,z_k)$ is presented by a 3D cube (or a sphere) whose centre is at (x_i,y_j,z_k). The point $M_i(x_i,y_j,z_k)$ is then normalized so that it can be directly mapped to the size of the cube at (x_i,y_j,z_k). It is important to note that although the process of normalization does indeed irreversibly alter the original data measure, the main purpose of visualization has been correctly served. The DIVE-ON system is still able to show, for example, that location X_s in the year Yj has sold \$500,000 of the product category Z_k while location X_t only sold \$250,000 of the same category within the same year by drawing small cubes as in Figure 6b. The first cube is the object at the coordinates (Xs,Yj,Zk) and the second is at (Xt,Yj,Zk). It is the relative size that is used to encode the information conveyed. To incorporate the second measure M_2 into this presentation, colour is used. A colour palette is chosen and the range of normalized values of $M_2(X,Y,Z)$ are discretized and mapped to the palette. For the purpose of illustration, a sample palette is presented in Figure 6a. Assuming that the user is monitoring market fluctuations and is interested in information that may reveal seasonal, regional and product patterns that have an effect on profitability, if the current active view in the IVE presents the three dimensions (D_0 =Year, D_1=Location, D_2=Product) along with the measure total sales, the DIVE-ON system can be used to quickly and effectively pinpoint anomalies that are "buried" down on a low aggregation level. This can be accomplished by employing the colour cue. At the lowest level of aggregation (high granularity) the colour (the second measure) merely represents the deviation from the mean along one of the dimensions. Formally, this is presented by the following equation: $M_2(x_i,y_j,z_k) = (M_1(x_i,y_j,z_k) - \mu$, where μ is the population mean along the time dimension. When the roll-up operation along the time dimension from "day" to "month" is activated, the second measure (M_2) assumes a different role. In the new "rolled-up" view, the M_2 for a month is the minimum M_2 found in all the days it aggregates. Just to demonstrate the effectiveness of this approach, consider the example in Figure 6b. After adding colour as above, a quick inspection of the two objects in Figure 6c allows to pinpoint the relevant information. Accepting the above scenario, although the total sales for the year Y_j of the product category Z_k for location X_s is double that of location X_t one of the months has a great deviation from the rest of the year. This will entice the analyst to pick that X_s object in an attempt to understand the reason. Alternatively, the user may be interested in knowing that a great stability dominates that product category in X_t.

3.3 Challenges in View Presentations

As seen in Section 2, one of the reasons for constructing the data warehouse on top of the target data source is to provide readily available materialized aggregate views; but how useful are these views within the IVE? Notice that the term "view" is in the database sense; it refers to a specific aggregation of attributes using the group-by operator. However, in the remainder of this section about the visualization contol unit, the term "view" refers to an actual view that consists of various graphical element within the walls of the VizRoom.

The VCU presents information obtained from the interface to the user in order to specify the working environment. At the time of system startup, the VCU requests the summary information regarding the data warehouse and its contents. The user then views all the available data dimensions and measures to select three dimensions (for the physical X, Y, and Z mapping) along with the desired measures. Having completed the selection, the VCU submits the request to build the desired data cube in the specified three data dimensions. What is then returned to the VCU is the data cube along with a description of the concept hierarchy for each dimension. This description is used as a map by the VCU to perform the drill-down and roll-up operations across aggregation levels. The interesting question here is how to create this data structure to include the targeted three dimensions, the two measures and all possible combinations of aggregation levels along these dimensions in the same structure.

The approach adopted is to have the DCC build the data cube in the form of a three-dimensional array that includes the data at its lowest aggregation level. The DCC will not perform any aggregation on the data, it will simply extract the information needed from the warehouse, attach the descriptive concept hierarchies in the form of an object and transmit it to the VCU (VizRoom). This approach has been adopted for several reasons including:

- **Transmission**: Even though the data cube can be sparse, if we consider all concept levels for the three dimensions, the data sets can become very large. If each concept hierarchy had N levels of abstraction then a full materialization of all aggregation levels requires N^3 distinct arrays. Computing all these aggregates and then transmitting them across would overload the network and make the system too slow.

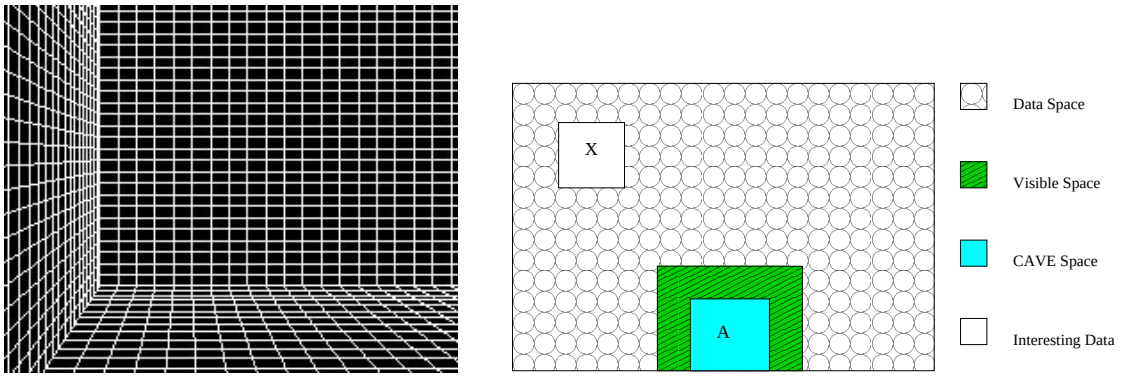
- **OLAP**: If the DCC is responsible for aggregation then any view changes (i.e. drilling-down or dowling-up along the concept hierarchies) initiated by the user would result in having the DCC rebuild and retransmit the entire data sets allover again. In our implementation the three-dimensional data cube is retransmitted only when the user exchanges a dimension with another.

- **Graphics**: As will be seen Section 4, the VCU builds a special local data structure in order to increase the rendering speed of stereo graphics. This process can be made to incorporate all levels of data abstraction. This is much faster than relying on network transmissions.

3.4 Orienteering

Orienteering is one of the most important design considerations in any IVE system. If you were suddenly transported into a foreign environment where everything around you is fundamentally very similar how would begin exploring this new world? Before taking a single step, you must first find some structural clues as to how this environment is assembled. This is done by examining the way objects are arranged within the environment and how they relate to one another. The next natural process involves the identification of some reference points that can be used in maintaining a concept of origin. These are the guidelines used in designing the IVE within the DIVE-ON.

It is fair to assume that the user of the system is familiar with the data that is being visualized. What is needed then is a means by which the layout of the 3D world is mapped to the conceptual correlation in the mind of the user. The system will clearly identify the physical direction of all



(a) (b)

Figure 7: (a)Reference grid in the CAVE. (b)Data space navigation.

dimensions present within the environment. As the users walk within the IVE, they may need to know where they are located with respect to the domain of the three dimensions. For that purpose, a pointer device is used to clearly indicate the (x, y, z) coordinates, in terms of the three active dimensions, within the IVE. In addition, the user needs a frame of reference for perspective and to determine the size of objects in the world[14]. Drawing a grid as illustrated in Figure 7a provides this frame of reference.

This grid also serves as a reference point for drawing the objects. Each cell of this 3D grid can be thought of as the maximum volume allowed for any data item in the IVE (all items are normalized).

3.5 Reality Vs. Scalability

Since the ultimate goal of the VCU is to provide an IVE that is as realistic as possible, image updates in correspondence with the user's motions must be optimized. From earlier versions of the DIVE-ON it was noticed that the graphical rendering speed has a dramatic effect on how the user perceives the immersion within the virtual world. Since each data item is presented by a set of six polygons for the cube (spheres are also possible), the total number of polygons that have to be generated grows rapidly. But the VCU contains a subsystem that uses an OpenGL graphics engine which makes a large percentage of the world rendered invisible to the user as it falls behind clipping planes. To design and implement a visualization system that is scalable, the architecture must support and expect very large data sets. J. Wilhems et al. provide a rich treatment of hierarchical data structures as they are used in volume decomposition[17]. The DIVE-ON system uses a modified octree structure for the purpose of graphics acceleration and aggregate representation.

The octree is a simple tree structure that is used for controlling volume traversal. Each node of the tree represents a subset of the total volume and has a key that relates information regarding the IVE volume that it represents. The actual decomposition of the volume is performed recursively. The starting point of the algorithm is the division of the world into 8 octants and then recursively applying the same algorithm to each octant. Figure 8 shows a cube that is divided into 8 octants and the tree octree presentation of such decomposition. This tree is of depth 1, each recursive call on an octant O will result in the addition of a new level (8 nodes) at node O. Since the volume being decomposed is created by the IVE, the (x, y, z) coordinates are used as a key for traversing the tree. To be specific, the key is the lower left and upper right (x, y, z) coordinates of the octant represented. Choosing such a key is of utmost importance in the DIVE-ON system because of the fact that sparse data cubes are frequent. Using this key format, the octree algorithm will exercise a branch-on-demand method in the recursive algorithm. For example, if after the initial decomposition

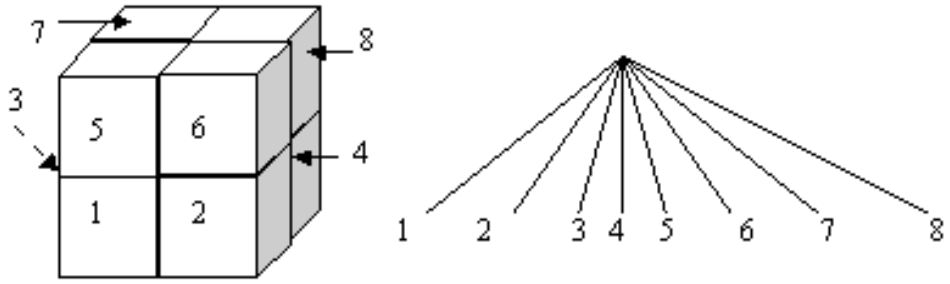


Figure 8: OCtree: division of space in 8 octants.

it was determined that there is no data to occupy octant 8, then there will be no further division for that node.

To use this concept of octrees to represent more that just one volume requires the creation of a special data structure that incorporates both the volume decomposition and the data aggregation. The main idea is to create an octree forest that will not only decompose the IVE volume into selectively rendered nodes, but also have all levels of aggregation readily available within their respective trees. In other words, the VCU will create and pre-decompose views of all possible aggregation levels with respect to the currently active dimensions and measures. The gain here is two-fold. One, the CORBA interface is now responsible for delivering only raw data (low granularity) avoiding a possibly congested network. Two, the user is guaranteed a reasonably fast response time for their inquiries.

4 Interaction: Challenges and Solutions

Using an immersed virtual environment to perform tasks involving analysis and data explorations demands an easy to learn and use user interface. There has been a great deal of research done in the area of user interaction within a 3D environment. R. Lindeman et al tried to provide data that may help IVE designers to produce environments in which "real" work can be done [14].

Navigational interaction remains by far the most challenging. Unlike other uses of VR, the world created by the VCU is inherently large and navigating through this massive amount of information is difficult.

Mice and keyboards are standard input devices for applications visualizing data on a 2D screen. However, there is no standard input device used in 3D interaction with data. We want to provide means for navigation, selection, filtering, comparing, and all major OLAP operations. Good interaction techniques can significantly improve the usability of the visualization system. With DIVE-ON, we use simple trackers, pointing devices and on-screen animation to provide this rich interaction. We categorize the possible interactions into two classes: spontaneous interaction resulting from unprompted actions, and active interaction that involves direct input from the user.

4.1 Spontaneous Interaction

In this class of interaction techniques the user plays a "passive" role in the VCU interaction. In other words, the spontaneous interaction methods are transparent to the user in all aspect and key components of VR creation. The user provides this transparent input by wearing an electromagnetic tracker that creates a data structure containing its exact location with respect to a preset origin within the 2.75 x 2.75 x 2.30 meter volume. This structure is then passed to the graphics engine, which calculates the corresponding relative (x, y, z) coordinates of the device. By attaching this

device to the user's head, the DIVE-ON system induces the user's eye location and orientation. This process is important since creating realistic virtual reality involves the calculation of all perspectives from the point of view of the user. As a result, the user's motion within the room is mapped to an appropriate image transformation on all three walls thus creating a realistic immersion in a virtual environment.

Another electromagnetic tracker is used to maintain information pertaining to the user's hand location. This hand-held tracker is a tool that enables the user to "point", inquire and manipulate the data objects in the IVE. The location of the user's hand is mapped to a 3D pointer that "floats" in the virtual world in correspondence to the user's hand motion. This pointer in conjunction with a time cue creates an effective system input. In terms of OpenGL and computer graphics, attaching a text label to every object in the world is computationally prohibitive and clutters the scene. Instead, the user is made able to inquire about any object by holding the pointer in the vicinity of that object for a short period of time to activate the label associated with it. As will be seen shortly, the hand-held tracker also provides an input command centre through the incorporation of three external buttons.

4.2 Active Interaction

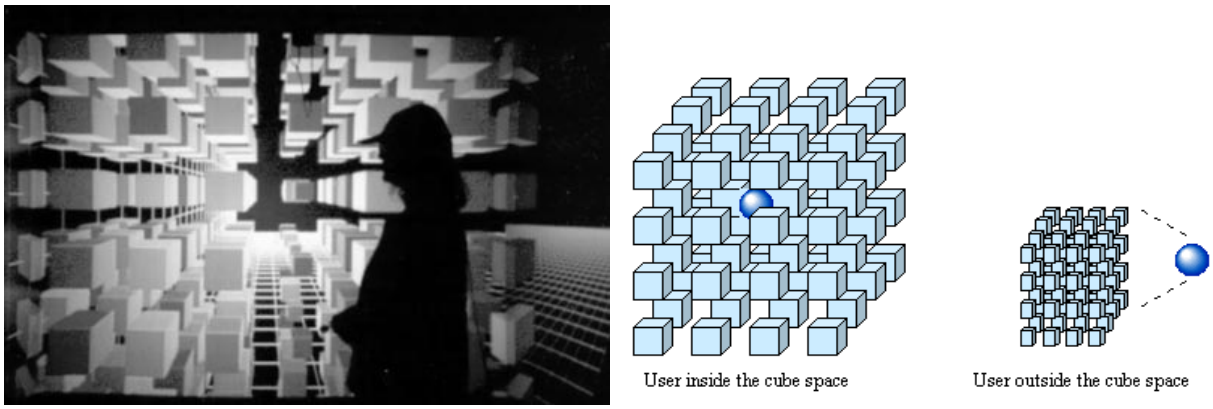
Active interaction refers to the human-computer interaction that is deliberately initiated by the user to fulfill an exact need, request or action. This class includes all navigational functions and menu options. Unlike spontaneous interaction that is transparent to the user and controls a basic IVE functionality, active interaction can be initiated and terminated anytime at the user's request.

Currently, the active interaction is triggered by pressing the three external buttons on the tracker and/or moving the hand-help tracker. The active interaction could also be triggered by voice or hand signs. We plan in the future to install microphones and specialized video cameras in the VizRoom in order to capture voice commands or a limited vocabulary of hand signs. In the current implementation the active interaction is made by pointing on objects, such as the three axes to drill-down or roll-up on a given dimension, or the cubes to read the measures, etc. Other interactions are made by means of a series of cascading floating menus that appear when needed. Notice also that since we have two trackers, one on the head and one hand-held, the distance between the hand and the rest of the user's body can easily be calculated. We use this technique for some interaction related to navigation such as direction, motion, and speed.

4.2.1 The Floating Menu

As the user explores the data presented, there must exist the means for presenting to the user what operations are available along with different means of performing them. A menu system has been developed within the VCU using MR-Toolkit[15]. Through many years of research and development, GUI interfaces have become very successful and are used by a large number of people. Building on that success in a 3D environment facilitates the creation of an effective windowing mechanism that is easy to learn and use.

The hand-held tracker is equipped with three buttons that can be used to pop a menu, scroll through it and execute an option. This process is very similar to the 2D CRT-based working environment. Early experiments with world-fixed and display-fixed windows [8] have proven to be ineffective since the former tends to obstruct the view of data items behind it and the latter is hard to locate due to the size of virtual world. As a result the location of the menu is chosen to be that of the hand-held tracker and moves in correspondence to its motion creating the effect of a floating menu in the IVE. Such a menu can be popped at any time and "tossed" aside while the user continues to examine the data available before making a choice. The menu system enables the user to execute all OLAP, environment, remote and local functions by scrolling and locating the appropriate action. We are also experimenting with a new menu system on a rotating torus. The



(a) (b)

Figure 9: (a)Global vue of the cube space. (b) Navigation vis-à-vis the cube.

torus minimizes the occlusion by the menu while maximizing the options and the usability of the menu. It certainly reduces the number of selections in order to trigger a desired operation.

4.2.2 Navigation

The most important interactive functionality that the system can provide to the user in such an application is the ability to navigate effectively through the IVE. For this purpose many possibilities have been examined, some of which have been implemented and some which are still being fine tuned through experimentation. To illustrate the need and importance of an efficient navigation system an example is presented (Figure 7b). As seen in the previous section, when the virtual world is rendered using the OpenGL engine, only a fraction of the data items are visible to the user[2]. Unfortunately, due to the physical limits of the CAVE, the user is not able to walk throughout the entire virtual world. In Figure 7b, for example, the large area with circles is the total "area" occupied by the data items along the (Z = 0) plane. Each cell of the grid holds a data item. The cyan area marked "A" is the area of the actual VizRoom. That area along with the dashed green area is that which is visible to the user while standing in its centre of the VizRoom. As the user moves within the physical room, A, the invisible area is visually accessible as it falls in front of the clipping planes imposed by the OpenGL engine. After inspection of the visible area, the user determines that the data of interest should exist somewhere within the area marked "X" and would like to make that area visible. This is a typical navigation problem. DIVE-ON presents a few solutions: walking, flying, transporting, and climbing. **Walking** is within the physical limits of the Vizroom. By walking, the user can interact with objects in the area "A", but can't reach the area "X". The most basic navigational means that the user has instant access to is **flying**. In the above example, the user can choose to fly through the "distance" that exists between the two points A and X. In this mode the user is given the illusion that they are flying within the IVE by performing the appropriate image transformations. The best example for this is the train station example. Looking outside the window from a motionless train in a train station, if a close train on a nearby platform starts moving, it will give the illusion to the person on the first train that it is the first train on which the person is that is moving. To perform this operation the user uses one of the buttons on the hand-held tracker and points to the direction of flight. The system can also adjust the flight speed by measuring the distance between the user's hand (hand-held tracker) and head (head tracker); thus, while in fly mode, the user can increase the flight speed by extending their arm in the desired direction.

Transporting is another navigational means that is made available through one of the system's menus. It may be the case that the data being visualized reflects the daily transaction for a 10 year period for a corporation with 200 locations. In a drill-down mode the model may include millions of data items, a situation that makes flying within the IVE slow and renders it virtually ineffective, if not useless. In such situations the user may activate a menu that presents a location indicator on a large scale enabling the user to quickly identify their current location and point to the desired destination. This operation will instantly move the user into the new location. In other words, the user is virtually taken out of the cube for a global view, then back in at a different location. Figure 9 shows an example of a global view of the data cube This view can be rotated on any of the three axes to provide the required perspective.

One of the OLAP operations that can be performed with DIVE-ON is the slice operation, when an attribute value is fixed on a given dimension. A slice of a three-dimensional data cube put on the horizontal plane becomes like a map of city blocks in the virtual reality environment. **Climbing** allows the navigation (up or down) from one slice to another.

5 On-going work and Conclusions

Users may gain a better understanding of the data if they can interact with the data. We have presented a data visualization system implemented on a CAVE infrastructure allowing total immersion of users in a virtual reality environment, without using head mounted displays, to visualize and interact with data from data warehouses. The system allows many users to be immersed in the data at once.

The architecture of the system is divided into three main modules: a data cube constructor, a visualization control unit and a user interface manager. These various modules may be distributed over a network and communicate via object request brokers or XML remote procedure calls. This architecture allows the data to be displayed, to be located anywhere, on any machine that is connected to the internet.

While the current system allows the visualization of data from distributed data sources, there are many enhancements that we plan to add to DIVE-ON.

- One major problem with visualizing data in three dimensions is the problem of occlusion. Data items in the forefront may hide relevant data items in second position or background. This makes the 3D data exploration more difficult. One research issue we plan to investigate is the use of distortion views, also called fisheye or detail-in-context, in 3D graphics[4, 5]. 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.

- Visualizing data mining results and interacting with the results, such as filtering and selecting, can be an interesting application - an augmentation to DIVE-ON. We plan to investigate the visualization of association rule mining, clustering and classification results in the virtual reality environment provided by DIVE-ON, and probably enrich DIVE-ON with new interaction operations specific for querying and interacting with such data mining results.

- We are in the process of enhancing DIVE-ON with an intelligent assistant attached to the rotating torus-shaped menu. With a virtual camera, the assistant could be directed to different locations and provide the user with views that could be displayed for comparison with data on

location. The assistant could also learn from user behaviour and predict user actions in order to prefetch data from the DCC when necessary.

Further developments, addition of visualization and interaction modules in the DIVE-ON system, and related research trials will be reported in the future.

Acknowledgements

The authors would like to express their thanks to Marc Perron for his invaluable help in the implemetation of the system, and to Mark Green and Lloyd White for their assistance with the MR-toolkit and VizRoom facilities.

References

[1] Sameet Agarwal, Rakesh Agrawal, Prassad Deshpande, and et al. On the computation of multidimensional aggregates. In *Proc. Int. Conf. on Very Large Data Bases (VLDB)*, 1996.

[2] Edward Angel. *Interactive Computer Graphics, A Top Down Approach with OpenGL.* Addison-Weseley Longman Inc., Harlow England,, 1997.

[3] Seán Baker. *CORBA Distributed Objects Using Orbix.* ACM Press, Addison-Weseley Longman Inc., Harlow England,, 1997.

[4] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. "distortion viewing techniques for 3D data". In *"INFO-VIS'96: Proceedings of the IEEE Conference on Information Visualization"*, pages 46–53, oct 1996.

[5] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications*, 17(4):42–51, July/August 1997.

[6] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26:65–74, 1997.

[7] C. Cruz-Neira, D. Sandin, and et al. Surround screen projection-based virtual reality: The design and implementation of the cave. In *Proc. Computer Graphics Conference (SIGGRAPH)*, Aug. 1993.

[8] S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon. Windows on the world: 2D windows for 3D augmented reality. In *Proc. UIST'93*, pages 145–155, 1993.

[9] Duncan Galloway, Eric Wolanski, and Brian King. Coastal oceanography data visualization using data explorer. Technical report, The IBM International Foundation, 1996. White papers.

[10] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatro, F. Pellow, and H. Pirahesh. Data cube: A relational operator generalizing group-by, cross-tab, and sub-totals. In *In Data Mining and Knowledge Discovery 1*, pages 29–53. Kluwer Academic Publishers, 1997.

[11] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, G. Liu, K. Koperski, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O. R. Zaïane, S. Zhang, and H. Zhu. DBMiner: A system for data mining in relational databases and data warehouses. In *Proc. CASCON'97: Meeting of Minds*, pages 249–260, Toronto, Canada, November 1997.

[12] Daniel A. Keim. Visualization techniques for mining large databases: A comparison. *Transactions on Knowledge and Data Engineering*, 8(6):923–938, Dec. 1996.

[13] Daniel A. Keim. Visual data mining (tutorial). In *Int. Conf. on Very Large Databases (VLDB'97)*, Athens, Greece, 1997.

[14] Robert Lindeman, John Sibert, and James Hahn. Towards usable VR: An empirical study of user interfaces for immersive virtual environments. In *Proc. CHI'99*, May 1999.

[15] *MR-Toolkit:Virtual Reality, 3D and 2D User Interface Software Tools.* http://www.cs.ualberta.ca/~graphics/MRToolkit.html.

[16] Kurt Thearling, Barry Becker, and Dennis DeCosta. Visualizing data mining models. In *Proc. Integration of Data Mining and Data Visualization Workshop*, 1998.

[17] Jane Wilhems and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[18] Xml-rpc home page: Simple cross-platform distributed computing, based on the standards of the internet. http://www.xmlrpc.com.