Issues in Using Knowledge to Perform Similarity Searching in Multimedia Databases without Redundant Data Objects

Leonard Brown
The University of Oklahoma
School of Computer Science
200 Felgar St. EL #114
Norman, OK, 73019
lbrown@cs.ou.edu
Tel.: (405) 946-7381
Fax: (405) 325-4044

Le Gruenwald
The University of Oklahoma
School of Computer Science
200 Felgar St. EL #114
Norman, OK, 73019
gruenwal@cs.ou.edu
Tel.: (405) 325-3498
Fax: (405) 325-4044

Greg Speegle
Baylor University
Department of Computer Science
Waco, TX, 76798
speegle@cs.baylor.edu
Tel.: (254) 710-4252

Abstract
This paper presents a possible way that artificial intelligence can be used to perform searching in a multimedia database management system without redundant data. Specifically, it finds the nearest neighbors of some query object without computing the distance between it and every other item in the database. Knowledge about the data in the system is required to perform searching. This knowledge is obtained from a specific storage format of the objects called a specification, which contains the editing operations used to create multimedia data. These operations are used to determine the similarity between multimedia objects without having to analyze their contents.

1. Introduction

A goal of any database management system is to be able to satisfy queries efficiently while minimizing the amount of redundant information stored in the database [KS91]. This goal is also desired in a MultiMedia DataBase Management System (MMDBMS). Given the nature of certain multimedia applications, however, accomplishing this goal can be difficult.

To illustrate this difficulty, consider an example application for cartoonists. This application allows the users to create, store, and retrieve individual panels of their cartoons. Since cartoons usually tell a story, a series of panels will often be very similar to each other. So, to create new panels quickly, the cartoonist can simply edit existing ones. For example, the second panel in a comic strip may be a close-up of one of the characters in the first panel. To simulate this close-up, a cartoonist can crop the desired character from the first panel, then enlarge it.

The key characteristic of this application is that both the first and second panels need to be stored in it. As the cartoonist creates several new images from editing existing ones, the underlying database is forced to store many similar images. This is a problem because such multimedia objects are generally quite large. For example, high resolution images can consume several megabytes of space [KA97]. So, storing thousands of these panels will quickly consume a database's storage area. In addition to our example cartoonist database, this characteristic may be present in any application in which users create new multimedia objects. This is because the users may save several different versions of their creations.

Saving several different versions of multimedia data items implies that an MMDBMS is often required to store many similar large objects. These similar objects represent redundant information. So, storing these objects in their traditional format means that the MMDBMS would contain a large amount of redundant information. Therefore, the MMDBMS would not achieve the goal of minimizing redundancy. Note that this is also true even if logical representations of multimedia objects are used such as color histograms for images. If a new image is created by rotating an existing one 90°, both of their histograms would be identical, and therefore redundant.

A storage format for eliminating the redundancy in such objects was proposed in [Spe95, GS96]. The goal of this paper is to present a method for performing similarity search using this storage format. This method requires an intelligent analysis and interpretation of the information that is contained directly in the format. In the next section, we will describe this storage format as well as the type of information that can be extracted from it. In section 3, we will describe the issues involved in searching for nearest neighbors in MMDBMSs using this information and explain why it is necessary. In section 4, we will describe how existing similarity search algorithms may be modified using an intelligent analysis of this information. Finally, we will summarize our paper in section 5.

2. Storing Multimedia Objects as Specifications

As stated earlier, in applications where users create, edit, and store multimedia objects, there will be a large amount of redundant data stored. In order to reduce the amount of space used in these types of applications, the redundant information can be eliminated by storing any new object created by editing another object in the database as a *specification* [Spe95, GS96].

This means that the new object is stored as a reference to the previously existing object along with the set of editing operations used to create it. So, for a user to access the new object stored in this manner, it must be recreated according to the information stored in its specification. Specifically, the existing object is referenced, then the associated editing operations are performed on it. This process is called *instantiation*. We also refer to the existing object as the *base*, and the new object as *derived*.

As an example of this concept, consider Figure 1 in which two images need to be stored in an example cartoon panel database. The base image is of two characters standing side by side. The derived image contains a close-up of one of the characters and was created by first cropping the character's head, then enlarging it. The base image would be stored normally, meaning that it would be stored in a conventional format for images. The derived image would be stored as a reference to the first panel along with the necessary instructions or operations for transforming it, which are crop and enlarge in this example. This allows us to store an entire multimedia object as only a few bytes of text.

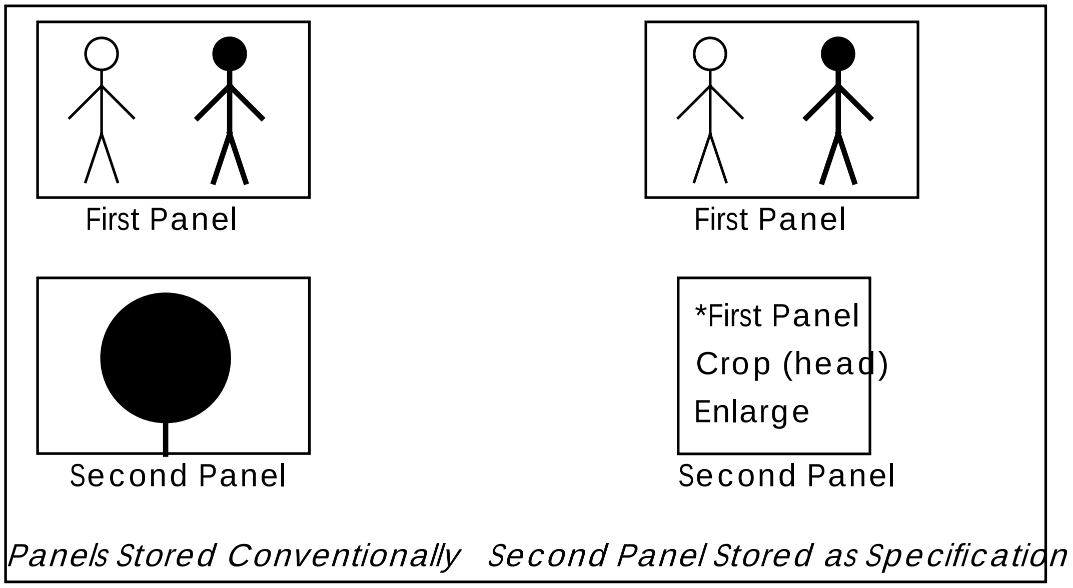

Figure 1 -- Storage of Similar Images with and without Specifications

An MMDBMS has many benefits from using specifications [GS96]. Since storing a derived object as a specification only requires a few bytes of text, specifications use much less space than conventional multimedia storage formats including compression algorithms such as JPEG [Wal91]. Specifications can also be used independent of the storage format of the base object which allows more flexibility in their use. In addition, the act of repeatedly instantiating and storing a specification is lossless, so there will not be any degradation in the quality of the stored multimedia objects. Finally, when the set of possible editing operations used in specifications is standardized [BGS97, BG98], any editor will be able to interpret and recreate them. This makes them portable.

There are disadvantages that arise from using specifications, however. The main drawback is that retrieving a specification involves instantiating it first, so it will take more time than simply retrieving a derived image stored conventionally. Another drawback is that an MMDBMS that uses specifications is dramatically different from one that does not. The existing techniques for performing common database functions for a conventional MMDBMS may not be appropriate for one that uses specifications. For example, in an MMDBMS that uses specifications, deleting a multimedia object is more complex because other objects may be derived from it. Thus, simply deleting an object may cause other objects to contain invalid references. In the next section, we will discuss the reasons why performing similarity searches are also more complicated when using specifications.

3. Requirements for Performing Similarity Search Using Specifications

As stated earlier, a common query in an MMDBMS is to retrieve the objects from a database that are the most similar to or nearest to some target object. In this section, we discuss the general issues that must be resolved to satisfy such queries by an MMDBMS that uses specifications. To discuss these issues, we must first review the issues present in performing a similarity search in any MMDBMS.

3.1 Similarity Search Requirements

Finding a multimedia object that is near another object implies that a distance function, D, exists that can measure the similarity. This function accepts two multimedia objects as input and returns some value. This distance function must be defined so that its returned value is nonnegative, it is symmetric, and it obeys the triangle inequality [SW90].

The most intuitive method to find the nearest neighbors of a query object, Q, is to compute the distance from it to each of the objects in the database. Once all the distances are known, they can be sorted in ascending order. The smallest distance corresponds to the nearest neighbor. The problem with this intuitive method for performing a similarity search is that it is often expensive computing the distance between two multimedia objects. The distance function described earlier computes the similarity between representations of two multimedia objects. The process of computing the similarity may be extremely complex or costly [SK98].

As a result of this, many researchers have proposed methods of finding the k-nearest neighbors of a query object that reduce the number of times the distance function, D, must be computed in the database [SW90, BK73, SK98]. These algorithms tend to determine a way of cheaply approximating the distance between two multimedia objects. These approximations are used to find and eliminate objects in the database that cannot possibly be the nearest neighbor to the query based on some known actual distance computations. For example, if the actual distance, D1, from the query to one of the objects in the database is known, the MMDBMS does not have to compute the actual distance from Q to objects whose approximate distance is greater than D1. This assumes that the approximate distance between two objects is always less than the actual distance [SW90, SK98].

More formally, assume that some function $\delta(M1, M2)$ is always less than $D(M1, M2)$. Now, compute one actual distance from the query object to one of the database objects, M. Thus, $D(Q, M)$ is known. If we compute $\delta(Q, O)$ for all objects O in the database, we can compare each value to $D(Q, M)$. If $D(Q, M) < \delta(Q, X)$ for some object, X, in the database, then $D(Q, M) < D(Q, X)$. Since the distance from Q to M is smaller than the distance from Q to X, Q is closer to M than X. Therefore, X cannot the nearest neighbor of Q. Note that we have derived this information without having to compute $D(Q, X)$. We can repeatedly select a new object and compute its actual distance which will consequently eliminate having to compute other objects in the database. This allows us to find the nearest neighbors of Q while reducing the number of times we have to compute the actual distance from Q to objects in the database [SW90, SK98].

3.2 Similarity Search with Specifications Requirements

As stated in the previous section, the goal of many similarity search algorithms is to reduce the number of times costly distance functions are computed. Many assume that the distance function is the largest cost for an MMDBMS [SW90]. This assumption changes when specifications are considered. If specifications are used, then the process of instantiation becomes the most expensive cost. So, we must avoid instantiating a specification to find its distance to the query object, Q.

Such distances can be found using knowledge derived from the information stored in specifications. For example, we can compute the distance from a derived object to its base as a function of the editing operations used to create the derived object. This distance can be computed by determining how much the associated editing operations change a multimedia object. This means that the distance between a specification and its base should be computed using a *transformation-based* technique as opposed to a *metric-based* one [Sub98]. Note that such a computation may not be expensive. Thus, we do not have to avoid computing the distance between a derived object and its base during a similarity search.

Determining how similar a derived object is to its base cannot be accomplished by simply attaching a weight to each editing operation. This is because the combination of two operations does not necessarily alter a multimedia object more than one of the operations by itself. To illustrate, imagine that the specification of a derived image contains two operations. Both operations are rotate 180°. After applying both operations, the resulting derived image is identical to its base. According to the nonnegative definiteness property [SW90], the distance between the specification and its base should be zero. If we attach a nonzero weight of w to each rotation operation, then the specification would have a total weight of 2w, and would therefore violate the nonnegative definite property of a distance function.

The MMDBMS, then, must be able to understand and interpret any combination of editing operations contained in a specification. In addition, the notion of similarity changes based on the underlying application. So, to produce a value that measures the similarity between a derived object and its base, the MMDBMS must consider the features contained in the base object and determine how they are affected by the combination of editing operations listed in the derived object's specification. Since there is an infinite number of different combinations of editing operations, the MMDBMS must contain an intelligent component that can either learn or deduce the similarity measurement.

4. Modifying Existing Similarity Search Algorithms for Specifications

If we use a transformation-based metric for computing the distance between a derived object and its base, we change the assumptions of existing similarity search techniques. Therefore, we must modify the conventional methods for finding the nearest neighbors of a query object, Q. Generally, we must determine a new inexpensive estimation, δ, of the actual distance function, D. Again, this function must be chosen such that $\delta(X, Y) < D(X, Y)$ for all multimedia objects X and Y.

Let B be a base object in our MMDBMS, and let S be an object derived from it. By modifying the similarity search technique proposed by Burkhard and Keller in [BK73, SW90], we will present a function $\delta(S, Q)$ that is an inexpensive lower bound for $D(S, Q)$. First, we must briefly review Burkhard and Keller's similarity search technique.

Burkhard and Keller's technique is to use the triangle inequality to derive their inexpensive distance function, δ [BK73, SW90]. They select one of the objects in the database, R, as a reference point, and compute the actual distance from R to all of the objects in the database. Once completed, they use these distance computations to find the k-nearest neighbors of any query object presented to the database.

Burkhard and Keller's distance function is derived from the fact that $D(Q, R) \leq D(Q, X) + D(X, R)$ for any object, X, in the database. When $D(X, R)$ is subtracted from both sides, $D(Q, X) \geq D(Q, R) - D(X, R)$ is obtained. Note that by starting with $D(X, R) \leq D(Q, X) + D(Q, R)$, the inequality $D(Q, X) \geq D(X, R) - D(Q, R)$ could just as easily be obtained. Thus, $D(Q, X) \geq |D(X, R) - D(Q, R)|$ which serves as the approximation function, $\delta(Q, X)$.

Given the intelligent component described in the previous section, we could inexpensively determine a value for $D(S, B)$ since S is derived from B. We could not, however, determine the distance from S to some other object X without first knowing $D(B, X)$. This includes the case when X is the reference object, R. So, to minimize distance computations, we must assume that we do not know $D(S, R)$ which means that Burkhard and Keller's approach must be modified to use it in an MMDBMS that uses specifications. Specifically, we have to find a new approximation for $D(S, Q)$ that does not use $D(S, R)$. We will call this approximation, $\delta'(S, Q)$.

We will use a similar approach to Burkhard and Keller's to derive this approximation. We know that $D(S, Q) + D(Q, R) \geq D(S, R)$. We also know that $D(S, R) \geq |D(B, R) - D(B, S)|$ from reasoning similar to the logic presented earlier. Thus, $D(S, Q) + D(Q, R) \geq |D(B, R) - D(B, S)|$. This leads to the inequality $D(S, Q) \geq |D(B, R) - D(B, S)| - D(Q, R)$. Since the distance of each base object to the reference point is known, we can use this as our inexpensive approximation function. So, $\delta'(S, Q) = |D(B, R) - D(B, S)| - D(Q, R)$. When it is known that this function is greater than the distance from Q to some other object, X, in the database, we know that S is not the nearest neighbor of Q.

5. Summary and Future Work

Specifications [Spe95, GS96] can be used to eliminate redundancy in MMDBMSs for systems that store many similar multimedia data items such as an application that lets a user design and create new objects. Since specifications store the editing operations used to derive new multimedia objects from existing ones, they have information that is not present in the conventional binary storage formats. This information can be used to measure how similar the new object is to the existing one. Because there is an infinite number of combinations of editing operations that can be stored in specifications, such a measurement must be either deduced or learned.

Given an intelligent component that performs this deduction, we can compute inexpensive approximations for the distance between a specification and some query object. These approximations allow us to modify existing techniques for finding the nearest neighbor of a query object similarly to the manner presented in section 4.

For future work, we must develop the component or module that determines the similarity between a derived object and its base using a specification. To develop this module, we must first define the set of editing operations that may be used in a specification. In addition, we will have provide training for the module so that it can determine the similarity between a derived object and its base.

References

[BGS97] Brown, Leonard, Le Gruenwald, and Greg Speegle, "*Testing a Set of Image Processing Operations for Completeness*", Proceedings of the 2nd Conference on Multimedia Information Systems, April 1997, pp. 127-134.

[BG98] Brown, Leonard and Le Gruenwald, "*Determining a Minimal and Independent Set of Image Processing Operations for a Multimedia Database System*", Proceedings of the 1998 Energy Technology Conference and Exhibition, February 1998.

[BK73] Burkhard, W. A. and R. M. Keller, "*Some Approaches to Best-Match File Searching*", Communications of the ACM, Volume 16, Number 4, April 1973, pp. 230-236. Referenced in [SW90]

[GS96] Gruenwald, Le and Greg Speegle, "*Research Issues in View-Based Multimedia Database Systems*", Proceedings of the 2nd World Conference on Integrated Design and Process Technology, December 1996, pp. 331-336.

[KA97] Klas, Wolfgang and Karl Aberer, "*Multimedia and its Impact on Database System Architectures*", Multimedia Databases in Perspective, Chapter 3, P. M. G Apers, H. M. Blanken, and M. A. W. Houtsma (Eds.), Springer, 1997.

[KS91] Korth, Henry F. and Abraham Silberschatz, Database System Concepts, McGraw-Hill, Inc., New York, 1991.

[SK98] Seidl, Thomas and Hans-Peter Kriegel, "*Optimal Multi-Step k-Nearest Neighbor Search*", ACM SIGMOD, 1998, pp. 154-165.

[Spe95] Speegle, Greg, "*Views of Media Objects in Multimedia Databases*", Proceedings of the International Workshop on Multimedia Database Management Systems, August 1995, pp. 20-29.

[Sub98] Subrahmanian, V. S., Principles of Multimedia Database Systems, Morgan Kaufmann, San Francisco, California, 1998.

[SW90] Shasha, Dennis and Tsong-Li Wang, "*New Techniques for Best-Match Retrieval*", ACM Transactions on Information Systems, Volume 8, Number 2, April 1990, pp. 140-158.

[Wal91] Wallace, Gregory K., "*The JPEG Still Picture Compression Standard*", Communications of the ACM, Volume 34, Number 4, April 1991, pp. 30-44.