Persistent Data Management for Visual Applications*

Gökhan Kutlu, Bruce A. Draper, J. Eliot B. Moss, Edward M. Riseman, Allen R. Hanson

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

A *visual application* is an application that manipulates visual data as part of its processing. Visual applications need to represent, manipulate, store, and retrieve both raw and processed visual data. Existing relational and object-oriented database systems fail to offer satisfactory visual data management support because they lack the kinds of representations, storage structures, indices, access methods, and query mechanisms needed for visual data. We argue that extensible visual object stores offer feasible and effective means to address the data management needs of visual applications. ISR4 is such a visual object store under development at the University of Massachusetts for the management of persistent visual information.

1 Introduction

A *visual application* is an application that manipulates visual data as part of its processing. With advances in image analysis, visualization, and video technologies, increasingly large amounts of digital visual data are being generated by visual applications in a tremendously diverse range of domains, such as geographic, astronomical, and environmental information management, engineering and scientific visualization, military intelligence, computer aided design and manufacturing (CAD/CAM), and medical imaging.

Visual data consumed in applications consist not only of raw sensory data such as images, but also processed data, such as the knowledge structures used in visual interpretation systems, and associated model data (such as CAD/CAM models). As the scale of visual applications grows, the need to efficiently process, store, and access the raw and processed data becomes more acute.

Typically, a large amount of the data generated in a visual application is needed for temporary use only. A line extraction algorithm, for example, may produce hundreds (or thousands) of line segments from an image. Although not permanent, such data may be accessed repeatedly by other modules (e.g., line grouping or model matching algorithms), so the efficiency of in-memory data representation and retrieval is critical.

On the other hand, *a priori* knowledge such as maps and models, make up a permanent data base of information that visual algorithms access repeatedly and alter only occasionally. Similarly, generated long-term data, such as the set of extracted line segments that make up a site model, have to be stored for future access. Although less voluminous per image than the temporary data mentioned above, this data is persistent and grows to larger total amounts over time. It therefore must be managed by efficient storage and access mechanisms which are geared to the nature (e.g., spatial, temporal, 3D) of the data.

We addressed the management of temporary data in an earlier visual data management and process integration tool, called ISR3 [Draper, 1994]. In this paper, we first discuss the issues related to the management of persistent data in visual applications, and the shortcomings of current relational and object-oriented systems in dealing with these issues. We then argue that extensible visual object stores offer a feasible and efficient means to address the data management needs of visual applications, and present ISR4, a visual object store under development at the University of Massachusetts.

*This work was supported in part by the Advanced Research Projects Agency (via U.S. Army TEC) under contract number DACA76-92-C-0041, (via TACOM) under contract number DAAE07-91-C-R035, and by the National Science Foundation under grant number CDA-8922572.

2 Persistent Data Management Issues

As described below, the efficient storage and retrieval of large volumes of permanent visual data, such as aerial images, site models, and MRI scans, imposes requirements that are vastly different from those found in conventional data processing. As a result, existing relational and object-oriented database systems fail to offer the kinds of storage structures, indexing and access methods, and query mechanisms needed for visual data.

2.1 Efficient Storage Structures

Large, Multi-dimensional Objects. One issue is how to manage the storage and retrieval of large, multi-dimensional objects such as images. Space- and time-efficient storage and access of large visual objects is critical in projects such as The National Digital Library program at the Library of Congress, which involves providing access to a major subset of approximately 105 million items, among which are large numbers of digitized pictures.

Most applications store images in files, and leave the management of memory (page swaps, etc.) to the operating system. This approach can result in a large number of page swaps, especially when the physical clustering of the image on disk does not match the access pattern of the application [Stonebraker, 1994]. Traditional database systems do not provide appropriate data types or built-in support for images or similar 2D objects (e.g. maps). Adaptive clustering techniques used for clustering multi-dimensional data according to patterns of access are not mature, and the ones suggested depend on complex access pattern statistics [Dröge, 1993, Stonebraker, 1994].

Associative clustering. As discussed above, many visual applications need to store not only raw images, but also symbolic data extracted from (or associated with) images. In content-based image retrieval, for example, commonly stored data include color histograms, invariants of shape moments, and texture features. Moreover, symbolic data often need to be associated with the image region they came from so that they can be retrieved with the sub-image. In the RADIUS [Mundy, 1992] program, for example, site models reconstructed from sets of aerial images need to be grouped, stored, and retrieved according to their functional areas.

Most database systems provide little control over clustering of information in external storage so that a sub-image and the related analysis results can be stored on the same disk page, and retrieved together efficiently.

2.2 Multi-dimensional and Temporal Indexing

Visual data that are spatial in nature, such as geometric image structures, often need to be accessed according to their spatial properties in an image and/or 3D world positions. Therefore, spatial indices need to be maintained for efficient access to such data. Also needed, especially in military intelligence and medical imaging applications, are temporal indices defined over a time-sequence of image data. A typical medical query example is to find the first sign of a tumor in a history of MRI data.

Unfortunately, there is a lack of effective support for multi-dimensional and temporal indexing techniques in existing database systems. Moreover, simply adding one or two popular indexing methods, such as n-dimensional R-trees, is only a limited solution awaiting situations where a completely different index is needed. Instead, the ability to incorporate one's own indexing mechanism into the data management system is clearly called for.

2.3 Query Mechanisms and Optimization

Spatial, temporal, and geometric representations. Current relational and object-oriented query languages do not express the necessary spatial, temporal, and geometric concepts and operators effectively. For example, one must usually build specific concrete representations of n-dimensional points, lines, curves, regions, etc., and most systems provide no appropriate treatment of geometric anomalies that arise from, for example, numerical roundoff errors. Although one can represent concepts such as points and lines, attempting to express notions such as "distance" and "collinearity" leads to very inefficient query processing in traditional database systems. Moreover, explicit coding of such data types sacrifices representational independence. Likewise, one typically does not have the most efficient algorithms available, e.g., from computational geometry.

Approximate, ranked retrieval. A deeper problem with existing query languages is that they are boolean. A fact or record either definitely lies in the query result set or it does not. Many queries in visual applications are more likely to be concerned with approximate matches and/or ranked retrieval, where the goal is to find the best answers to a query and to rank them according to their degree of quality. A simple example would be finding objects "near" another object: we might return a list of objects ranked according to their distance from the query object, up to some maximum distance and/or maximum number of objects.

Query Optimization. In addition to query languages being limited in concepts and operators, existing query optimizers are not prepared to take into account geometric algorithms, spatial/temporal indexing, and ranked retrieval. This may become a critical issue when scaling to large systems, since query optimization frequently has orders of magnitude impact on performance.

3 Extensible Visual Object Stores

A visual object store is an object store and its associated tools and facilities provided to support the representation, manipulation, storage and retrieval of visual data. An extensible visual object store will have a number of unique features, which help overcome the problems discussed in Section 2:

- It provides a powerful core of functionalities to answer the basic data management needs of an application, including efficient built-in data types, basic storage and retrieval ability, and efficient storage structures, access methods, and query mechanisms for complex visual objects.
- Applications will be provided the flexibility to add new features as needed, at *all levels* of the system.
- Multiple policies and implementations will be available for the database implementor to choose from.
- Buffer management and data clustering policies will be accessible for customization and fine tuning.
- Applications will be lighter-weight since the features of the visual object store will be well integrated, and only those features needed will be part of the application; unnecessary features will be turned off.

4 ISR4

ISR4 [Kutlu, 1996] is an extensible visual object store under development at the University of Massachusetts. As shown in Figure 1, ISR4 is the integration of an earlier visual data management and process integration tool called ISR3[1] [Draper, 1994], with Mneme [Moss, 1990], a persistent object store (also developed at the University of Massachusetts).

[1]ISR (Intermediate Symbolic Representation; [Brolio, 1989]) is the name of a series of symbolic databases for visual information developed at the University of Massachusetts; ISR4 is the most recent version.

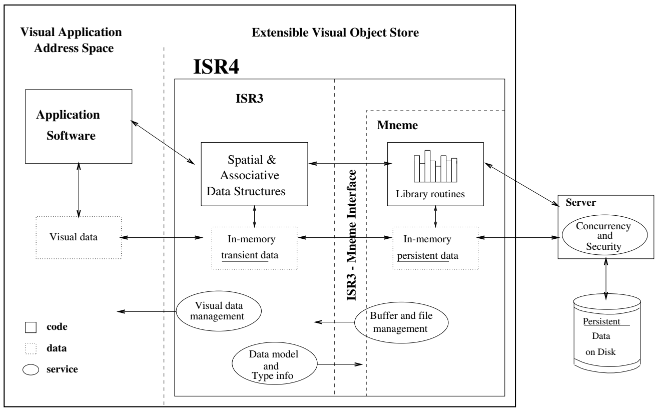

Figure 1: ISR4 System Architecture.

Spatial and geometric representations. Embedded in its host language C++, ISR4 allows arbitrarily complex objects to be defined and processed, and provides an initial set of standard visual representations, including single-band and multi-spectral images, 2D points, lines, edges, and regions, and 3D points, lines, surfaces, and volumes. Moreover, as the (object-oriented) data model is uniform throughout ISR4, complex representations such as spatially-indexed sets of line segments or histograms of image features, can transparently move between ISR3 and Mneme.

Customizable data clustering. ISR4 offers more than storage support; it provides methods for customizing Mneme's buffer management and clustering policies according to an application's needs. For example, the database implementor can use Mneme's basic capabilities to introduce data clustering policies that reduce data access delays for specific applications, such as storing an image region and its computed features in the same physical segment. Similarly, features which are multi-dimensional in nature, such as geometric image structures, can be clustered on disk according to user-specified access patterns for efficient access.

An example of customizing storage and access for visual applications is the ISR4 *tile-image* format, where an image is clustered on disk, and only required sub-images are brought into memory (on-demand). This ability significantly reduces the number of page swaps during common image processing operations [Kutlu, 1996].

Concurrent, Distributed Database Operations. Mneme supports concurrent database operations on arbitrarily complex objects within a distributed setting. It also provides customizable transaction and concurrency support, as well as extensible caching for use in client-server modes of operation.

Spatial and temporal indexing and query methods. ISR3 is equipped with a hierarchy of C++ classes that provide representations and methods for associatively and spatially organizing and accessing sets of memory-resident objects [Draper, 1994]. In particular, 2D geometric objects in images can be spatially stored into two-dimensional *grid*s [Brolio, 1989] and retrieved according to spatial position in the image.

We are currently developing persistent versions of these access methods. When manipulating persistent data, these techniques can significantly reduce data access times because only the index data structures need to be kept in-memory when indexing persistent objects. Visual data reside on disk and are brought into memory only when accessed, *on-demand*. The access data structures are stored on disk at program termination for later use.

Mneme already provides one such standard indexing mechanism: the B+ tree. Moreover, Mneme provides the database programmer with a flexible and powerful interface for building different types of indices, including spatial indices, such as quad-trees and R-trees, and other multi-dimensional indices.

We are also adding 3-D access mechanisms, and spatial and temporal query languages and techniques to this framework. A temporal index based on the Time Index [Elmasri, 1990], and optional versioning will also be provided to support historical queries. Once indices are built, query languages and techniques will also be implemented within this framework.

Extensibility. ISR4 offers generic solutions that lend themselves to immediate use by the visual database implementor, such as concurrent and distributed storage and retrieval ability for arbitrarily complex objects. However, a single generic solution is not suitable for more specific needs, such as application-dependent data structures, query methods, and indexing mechanisms. In such cases, ISR4 provides an initial set of powerful tools, and leaves it to the database implementor to generate representations, operators, indices, and query facilities tailored to the application. As an example, ISR4 allows—and encourages—the user to extend its initial set of representations by adding new ones [Kutlu, 1996]. Visual data types can be easily defined and integrated with the system using ISR4's data definition language (DDL). Likewise, Mneme is fully accessible for building multi-dimensional indices, or for tuning the buffer management and data clustering policies to the application-specific data requirements.

5 Motivating Examples

5.1 Content-based Image Retrieval

A number of current application areas exist that would immediately benefit from using ISR4. One is content-based image retrieval, for example, the QBIC [Niblack, 1993] project. In QBIC, color, texture, shape and sketch features are computed for image areas outlined by the user, and used at query time for image retrieval. The features, which consist of objects as complex as histograms and reduced resolution edge maps, are currently stored in an extensible relational database called Starburst [Lohman, 1991]. The images themselves, on the other hand, are stored in flat files.

One can achieve better data clustering and faster data access if the images and related features are stored using the strategies of ISR4. First, ISR4 will directly support the storage of QBIC objects, so there is no need for disk-to-memory data format transformations, as in the current transformation from tuples to objects. Second, image features can be associated with the image region they came from and stored and retrieved with the sub-image. This is useful in QBIC, especially when one wants to see which features (if any) were selected from an image region. Accessing the region will retrieve the corresponding features as well, which can then be displayed. Feature indexing capability is also critical in QBIC. The current B+ tree index can be used for fast object retrieval, and different types of multi-dimensional indices can be built and incorporated into ISR4. Along with indices, query mechanisms can also be implemented.

5.2 Site Models for Photo-interpretation

Intelligence gathering operations provide other applications. As an example, the RADIUS project [Mundy, 1992] is developing Image Understanding (IU) tools for image-analysts to support automated 3D cite model acquisition, model extension, and change detection. In a typical scenario, analysts build up a folder of image data and other intelligence about a site. Based on this information, analysts form a 2D map of the functional areas of the site, including abstract features such as the typical number of cars found in each parking lot. Finally, 3D models of the permanent structures in each area are built. Once a site model has been developed, future images and intelligence reports can be compared to it in a set of processes called "change detection," in which analysts search for any temporal change in the functional areas, features, and/or structures in a site.

Here, 3D geometric site models plus collateral information, such as text, maps, and representative imagery, need to be stored in a fashion that allows efficient data retrieval for change detection programs, as well as interactive query support for photo-analysts and military planners. Currently, the RADIUS Testbed Database (RTDB)[Hoogs, 1994] stores complex objects such as geometric models, collateral data, and imagery information in a relational DBMS (Sybase), while image pixel data are stored in flat files.

ISR4 would allow RADIUS features to be grouped, stored, and retrieved according to their functional areas. Images would be partitioned according to functional areas, and the sub-images would be clustered on disk with their associated features. In addition to fast access to image objects, this approach leads to better buffer management, especially with large aerial site images, since it restricts data movement to only a small, relevant portion of the image. Since RADIUS images are typically 10K$\times$10K pixels or larger, such efficient buffering mechanisms are required. As with QBIC, spatial indices, as well as query languages, can be built using ISR4 to answer interactive queries from analysts and planners such as 'Give me the image (folder) of this site in which this building appears for the first time.' To support historical (time-based) queries, the functional areas can be linked over time to form a spatio-temporal sequence, over which site structures are indexed.

In a similar manner, ISR4 can support other applications with visual representations, operators, and storage management, including astronomy (sky survey) databases, geographic and environmental information management, CAD tools, and medical imaging.

6 Conclusion

Visual applications need to efficiently represent, manipulate, store, and retrieve both raw and processed persistent visual data. Extensible visual object stores offer effective means to address the data management needs of visual applications. ISR4 is an extensible visual object object store that will offer extensive storage and retrieval support for complex and large visual data, customizable buffering and clustering, and spatial and temporal indexing. In doing so, it will provide a variety of multi-dimensional access methods and query languages. Query optimization, along with approximate, ranked query methods, are among planned future additions.

References

[Brolio, 1989] J. Brolio, B. Draper, R. Beveridge, and A. Hanson. ISR: A Database for Symbolic Processing in Computer Vision. *IEEE Computer*, 22(12):22–30, 1989.

[Draper, 1994] B. A. Draper, G. Kutlu, E. Riseman, and A. Hanson. ISR3: Communication and Data Storage for an Unmanned Ground Vehicle. In *IEEE International Conference on Pattern Recognition*, pages 833–836, 1994.

[Dröge, 1993] G. Dröge and H.-J. Schek. Query-Adaptive Data Space Partitioning Using Variable-Sized Storage Clusters. In *Advances in Spatial Databases: Proceedings of the 3rd International Symposium SSD*, pages 337–356, 1993.

[Elmasri, 1990] R. Elmasri, G. Wuu, and Y. Kim. The Time Index: An Access Structure for Temporal Data. In *Proceedings of the Conference on Very Large Databases*, Brisbane, Australia, August 1990.

[Hoogs, 1994] A. Hoogs and B. Kniffin. The RADIUS Testbed Database: Issues and Design. In *IUW, Monterey, CA*, volume 1, pages 269–276, Nov. 1994.

[Kutlu, 1996] G. Kutlu, B. A. Draper, J. E. B. Moss, and E. Riseman. Support Tools for Visual Information Management. To appear in *SDAIR*, 1996.

[Lohman, 1991] G. M. Lohman, B. Lindsay, H. Pirahesh, and K. B. Schiefer. Extensions to Starburst: Objects, types, functions, and rules. *Communications of the ACM*, 34(10):94–109, 1991.

[Moss, 1990] J. Eliot B. Moss. Design of the Mneme Persistent Object Store. *ACM Transactions on Information Systems*, 8(2):103–139, April 1990.

[Mundy, 1992] J. L. Mundy, R. Welty, L. Quam, T. Strat, W. Bremmer, M. Horwedel, D. Hackett, and A. Hoogs. The RADIUS Common Development Environment. In *IUW, San Diego, CA*, pages 215–228, Jan. 1992.

[Niblack, 1993] W. Niblack et. al. The QBIC Project: Querying Images By Content Using Color, Texture, and Shape. In *SPIE, Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–187, 1993.

[Stonebraker, 1994] S. Sarawagi and M. Stonebraker. Efficient Organization of Large Multidimensional Arrays. In *International Conference on Data Engineering*, volume 10, pages 328–336, 1994.