ARTICLE IN PRESS

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Visual Languages and Computing
▮ (▮▮▮▮) ▮▮▮–▮▮▮

Journal of Visual Languages & Computing

www.elsevier.com/locate/jvlc

Image stack stream viewing and access

Alfonso F. Cárdenas*, Raymond K. Pon, Panayiotis A. Michael, Jen-Ting T. Hsiao

Computer Science Department, University of California at Los Angeles, 3731 Boelter Hall, Los Angeles, CA 90024, USA

Received 15 August 2002; received in revised form 31 January 2003; accepted 13 March 2003

Abstract

Growing amounts of multimedia data (alphanumeric, image, sound and video) are being captured with the increasing deployment of sensors. Data streams from sensors are being increasingly broadcasted via the Internet. We review the *image stack stream model* or *view* of data as a convenient basis for both querying and visualizing the situation and changes through time of phenomena buried in the multitude of such variety of data streams. We outline briefly the requirements with motivating queries, and the recommended extensions to the latest object database management developments, particularly the ODMG standard, to support this. The image stack view should be provided whether or not data streams are stored in a DBMS. We present a common user interfaces over the heterogeneity of data stream sources to define, access and view stack streams, illustrating it with a testbed of Internet stream sources in the geophysical domain. A systems architecture and design is outlined. Image co-registration is an important element and we provide a brief on our approach. Performance and scalability are addressed.

Keywords: Multimedia streams; Querying and visualization; Stack stream model/view; Evolution over time; ODMG extensions; Image co-registration; User interface

1. Introduction

An increasing multitude of sensors are being deployed to capture growing amounts of multimedia data (alphanumeric, image, sound and video) [1]. The querying and visualization of relevant information from the multitude streams of

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

1045-926X/03/$ - see front matter
doi:10.1016/S1045-926X(03)00022-3

different but related types of data being captured is a major challenge that we address in our R&D efforts. Of particular focus is how to access and present the behavior through time of the various types of related information or phenomena in various data streams.

We recently introduced the image stack model/view [2] as a convenient way to not only visualize different types of data co-referenced to the same location, but also as a basis for querying a snapshot of it, even if the data is not stored as such. We address herein the dynamic *image stack stream model* to support requirements and example queries, focusing on accessing and visualizing change or trend patterns of multimedia data streams of interest through time, and the needed extensions to the latest object database management developments particularly in the ODMG standard to support this.

Significant work has been done in recent times on video indexing and accessing by content and on visual languages [3–6]. However, it has usually focused on viewing individual video streams and not on the emerging multiple heterogeneous sensor stream world and querying that we are addressing.

Geographical Information System (GIS)-related works [7,8] do not envision the image stack view which is at the heart of this paper. Pissinou et al. [7] designed a model that represents spatio-temporal data among geographic objects in geographic video sequences, or digital video. The main objective, the proposed geographical topological-directional model (GTDM) described by Pissinou et al., is to represent the topological and directional relationships, as well as distances among geographic objects that are used for the necessities of the video indexing of raw geographic video. These geographic parameters serve as a metadata basis for storing and querying spatial and temporal data from geographic video database systems. Objects representing geographical areas (e.g. rivers, roads, schools) may be related to Frame objects.

Huang et al. [8] have developed a spatial query language called SQL/Spatial, which is designed to support the expression of user requests dealing with a variety of GIS analysis functions in the Web environment. SQL/Spatial is an extension of the on-going SQL standards for spatial extension. By restructuring the FROM clause of an SQL-type query via a subquery, SQL/Spatial can be adapted to general spatial analysis procedures by using available GIS packages. In our proposed model, we propose to extend the ODMG OQL facility so that similar spatial GIS-type queries (e.g. POLYGON_LINE_INTERSECT) are possible in addition to the generation of intuitive visualizations. However, the major thrust of the extensions we proposed is the extension of multidimensional arrays.

RasDaMan [9] is domain independent array database system for multidimensional arrays of arbitrary size and structure. Its conceptual model centers around the notion of a multidimensional array of arbitrary dimension, extent in each dimension (whereby lower and upper bound can be fixed or variable), and base type. The storage architecture is based on flexible array tiling and compression. Internally and invisibly, arrays are decomposed into tiles, or rectangular parts, which form the unit of storage and access. A multidimensional index is built to help identify the tiles involved in a query and to calculate the costs to retrieve them. RasDaMan is used

primarily for the management of geographical and healthcare data of various dimensionalities. Like RasDaMan, we propose extensions to ODMG for variable-length multidimensional arrays.

2. Multimedia data streams and image stack views and streams

A *stream* is an ordered sequence of frames or values, for example Fig. 1 top left. A frame could be an image, a photograph, a frame in a video stream, a text report, or even an alphanumeric record changing through time. Conventional current DBMS (relational or newer object DBMS) deal well only with alphanumeric record type structures once they are stored and loaded into a database. Unfortunately, it is impractical to store in a DBMS the voluminous data that is coming in increasingly rapid rates from an increasing number of sensors generating such streams; furthermore, there are no DBMS facilities to search for and to select specific elements coming rapidly in a stream of alphanumeric data. We are currently addressing this challenge and the broader challenge of querying and visualizing at selected points in time-specific elements based on the content of the element. For example, select the images with red objects, or select images with tumor sizes larger than 10 cm (a very complex operation involving image segmentation of tumors that is not readily possible today automatically in real nor even near real-time basis) or on content of other elements in other streams (e.g. select photographic image slices corresponding in time to the largest peak value of the seismic stream between time A and B).

It is feasible to store selected substreams in a database, sampled or selected according to various criteria of interest. However, it is not feasible to store in a DBMS much of the voluminous 24/7 data streams; thus we address support in a non-DBMS environment.

We have proposed the *image stack model/view* [2] as a very attractive way to not only visualize and present multiple types of multimedia data but also to set up as a local or user view database to directly support major types of multimedia queries that we illustrate. Fig. 1 shows an example of the image stack which consists of several planes (a stack can have an arbitrary number of planes); each plane with a different type of two dimensionally encoded data, and co-registered to the same coordinate system. This stack would be composed of elements at a point in time from different data streams and other sources.

Fig. 1 shows several data sources from which data at one point in time could be gathered logically or physically and viewed as planes in the image stack, each plane with related alphanumerically recorded data (not geographically encoded):

- Digitized aerial/satellite photos, with a plane for each type of data, for example, poison or pollution levels, and reflectivity/rain levels.
- Static topological maps which are digitized and segmented to identify the geographical location of streets and roads and city boundaries (one plane), of school (including university) districts (another plane), and of contour elevation

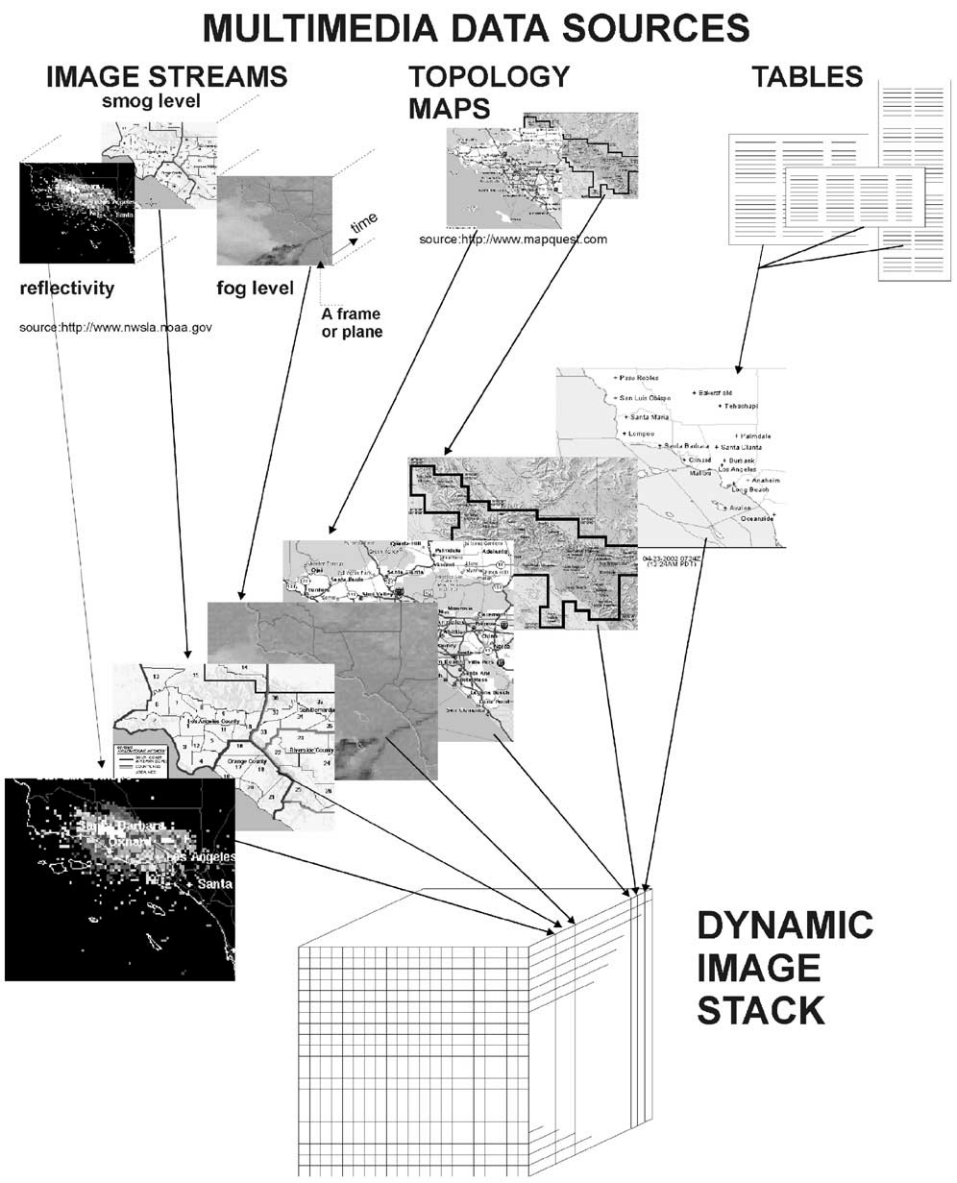


Fig. 1. Logical image stack model/view and distributed multimedia data sources.

lines showing peaks and valleys (another plane in the stack). Each of the planes could be complemented by its associated alphanumeric data that is not geographically encoded; for example, for a school districts plane, the related school name, student population, etc.

- Relational or object DBMS tables with alphanumeric information that is converted to two-dimensional representation. For example, smog numeric data or biochemical or poison fumes data from a number of sensors could be stored in a DBMS and then its geographic distribution presented as a two-dimensional plane of the stack with the grid cell values filled in as a result of interpolation of data between sensors. If this distribution was needed only for a one-shot presentation, then there would be no need to permanently store it physically as a plane in the stack.

Image co-registration and differing grid sizes (resolution) issues may arise in materializing the image stack view due to the variety of sensor sources. Various image processing may be involved in taking a frame from a stream to place it in an image stack. We have a prototype materializing superimposed image stacks addressing co-registration issues, as we illustrate in the last section; however, co-registration is not a major focus of our research.

In many multimedia applications the main interest is in finding out and visualizing what the change and trend of the world order is. We would like to then see image stacks taken at various points of interest and to find out what the change is from one point in time to another, that is, *stream of image stacks*, illustrated in Fig. 2 for a

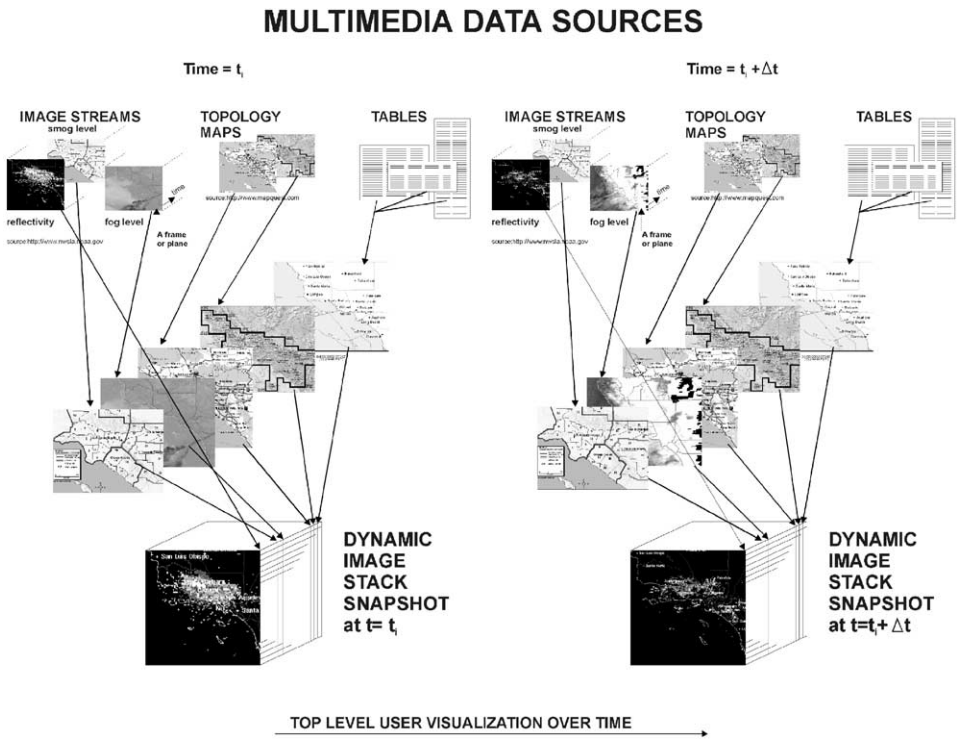


Fig. 2. Stream of two image stacks showing change through time.

stream with two stacks showing change through time. This is the type of queries demanded in high-level decision making, illustrated by motivating queries in a subsequent section.

3. Stack streams and ODMG database structure and extensions

The latest database management system architecture ODMG [10] proposed by the Object Database Management Group provides data modeling and querying facilities not in standard SQL (predominating in today's commercial DBMS world) that are needed towards supporting the dynamic image stack model. These include the definition of lists, sets and bags (collectively referred to as collections) to complement record structures or classes and arrays, and the great ability to intermix these and define:

- arrays of data structures or arrays of arrays or arrays of these collections, or
- a collection of data structures or a collection of arrays or a collection of another collection, and recursively.

The image stack illustrated in Fig. 1 and the *stream* of image stacks in Fig. 2 may be supported by the following database architecture whose merits will be more evident through the wide range of example queries below supported by it. This requires extending the ODMG model to handle two- and three-dimensional arrays and a fourth dimension for time, and the corresponding extensions to the Object Query Language (OQL) [10] commands. We present the following extensions through illustrative examples for reader convenience. The I plane in the stack S at the point in time T is denoted by S (*, *, I, T). The stream of I planes at all points in time is denoted by S (*, *, I, *).

We propose the use of subsets of the array in any dimension. For example, the first M by N points in the x and y domain for I planes at points in time between T_i and T_j is denoted by $S\ (0:M, 0:N, I, T_i:T_j)$. A single cell at point (x_i, y_i) in a plane I at one point in time T_i would be denoted by $S\ (x_i, y_i, I, T_i)$; the situation of this cell for several points in time a, b, c, d would be denoted by $S\ (x_i, y_i, I, T = a, b, c, d)$. Such array handling is not available in current DBMS technology.

Fig. 3 shows the skeleton of the ODMG schema definition for a dynamic image stack stream with the extensions to the ODMG standard needed to support it. The initial part not shown herein would have the class definitions for the database diagram representing roads, city boundaries and school boundaries appearing in a topological image so as to support answering point, line and polygon queries (like queries 1 and 2 below); this would be stored in addition to the original two-dimensional image. Each array cell at one point in time is a class (or record structure) composed of a pixel value and a List_X of classes:

```
class Cell
        attribute Cell_Value; /*pixel value or
```

```
/*Class and relationship definitions for a plane in the dynamic image stack that hold the information on the
layout of roads, school districts and city boundaries, go here*/

class Stack
{       attribute string Stack_Name
        attribute array(x, y, i, t) <Cell> Dynamic_Image_Stack; };
/* ODMG Standard to be expanded to handle from 1 to four-dimensional arrays. Time domain denoted by the
fourth dimension t */

class Cell
{       attribute Cell_Value;    /* data type to be added: bit_value or other data type */
        attribute list<X> List_of_X;      };

class X
{       attribute string X_Name;
        attribute list<X_Attribute> List_of_X_Attributes; };

class X_Attribute
{       attribute string Attribute_Name;
        attribute Attribute_Value;        /* data type to be defined */       };

typedef Plane_Name_I Dynamic_Image_Stack(*,*,I,T)
/* This assigns the name Plane_Name_I to a plane in the image stack, part of 4-dimensional array extension
to ODMG */

typedef array (t) <Dynamic_Image_Stack (*,*,I, t)> Plane_Name_I_Stream
/* This assigns the name Plane_Name_I_Stream to a stream of plane I instances in the image stack, part of
4-dimensional array extension to ODMG */

typedef array (t) <S (*,*,*,T = a, b, c, d)> Stack_Stream
/* This assigns the name Stack_Stream to a stream of four stack snapshots at four points in time a, b, c, and
d, part of 4-dimensional array extension to ODMG */
```

Fig. 3. Partial database schema definition of stack model/view and of stream of instances a stack in extended ODMG.

gray level of an image*/

attribute list ⟨**X**⟩ List_of_X;

where the list List_of_X is composed of instances of the class X with attributes X_Name and List_of_X_Attributes; and the List_of_X_Attributes is composed of instances of the class X_Attribute with attributes Attribute_Name and Attribute_Value, see Fig. 3.

In our proposed model, we envision the lowest or smallest granularity stream Cell_Stream of one-pixel frames as a one-dimensional array of Cells (with t points in time):

$$\textbf{array(t)} < \text{Cell} > \text{Cell\_Stream}$$

The stream of plane I instances in the image stack S along the time domain is envisioned as

$$\textbf{array(t)} < S(*, *, I, t) > \text{Plane\_Name\_I\_Stream}$$

This assigns the name Plane_Name_I_Stream to the stream of plane I instances in the image stack stream S.

A stream of stacks at four selected points in time a, b, c and d (T is the set of these points in time) is denoted in our model by

$$\textbf{array(t)} < S(*, *, *, T = a, b, c, d) > \text{Stack\_Steam}$$

Note that the database approach and structure that we are introducing herein has the ability to optionally store with the pixel value of any image in the stack a list of class or record structures and their attributes along with the actual value for each attribute. Thus, it is possible to store any kind of alphanumeric data particular even at the pixel level, and at any point in time (!), although in most cases such data applies to larger objects composed of many pixels (e.g. a tumor spanning several pixels in an X-ray or MRI image). This is a major flexibility avoiding the schema restructuring restrictions that do not allow such dynamic and open-ended on-the-fly addition of any arbitrary data that may be specific to one or a few cells. Illustrative examples in the following section shed more light on this significant degree of data management functionality down to the level of a single pixel, an innovation we propose.

4. Queries

4.1. Example queries

Below are several motivating sample queries in English leading to our innovations. These queries are typical of the type of major decision making queries that are not automated today by any generalized DBMS, GIS or visualization system, and would take much painful and custom data retrieval, processing and visualization to implement. The requirement for easily accessing and visualizing the situation and changes through time of phenomena buried in the multitude of multimedia data streams is illustrated.

We show the hypothetical extended OQL query incarnation for two queries. The extensions are significant, but are possible to implement in an extended ODMG DBMS. The results of the queries would be sent to an imaging or visualization system for display. Our focus in the article is on the dynamic image stack streams model and some highlights of its database structure model support and query language support needed for the types of queries illustrated below. The focus is not on detailed internal query processing, traditional image processing, or two- or three-dimensional visualization presentation interfaces that are obviously used as the final step to each of the queries below for final visualization of query results; these are beyond space limits herein.

4.1.1. Query 1

Retrieve the location of the intersections of the UCLA boundaries with Westwood Blvd and Sunset Blvd where the poison fume level exceeds value Y now, and compare the fume level to 12 and 24 hours earlier showing clearly the differences in red for higher poison levels and in blue for lower poison fume levels.

Internal query processing to produce the response uses a polygon and line intersection function on x, y coordinate data retrieved from the alphanumeric database section, and then checks that the involved poison fume cell values at precisely these points exceed Y for the latest available poison fume profile. Then the same query is run for the stream world of 12 and 24 hours earlier, and compares the resulting poison levels with the now levels at the corresponding location points to calculate and show the changes that occurred.

Extended OQL query for retrieving the data at the "Now" point in time:

```
SELECT*
FROM Road AS R, School_Boundaries AS SB,
        Poison_Levels AS PL
WHERE      SB.Name = "UCLA" AND
  (R.Name = "Westwood Blvd" OR
  R.Name = "Sunset Blvd") AND
  POLYGON_LINE_INTERSECT(SB, R)
  AND PL.Cell_Value > Y
  AND Time = Now;
/* Poison_Levels plane PL is an image in the stack. PL.Cell_Value checks for the
poison level at each grid point of interest in image PL at time point Now */
```

Next, the above query modified to get the PL.Cell_Value at the locations of interest is run for 12 and 24 hours earlier. The resulting PL.Cell_Value's are then compared to obtain the desired output showing the increases or decreases in poison levels. Note that we thus create a stream of three stacks (one image per stack in this example) specific for this user view and query.

4.1.2. Query 2

For locations with smog levels over X, show terrain elevation with contour lines every 25 ft for places in which there are school districts, showing such smog level in red. Compare the smog level to a year and 2 years earlier showing clearly the differences in red for higher smog levels and in blue for lower smog levels.

This involves thresholding or segmentation, a contouring operator, and a pictorial superimpose operation, followed by a difference operation

Extended OQL query:

```
DISPLAY S.Cell_Value IN RED,
            (E.Cell_Value MOD 25) AS Contours
FROM Smog AS S, Elevation AS E,
            School_Districts AS SD
WHERE S.Cell_Value > X AND
            SD.Cell_Value EXISTS AND
            Time = Now;
/* Contours operator produces lines every 25 feet (MOD 25) */
```

Next the above query but modified to get the S.Cell_Value at the locations of interest is run for one and two years earlier. The resulting S.Cell_Value's are then compared to obtain the desired output showing the increases or decreases in smog levels.

4.1.3. Query 3

Using query 2 as the initial base with the same qualifiers, obtain in addition the alphanumeric information on specific schools within the school districts meeting the qualifiers.

Here we assume that there may be different types of information for each school, and thus for each set of cells bounding the grounds of each school there is specific information. This query shows the requirement to have data associated with groups of cells as an entity, as well as with a specific cell if this large degree of granularity is desired. Note that from the visualization point of view of the user, where and how this data is stored for an object or a particular pixel should be transparent. In the following section we illustrate in diagrammatic form how this information could be stored internally for access.

4.1.4. Query 4

For the East Los Angeles area where smog level is greater than the average smog value X within last year, show an aerial view with smog levels shown in purple, superimposing major freeways, and provide a display (an image in the resulting stack view) of detailed information on population density and ethnic mix for where this information is available in the database. Then compare the smog level to 2, 4, 6, 8 and 10 years earlier and show a stream of smog images/pictures clearly indicating for each smog picture the differences in red for higher smog levels and in blue for lower smog levels compared to the prior period; provide also the percentage change in population density and ethnic mix versus the prior period.

This is indeed a complex query that we would expect to be highly automated by an intelligent multimedia stream data management system embodying the innovations that we propose herein. The system would reissue the first query modified appropriately to retrieve the smog information for the previous periods, and then calculate the differences requested for each point in time.

Alphanumeric population density could be either stored outside the stack with a pointer to it from each pertinent cell in the image stack, or else it could be recorded with each cell as defined in Fig. 3 with an object DMBS with the extensions indicated, or using an implementation (per next section) without a DBMS. This is a matter of access time and storage efficiency which should be transparent to the user.

4.2. Query execution model

The execution of queries is performed on a cell-by-cell basis. A query can be described as a (S, C) pair. S is the stack to which the query is to be executed. C is the condition for which locations of the stack satisfy the query. The following algorithm

describes the execution model:

```
Execution Model (Stack S, Condition C)
Result ← nil
For y = 1 to height of S
        For x = 1 to width of S
                If S(x,y) satisifies C
                        Result ← Result ∪ (x,y)
Return Result
```

From the above query, we can conclude that the run-time order of the query is $O(xyz)$ where x is the width of the stack, y is the height of the stack, and z is the number of planes being evaluated for the query. It is important to note that the $z \ll x$ and $z \ll y$ in many cases, except when the query is being evaluated over many frames of the same stream (i.e. historical query with no bounds), hence run-time then becomes $O(xy)$.

However, if more information about a region of interest is available, we can use that information to limit the search space. For example, if a query refers to on only part of the coordinates such as a strip of a road, and information is available regarding the particular location of that road (stored in an object somewhere in the database), a query can then be executed only along the coordinates of the strip of road, instead of the entire frame. Consequently, the search space is limited to only the area along the strip of road, which is much smaller than the space of an entire frame. Further research regarding such query optimizations is ongoing.

5. Data sources, user interface and visualization

Unfortunately, the growing multitude of available data streams of interest does not follow a common interface standard or Application Programming Interface (API). We propose a common user interface over this heterogeneity to relieve the user wishing to view and visualize image stacks and streams of stacks. Let us address first data sources in our testbed to illustrate the wide variety of stream characteristics, and then provide highlight of the prototype user interface that we have designed and are developing to define and view image stack streams.

5.1. Data sources, testbed example

The system should be extensible such that numerous data sources can be visualized and analyzed in the system. One available data source is the data generated by Next Generation Radar (NEXRAD) [17]. The data generated is a stream of reflectivity images, which displays weather information, such as precipitation and wind based upon returned energy. Two types of data streams are available from NEXRAD. One is a 620×620 GIF image. Also available are streams consisting of 10 GIF images providing an animation of the weather for the previous 4.5 hour, with a frequency 1

GIF image every 30 minutes. Image updates are based upon the operation mode of the radar at the time the image is generated. The WSR-88D Doppler radar is operated in one of two modes—clear air mode or precipitation mode. In clear air mode, images are updated every 10 minutes. In precipitation mode, images are updated every 5 or 6 minutes.

Another available data source is a stream of nighttime fog and daytime reflectivity images generated by Geostationary Orbiting Environmental Satellite (GOES) [18]. Currently there are two GOES satellites in operation. GOES 8 covers the Western Atlantic and eastern half of North America, and GOES 10 provides pictures for the eastern Pacific and western half of North America. Available are views over the earth ranging from the 1 km visible image over Central California to full disk views. The image available is a 640×480 GIF image, generated on the order of many times per day. The image is most useful at night for determining the location of low clouds and fog.

Other data sources include those that represent smog and air quality levels, geographical maps, and traffic conditions. The South Coast AQMD (Air Quality Management District) provides a stream of 670×430 GIF images, updated every hour, that are taken directly from the District's telemetry system [19]. The Environmental Protection Agency (EPA) also provides streams consisting of 575×432 GIF images providing an animation of weather for the previous 24 hours with a frequency of 1 GIF image every 10 minutes [20]. Aerial photos and topology maps are also made available by Mapquest [21] and National Geographic [22], respectively. An XML web service is also made available by the National Weather Service that allows clients to retrieve numerical weather information, such as precipitation, atmospheric pressure, cloud cover, wind, temperature, and visibility. The weather data may be up to 2-hour old, with reports taken once an hour between 50 minutes past the hour and the top of the next hour [23]. The California Transportation Agency (Caltrans) provides real-time freeway speed data via a Java applet [24].

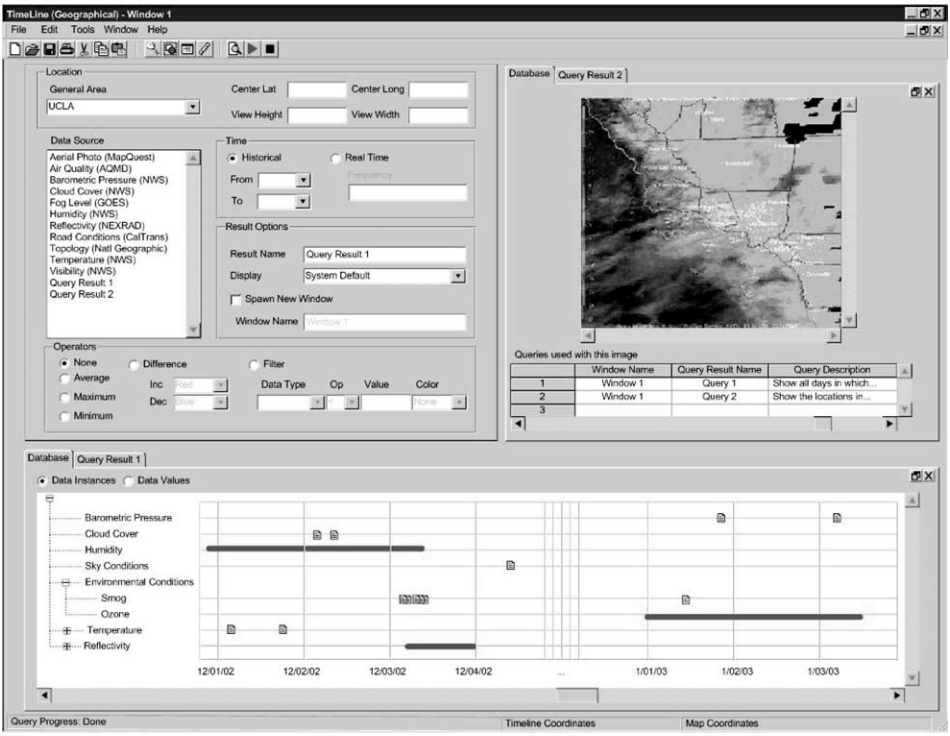
With this multitude of various heterogeneous data sources, a particular low-level interface will need to be designed for each data source. Sitting on top of each of these low-level interfaces will be a common user interface providing a view of the heterogeneous data sources as if it were a single homogeneous source.

5.2. User interface and access

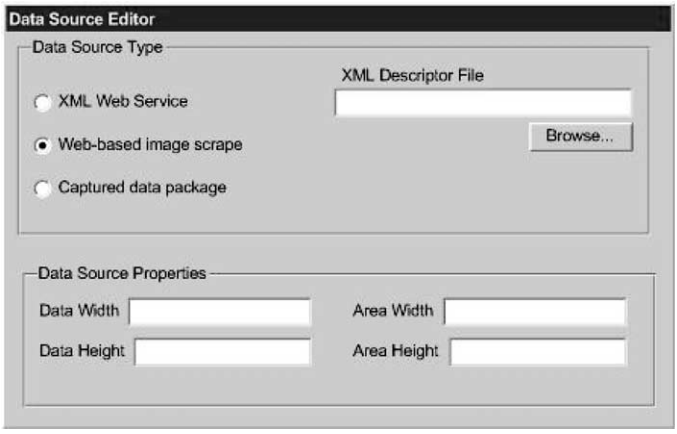
The interfaces to access data streams from sensors and other sources available in the Internet today unfortunately do not follow a common user interface. There are no interface standards even for the same domain (e.g. medical, geophysical). The sources of available image streams, developed by different organizations (as the case is the large majority of times), have a very different look and different level of detail. Ideally, we should have a common or at least rather similar user interface to the world of streams available for users interested in the image stack view/model over multiple streams of geophysical data and for querying through this view. Our approach is to find a common attractive user interface at least for each major

domain. After various designs and experimentations, we have arrived at a highly visual "stream/stack timeline user interface" particularly for the geophysical domain shown in Fig. 4a that we highlight below (inspired in part by the Timeline approach proposed for multimedia patient databases [25,26]).

- The top left area denoted Query editor includes the list of Internet stream data sources that have been previously "registered" for access. Users are able to select a number of sources to build a stack by highlighting the desired sources, and specify the time that the data is gathered, either historical or real time. Users are also able to retrieve details of a source of data by highlighting the source, and clicking on the details button. Details of sources include URL, responsible organization, type of data and definitions, frequency or data rate of images, size and resolution of images, etc. Users are able to define a name for each created stack, and the stack will be added as a source which can be chosen for querying, thus allowing queries that build upon other queries. The lower part entries, query result 1 and query result 2, are examples of previously created stacks, which can be then selected as a source for a future image stack views. Clicking on an Internet data source will show on the right-hand side of the timeline screen the live data produced by the source, and clicking on query result will show the image stack defined and materialized. Users are also able to spawn newly created stacks in a new window, thus allowing for side-by-side comparison of multiple image stacks. Aggregate operators are also supported with color filters, allowing the users to better visualize variations between different stacks.
- The right side of the interface shows the real-time data feed from an Internet stream source; or an image frame of a stored stream at a point in time, or the image stack view previously specified and developed and shown at a specified point in time, or a dynamic cine sequence of the previous images.
- Above the data sources within the Query Editor area is an indication of the location of the data and the latitude and longitude covered by the particular data source or stored query selected. To the right of the data sources is an indication of the interval of time for which the data source is desired, assuming it is in archived form in a file or data base, or that real-time live feed is to be shown on the right side of the interface.
- The lower part of the timeline under the tab Databases shows visually for each named data source or query the existence of the corresponding data for the indicated points in time or time intervals shown horizontally. A solid line in the timeline indicates that the data actually was archived for the time interval shown (e.g. humidity). A document icon or a stack icon rather than a solid line indicates the specific dates for which the particular data is available. Pulling the Query Result tab will show the stored queries that are accessible for the points in time or time intervals visually shown. Clicking on a document icon will display on the right side the image details. Users are able to build an image stack by selecting a set of data instances from the timeline view. This gives the users a visual representation of the selected data to build an image stack.



(a)



(b)

Fig. 4. (a) Timeline user interface for stream sources and image stack stream viewing. (b) Data source editor to register stream sources for access via the stream/stack timeline interface of (a).

- Moving the slider knob horizontally below the timeline will shift the timeframe involved forward or backward chronologically. The timeline will show for what points or intervals data is available in the sources that have been registered for access in our interface.

Fig. 4b shows the Data Source editor window that we are developing for registering stream sources for access via our stream/stack timeline interface, Fig. 4a.

We foresee that in the future the proliferation of stream sources available particularly via the Internet with different user interfaces will necessitate and may force industry standards to achieve some user interface commonality. The image/stack timeline proposed herein is a visual and dynamic thrust towards this end in the geophysical data stream domain, in parallel with the Timeline proposed for the medical/health care domain [25,26].

6. System architecture, design and implementation

6.1. System architecture

The image stack stream view is not supported by current database management systems nor by specialized GIS, and thus we present above the object DBMS extensions needed to be able to provide it. We will now outline the system architecture and design to accomplish it, which is the basis for an initial proof of concept implementation.

First, we point out that the data streams that we are dealing with are not necessarily going to reside in a conventional DBMS before or after the image stack view is provided. This assumption is necessitated by the fact that the large majority of sensors will be too voluminous and will use the Internet as the primary means of broadcasting and providing such data. Fig. 5 shows the system architecture. If an object-relational DBMS or a new ODMG DBMS version becomes available with the extensions recommended above using the OQL base, then its situation is shown as the path on the right in Fig. 5. Data from heterogeneous data sources are captured and then stored in the DBMS. Once the data is stored in the DBMS, the data may be compressed or decompressed, co-registered manually or automatically. The results of compression, decompression, or co-registration may be stored back into the DBMS so that when the stack view is generated at a later time or is generated frequently, such processing does not need to be done over again. The DBMS path would be used in cases in which the user wishes to examine historical data.

If no DBMS is involved, then we can take the underlying streams directly through a real-time stream processing phase and then provide the image stack stream view, as depicted on the left path. Unlike in the DBMS path, in the real-time stream processing phase, the streams are co-registered with one another with less accuracy. In this processing phase, the co-registration process will need to be fast enough so the latency between the capturing of data and the generation of the stack view will have

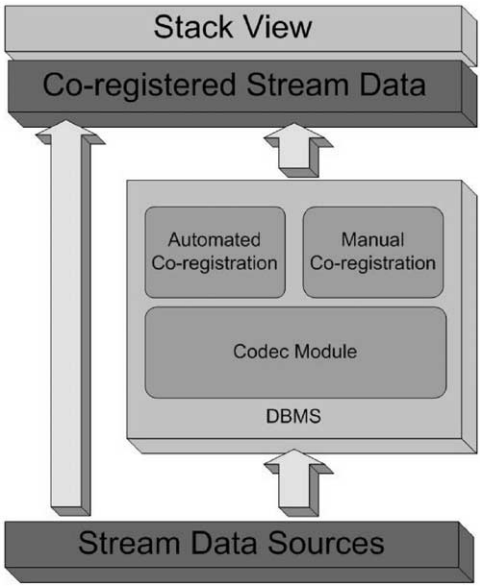


Fig. 5. System architecture diagram.

to be low. As a result, we can expect that such a co-registration may not be perfect. We expect that the deployment of sensors gathering a growing number of data streams are going to exceed DBMS storage capacity and thus we must provide the image stack view without use of a DBMS.

Both (DBMS and real-time) stream paths facilitate the capturing of data from heterogeneous data sources. The paths have knowledge of the various interfaces to the different data sources. Once the data is captured, the data is presented to all the other modules in the system architecture in a uniform manner. As stated before, the real-time stream path co-registers the data so that it maybe viewed in the stack view immediately, unlike DBMS stream path, which only presents the data to the DBMS for any further necessary processing and storage. Furthermore, the DBMS stream path must also selectively determine which data items should be stored in the DBMS since the data source may provide an infinite stream of data, which cannot all be possibly stored. On the other hand, the real-time stream path allows most (if not all) data to pass to the stack view.

6.2. Stream co-registration

We use the Java Advanced Imaging (JAI) technology [11,12] to implement the ability to take different streams of data, pick frames at any time, and superimpose with co-registration the frames to materialize an image stack. The co-registration step is significant [13] and is needed to properly superimpose images in the stack. It is preliminarily implemented as a user-defined mapping of an image's coordinate system to another image's system. Four geographical landmarks are identified by the

user of the image corresponding to those of another image, as inspired by co-registration done in GIS work. Manual co-registration is acceptable in the case where not many images need to be co-registered with one another. In this case, this is true. For example, it is expected that the number of different streams to be looked at is much smaller than the total number of images available in a stream (i.e. the stream may be infinite). If we were to co-register representative images from two different streams, we can apply the same co-registration information that is generated by the manual co-registration process to all the subsequent images in the two streams automatically. Furthermore, this method of co-registration would be adequate if the pixel-to-world scales of the two raw images are approximately the same. However, the problem of co-registering two images, where one image's pixel represents a distance much greater than another image's pixel, is a much more difficult to solve. For example, if the width of one image's pixel may represent 10 km in the Universal Coordinate Space and the width of another image's pixel represents 10 cm in the Universal Coordinate Space, the error involved in co-registering these two images would be quite significant. As a result, a much more sophisticated co-registration process may need to be explored in ongoing research.

Fig. 6 shows a superimposed image stack composed of (a) a frame from an ozone-level image stream over the Los Angeles and vicinity topology, with county boundaries superimposed, Fig. 6a, and (b) a frame from a traffic conditions data stream, Fig. 6b. These data streams are dynamically changing through time, updated several times a day and broadcasted via the Internet. The superimposition of these two frames results in Fig. 6c showing the ozone levels and traffic conditions information for the area simultaneously. Images from the two streams are co-registered semi-automatically in our implementation. The user indicates four points in common in the two images, visible as crosses in the two frames in Fig. 6a and b, which serves as the base points for the rest of the co-registration process to take over and automatically line up the Los Angeles and vicinity coast boundary of the two images on top of each other toward the result Fig. 6c. This quadrilateral-to-quadrilateral mapping warps images so that they are co-registered to one another, regardless of the image size and orientation and the perspective at which the image was taken.

Within this co-registration module, we define a Relative Coordinate System assigned to each picture frame. We also define a Universal Coordinate System Frame. The Universal Coordinate System Frame has a Universal Coordinate System assigned to it. Each one of the images is co-registered to this hypothetical Universal Coordinate System Frame over their (x, y) information via a transformation. This methodology is along the lines of the methodology used by various GIS and rubber-sheeting software packages.

More complex and automated methods of co-registration are possible. Chen et al. [14] propose an efficient global optimization for image registration. Work at the University of Texas at El Paso has also spawned an automatic image registration based on Fast-Fourier Transform [15]. Sester et al. [16] also describes three approaches for the linking of objects from heterogeneous data sets, such as cadastral, topographic, geologic, and environmental data, which can be used to help solve the

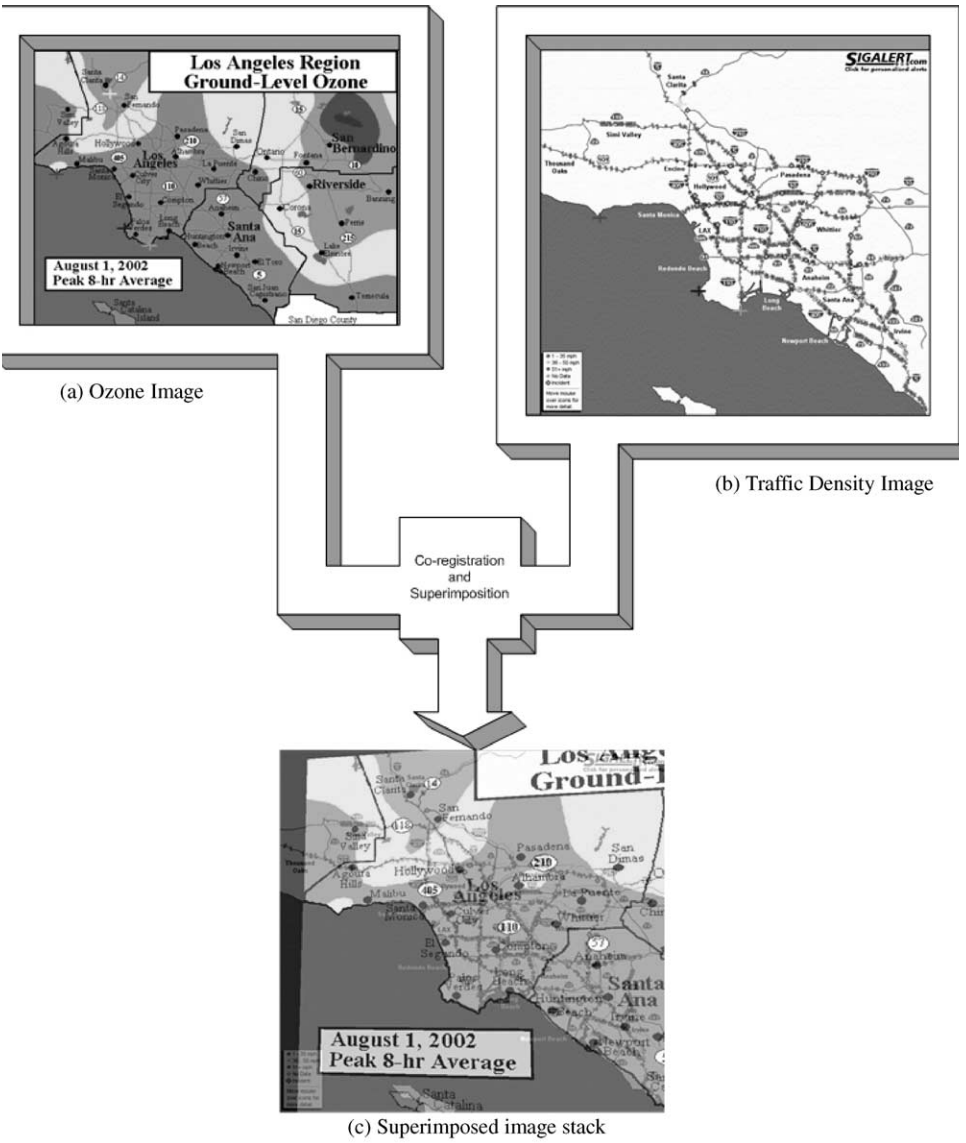


Fig. 6. Superimposed image stack (c) at one point in time for Los Angeles and vicinity, based on image frames (a) and (b) from two continuous image streams broadcasted in the Internet.

co-registration problem. The first defines the linking as a matching problem and aims at finding a correspondence between two data sets. The two other approaches focus on the derivation of one representation from the other one, resulting in the automatic generation of new data sets. Correspondences can be established at the level of individual object instances, at object class level, or at the geometry level. Future work may adapt some of these more advanced co-registration methods.

7. Performance and scalability

7.1. Space–time performance analysis

From the previous section, we see that the run-time order of the query execution model presented is $O(xyz)$ for an image stack, and $O(xyzt)$ for a stream of stacks spanning t time intervals. We can also come to a similar conclusion with the space order for storage of the stack. To store a single stack of z frames with width and height of x and y, respectively, requires xyz pixels of storage. If a single pixel requires a constant p amount of space, then the total stack requires $pxyz$ space. To account for the overhead of storage in an OODBMS, which is a constant c for each pixel, we will also require an additional $cpxyz$ amount of space. We can reduce the space order to $O(xyz)$. As seen in the previous section, the size of an image may be as large as 640×480 pixels, so x may be 480 pixels wide and y may be 640 pixels high. For query 1, z would be 1, since the query only uses the most recent poison levels plane. For query 2, z would be 3 because the query is executed over the planes describing elevation, school districts, and the most recent smog levels.

When images are first introduced into the system, they are in a compressed format, taking much less space than its uncompressed analog. However, to make use of the Image Stack model, we must decompress each image so that we can directly manipulate and examine each pixel. The worst-case storage requirement as we have seen from above is proportional to the size of the image. However, if we were to create Stack Views dynamically as queries are issued and executed, it may be possible to sacrifice timeliness for query results for storage space. The performance analysis for such a method is an issue to be explored further in future research.

Generation of the query execution plan from OQL queries is ongoing research. But it should be noted that the time and space needed to generate the query execution plans should be much lower than actual execution of the query.

7.2. Stream scalability

The heterogeneity of streaming data and their sources makes developing applications that use these types of data difficult. A common interface for such applications is needed so that applications see streaming data from many different sources from a common interface. This architecture allows for extensibility when adding new data sources. Since the development of the top-level application is independent of the existence and interface of these individual data sources, adding new data sources will not affect the design of the top-level application.

Furthermore, since many of these data sources may arrive at high rate and much of the data is not needed (e.g. containing already seen data), we may not need to store and/or visualize every data item over time. In our first phase of implementation, we currently have implemented a tool that will extract frames from an MPEG stream within ± 1 frame of the desired frame. This allows us to extract frames at a frequency much lower than that of the original MPEG stream (e.g. 25 frames/second), discarding frames that do not contain much new data.

Since streams of data may be infinite, some type of automated co-registration process will be needed. In the current implementation, we can manually co-register two images. As stated before, the co-registration information generated by this manual process may be saved and applied to all future images in the same stream that may be seen later in the future. Thus, co-registration of a whole image stack without significant manual work is possible using this method.

8. Conclusion

We are currently progressing with further development and implementation of the image stack model view and of streams of stacks over a multitude of multimedia streams. The intent is to provide such an image stack view through an attractive common user interface, whether or not data is actually physically stored as such, and independent of whether or not a DBMS is used at all. We highlight herein proposed extensions to object database management developments particularly the ODMG standard that we consider feasible. We are investigating and developing a prototype in Java, and without use of a DBMS as much of the data streams being generated are increasingly being broadcasted via the Internet and will not reside in a DBMS.

Acknowledgements

This research work is partially supported by National Science Foundation Grant # IIS 0140384, "Multimedia Stream Modeling, Relationships and Querying." Thanks are due to He-Joon Kim who as a Junior and Senior at UCLA assisted in testing of building image stacks and co-registering frames from real-time streams, and to graduate student Bassam Islam for supporting our effort on multidimensional array database management. Our appreciation is also given to the anonymous reviewers of this article.

References

[1] A.F. Cárdenas, A 2025 scenario and vision on stream data modeling, in: S. Brinkkemper, F. Lindencrona, A. Solvberg (Eds.), Information Systems Engineering—State of the Art and Research Themes, Springer, London Great Britain, 2000.

[2] A.F. Cárdenas, P.A. Michael, B.S. Islam, Stack database model/view of multimedia data, Proceedings of the 2002 International Conference on Information and Knowledge Engineering, Las Vegas, Nevada, June 24–27, 2002, 6pp.

[3] M. Flickner, et al., Query by image and video content: the QBIC system, Computer 28 (9) (1995) 23–32.

[4] S. Chang, W. Chen, et al.,VideoQ: an automated content based video search system using visual cues, ACM Multimedia (1997) 313–324.

[5] S. Srinivasan, D. Ponceleon, et al., CueVideo: automated video/audio indexing and browsing, Proceedings of the ACM SIGIR '99, Berkeley, CA, 1999.

[6] T. Catarci, M.F. Costabile, et al., Visual query systems for databases: a survey, Journal of Visual Languages and Computing 8 (2) (1997) 215–260.
[7] N. Pissinou, I. Radev, K. Makkai, Spatio-temporal modeling in video and multimedia geographic information systems, GeoInformatica 5 (4) (2001) 375–409.
[8] B. Huang, H. Lin, Design of a query language for accessing spatial analysis in the web environment, GeoInformatica 3 (2) (1999) 165–183.
[9] P. Baumann, A. Dehmel, P. Furtado, R. Ritsch, N. Widman, The multidimensional database system rasdaman, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, June 1998.
[10] R.G. Cattell (Ed.), The Object Database Standard: ODMG, Release 3.0, Morgan Kaufmann, San Francisco, CA, 2000.
[11] Sun Microsystems, Programming in Java advanced imaging, Release 1.0, July 1999, [Online] Available: http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1 guide-unc/index.html
[12] Sun Microsystems, JAI API [Online], Available: http://java.sun.com/products/java-media/jai/forDevelopers/jai-apidocs/index.html
[13] F. Lee, S. Bressan, B.C. Ooi, Hybrid transformation for indexing and searching web documents in the cartographic paradigm, Information Systems 26 (2) (2001) 75–92.
[14] Y. Chen, R.R. Brooks, S.S. Iyengar, S.V. Rao, J. Barhen, Efficient global optimization for image registration, IEEE Transactions Knowledge and Data Engineering 14 (1) (2002) 79–92.
[15] H. Xie, N. Hicks, G.R. Keller, H. Huang, V. Kreinovich, Automatic image registration based on a FFT algorithm and IDL/ENVI, [Online] Available: http://www.geo.utep.edu/pub/xie/Auto.pdf
[16] M. Sester, K.H. Anders, V. Walter, Linking objects of different spatial data sets by integration and aggregation, GeoInformatica 2 (4) (1998) 335–358.
[17] National Weather Service, Los Angeles/Oxnard Homepage, [Online] (2003) Available: http://www.nwsla.noaa.gov/
[18] National Weather Service, Nighttime fog/daytime reflectivity image, [Online] (2003) Available: http://nimbo.wrh.noaa.gov/hanford/nowcast/satpix/fogrefl.htm
[19] South Coast Air Quality Management District, AQMD air quality area map, [Online] (2003) Available: http://www.aqmd.gov/smog/areamap.html
[20] Environmental Protection Agency, AIRNow ozone map archives, [Online] (2003) Available: http://www.epa.gov/airnow/ozone.html
[21] Mapquest, [Online] (2003) Available: http://www.mapquest.com
[22] National Geographic, Maps and geography, [Online] (2003) Available: http://www.nationalgeographic.com/maps/
[23] Cape Science, Global weather service, [Online] (2003) Available: http://www.capescience.com/webservices/globalweather/index.shtml
[24] California Transportation Agency, Caltrans realtime freeway speed map, [Online] (2003) Available: http://www.dot.ca.gov/traffic/
[25] A. Bui, D. Aberle, J. Goldin, M. McNitt-Gray, A.F. Cárdenas, E. Kleerup, O. Ratib, TimeLine: a multimedia, problem-centric visualization of patient records, Journal of American Medical Informatics Association Annual Proceedings, 1999.
[26] D. Aberle, A. Bui, E. Kleerup, J. Goldin, M. McNitt-Gray, M. Brown, A.F. Cárdenas, TimeLine: a multimedia problem-centered interface for healthcare, Computer Aided Radiology Symposium (CARS 99), Vol. 1165, Paris, France, 1999, pp. 562–566.