Issues in Using Specifications to Improve Content-Based Search of Multimedia Data

Leonard Brown
The University of Oklahoma
School of Computer Science
Norman, OK, 73019
lbrown@cs.ou.edu

Le Gruenwald
The University of Oklahoma
School of Computer Science
Norman, OK, 73019
gruenwal@cs.ou.edu

Greg Speegle
Baylor University
Dept. of Computer Science
Waco, TX, 76798
speegle@cs.baylor.edu

Abstract

Many current multimedia database management systems perform content-based retrieval of images by extracting the values of various features from every object stored in their system. This can be time-consuming, especially if extracting the features requires a human to analyze each object. This process can be minimized in multimedia database systems that store images as a sequence of editing operations, called specifications, instead of the usual binary format. This paper discusses the advantages and applicability of such systems and the issues that must be resolved in order to develop them.

1. Introduction

DataBase Management Systems (DBMSs) have become the cornerstone of the business community. The ability to efficiently store and retrieve huge quantities of information on products, consumers, suppliers, employees and other facets of an enterprise have revolutionized commerce in developed nations. The next logical step in the management of information is to extend databases from text based information to multimedia data, such as images. Extending these databases, however, is a difficult task. The reason is that one of the main functions of any DBMS is to store and retrieve data [43, 33, 47], and the conventional ones are not appropriate for storing and retrieving images [4].

One of the main reasons that conventional DBMSs are inappropriate for performing these tasks is that images need to be interpreted. It is not desirable to search for images based on some textual description that may be associated with it such as a filename or a set of keywords because such descriptions are subjective [23]. So, ideally, an image should be retrieved based on its content, which should be extracted automatically. This is not possible in conventional DBMSs. For example, users should be able to query the database requesting all images that contain a picture of a dark blue sports car. A conventional DBMS would require that the keywords "dark", "blue", "sports", and "car" be attached to any image returned as a result of this query. Not only does this require that humans must inspect each image and attach these keywords, it does not permit humans to use other phrases such as "navy automobile" or make judgment errors such as "black car".

One of the requirements, then, of a MultiMedia DataBase Management System (MMDBMS) is that it must provide users with the ability to retrieve images and other multimedia objects based on their content. Consequently, many researchers have developed techniques that retrieve images from the database based on queries using features extracted from them. Systems using these techniques are called *Content-Based Image Retrieval (CBIR)* systems [11]. Some examples of these include QBIC [20, 15, 14], ARTISAN [11], IRIS [21], FIBSSR [30], and ImageRoadMap [32].

CBIR systems typically use the same approach to retrieve images based on their content. As each image is entered into the database, the values of the features that can be used for querying are automatically identified. Each image, then, can be represented by the set of values of the features extracted from it, called a *feature vector*. The result of this is that each image functions as a point in a multidimensional feature space [13]. For example, consider a CBIR system intended to allow searching based on the color of images. In such a system, a histogram can be created for each image where each bin stores the number of pixels in that image that contains a particular color. When normalized, each bin represents the percentage of pixels in the image that contains a particular color. So, as long as each image is represented by such a histogram, the users can query the database requesting the images that have a specified percentage of pixels containing a certain color. An example of such a query is "*Retrieve all images that are 25% blue and 25% green*." Similar histogram methods are used by numerous CBIR systems including the aforementioned QBIC [20, 15, 14], ImageRover [39], RECI [10], and IRIS [21].

Color is not the only feature used frequently in CBIR systems. Many of them allow users to query based on texture [10, 21, 25] and the shape of the primary object [10, 11, 32]. Again, however, these features are extracted from each image as it is entered into the database and stored in feature vectors. A common type of query required by users in an MMDBMS is to present a query object to the database and request the *k* data items that are the most similar to it. When this occurs, a feature vector is generated from the query object as well. This vector is compared to the feature vectors representing the images in the database in order to determine which of them are the most similar to the query. This is generally referred to as the *k-nearest neighbor* problem [13, 5].

A disadvantage of automatic feature extraction, however, is that it is generally limited to the global or primitive features of color, texture, and shape [11]. Extracting more complex features requires some degree of human interaction such as labeling automatically identified heterogeneous regions in an image [2]. Using humans is not only slow, but it also introduces inconsistencies in the process, since even the same person may view the same image differently at separate times.

Once the images and feature vectors are in the database, locating the k-nearest neighbors of a goal or query object has a worst-case requirement of computing the distance between it and every other object in the database [12]. Although multidimensional indexes such as the R-tree [19] and its variants [13, 6] have been designed to reduce the number of distance computations, two problems still remain [40, 14]. The first is that these indexes become ineffective when considering the large number of dimensions typically required by the feature vectors of multimedia objects [5, 14]. The second problem is that the time needed to compare two vectors is quadratic with respect to the number of dimensions [40].

The goal of this paper is to present issues regarding a more efficient method of performing content-based retrieval for images. Our method is more efficient because it avoids extracting features from all images and avoids executing computationally expensive comparisons of the query object to every stored image for each query. Our approach is to perform content-based retrieval using images stored as *specifications* [44, 18, 46], which will be discussed in the next section. After reviewing the concept of specifications, we will review various related approaches proposed by other researchers and why our research differs in section 3. In section 4, we will list the issues that need to be resolved to use specifications to perform content-based retrieval. Finally, in section 5, we will summarize the main points of this paper.

2. Specifications

One of the characteristics of images that makes their storage difficult is that they tend to require more space than conventional data. For example, very high-resolution images may require several megabytes [26]. Even with the falling cost of memory, attempting to store thousands of images can quickly exhaust storage space.

A method for reducing the space used by storing such images becomes evident when some example multimedia applications are considered. For example, consider a video game that contains several similar images. Each picture may display a character wearing a different uniform, where each uniform varies only by color and team logo. Another application is one that allows a cartoonist to create and store several comic strips. Each successive panel of a strip will be similar to the previous one, so the cartoonist will create new panels by editing existing ones. For example, the cartoonist may crop a character's head and enlarge it to illustrate a close-up.

This characteristic of storing several similar images will be true of any application where a user creates new images from other ones previously stored in the system. As each new image is created, the user will want to save both the old and new versions of it. So, these images will contain a lot of redundant data. As in traditional databases, such redundant data should be eliminated.

One method for eliminating the redundancy in these types of applications is by changing how the images are stored [45, 18, 46]. The idea is that instead of storing two similar images in their binary formats, only the original, called the *base* image, is stored in that manner. The new, or *derived*, image is stored as a reference to the first (base) image along with a set of instructions for transforming it into the new one. This representation of the derived image is called a *specification* of that image. Displaying a derived image stored in this manner can be accomplished by accessing the referenced base image and performing the associated instructions for transforming it.

As an example of this concept, consider Figure 1 in which two images need to be stored in the example video game application. The base image is of a player wearing a blue uniform with a logo of a tiger. The derived image contains a player on a different team created by changing the color of the uniform from blue to red and replacing the tiger with an eagle. The base image would be stored conventionally. The derived image would be stored as a specification, meaning that it will contain a reference to the base image along with the operations used to transform it. For this example, we will call the transformation operations "change color to red", "remove tiger logo", and "add eagle logo" although the operations would be lower level in practice. So, these operations are the instructions for transforming the base image into the derived image. Therefore, whenever a user wants to see the second player displayed, the base image would be accessed, and then the operations would be performed sequentially. This process is called *instantiation* [18].

In addition to using less space, using specifications to store images offers other advantages. Unlike many compression methods, storing and instantiating a specification is a lossless process. The derived image, then, can be retrieved and stored endlessly without any degradation. Another advantage of specifications is that they are not dependent upon any particular compression or storage format such as JPEG [50], nor are they dependent upon any specific computing platform. Thus, specifications are portable. This portability allows derived images to be stored as specifications instead of various compression formats. So, specifications can save space even when using data that is already compressed.

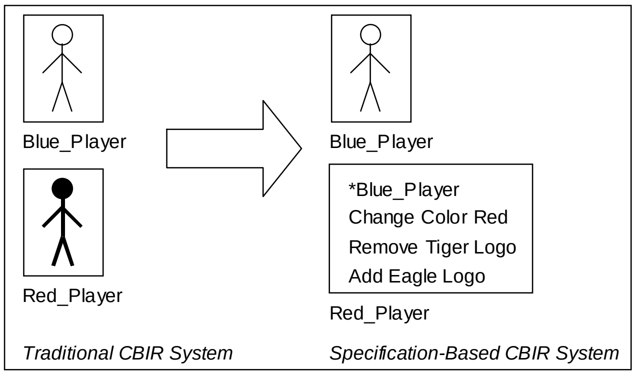

Figure 1 - Storage of similar images in traditional and specification-based CBIR systems

There are numerous application areas that store many similar images, and will therefore benefit from using specifications. In addition to the applications presented earlier, this approach will be useful in multimedia authoring environments where several versions of various images created by a user can be archived [17]. One example of such an environment is web design. Specifications will also be useful in the area of medicine where plastic surgeons can illustrate the changes they intend to make to their patients by editing their photo. Law enforcement is another area that can benefit by allowing users to save and retrieve several possible disguises of a suspect such as changing hair color, removing or adding glasses, or removing or adding facial hair. When other multimedia objects are also considered, such as audio or video, storing their objects as specifications can benefit such areas as broadcast media or the recording industry as indicated in [22].

3. Related work

There are two main areas of research related to our work. The first contains the numerous CBIR systems currently being developed. The second contains the research that is related to the concept of specifications and the standardization of the operations used to edit multimedia objects. We will review their research in this section and discuss how our work is different.

3.1. Content-based image retrieval systems

As stated before, there are numerous CBIR systems developed by researchers. Most of these systems do not use specifications or editing operations to improve the efficiency of their searching. There are, however, some aspects of the systems that will greatly affect our research.

First, there are several features commonly used in various CBIR systems for querying that are extracted from images automatically. These include color used in systems such as QBIC [20, 15, 14], ImageRover [39], RECI [10], and IRIS [21], texture used in [10, 21, 25], and shape used in [10, 11, 32]. We will be able to use this research to extract these features from our base images. However, alternative approaches must be employed to extract them from the specifications of derived images.

Another issue that occurs in CBIR systems is the representation used to store the features once they are extracted from the images. For example, many systems use feature vectors to represent the images contained in the system once the desired extractions are complete [34, 30, 39, 20]. However, using this representation in a specification-based CBIR system may result in storing redundant data. For example, consider a derived image that was created by rotating its base, and let both images be represented by feature vectors that are their respective normalized color histograms. Since the rotation operation does not add or remove pixels from an image, both vectors would be exactly the same. Thus, it would be redundant to store the percentages of pixels that contain each color in both histograms. Therefore, feature vectors should not be used for the representation of specifications.

In addition, CBIR systems must define some criteria for satisfying a query. This means that there must be some method of defining how similar one image is to another such as the Euclidean Distance [38, 30, 39] or other functions based on feature vectors [2]. Such vectors only contain information about the features that are extracted from the images, so they do not maintain any information about the relationship between a derived image and its base. Thus, that relationship does not affect the distance between the two images.

Another issue that must be resolved is developing a technique to access the data. Another common type of query described in [13] is the *range* query. As in conventional databases, an index is more appropriate than a hashing method for satisfying queries of this type since it preserves order. The indexing techniques commonly used, like the B-tree, use only one attribute called the key to search entire data records. In contrast, the DBMS should be able to search all of the attributes in a feature vector equally. For example, each dimension of a color histogram may be used equally in queries by users. Thus, traditional indexes for relational DBMSs are insufficient because they use one key to represent an entire vector.

The result is that images require an index that can be searched using any of the dimensions of the feature vectors representing them. Consequently, researchers have proposed many various multidimensional indexes. One of the more popular types of these indexes treats each feature vector as a point in a multidimensional space [13] and creates trees where each node corresponds to a section in that space. This category of indexes contains variations of the R-tree [19, 13, 6], which includes those data structures that group their data using Minimum Bounding Regions (MBRs) such as the X-tree [3] and others [24, 29] as well as those that do not use MBRs [5, 16, 28]. Those indexes that use MBRs partition only the portion of multidimensional space occupied by feature vectors stored in the database. Thus, they are able to reduce the time it takes to search the feature vectors because they eliminate the unused multidimensional space. They also, however, are burdened with having to compute an MBR for each node of their tree, unlike those indexes that do not use them [6].

3.2. Specifications / image editing operations

There are several proposed multimedia data models that use concepts similar to specifications such as scripts layers, or deltas [17, 37]. However, there is no capability to search or for data based on its content using the scripts or layers themselves.

There are many researchers that are compiling lists of editing operations for multimedia objects. For example, in [35], the researchers are producing an image algebra that will serve the same function as the relational algebra for conventional data. In addition, [27] lists and categorizes several image processing operations. In [49] and [31], researchers are developing languages for video objects. In [49], the Resolution Independent Video Language (RIVL) is introduced. This language is intended to allow users to create programs using images or video data in the same manner that programs are created today using numeric data. In [31], the researchers present a scripting language for producing video presentations. The main advantage both [49] and [31] cite is that using the high-level operations provides portability for multimedia data. However, neither group appears to be studying how content-based retrieval could be enhanced using their languages.

An algebra for manipulating video data is presented in [51]. This work is more closely related to our research than any of the others because it studies performing content-based access to video data using its algebra. In their work, searching is performed by using descriptions that are manually assigned to each video object using a description operator, while in our work, content-based retrieval can be performed without requiring the features of each multimedia object to be available. This is because we intend to determine whether or not a derived object satisfies a query directly from its specification.

4. Specification issues

To discuss the issues surrounding using specifications to perform content-based retrieval, it will be necessary to describe the advantages to our approach in more detail. Consider the base image of a player in a blue uniform described earlier. Also, consider a derived image of a different player wearing a red uniform that was created by changing all of the blue pixels to red. We are able to determine the percentage of each of the colors contained in the derived image directly from this specification. Its color histogram would be exactly the same as the histogram extracted from the base image, except that the percentage of red pixels is increased by the percentage of blue pixels, and the percentage of blue pixels is reduced to 0. Thus, for any query based on color that we can answer for the base image, we can answer it for the derived image. Note that we are able to answer such queries without having to extract the features from the derived image, nor even instantiate it. In addition, it was not necessary to compute the entire feature vector for the derived image. We only needed the percentage of pixels of the colors present in the user's query.

Now consider two other example images in the video game. The base image contains a player wearing a uniform with a tiger, and the derived image is created by enlarging the base by 50%. Now, consider the query, "*Retrieve all images containing a picture of a tiger*". Automatically determining whether or not an image contains a tiger is beyond the capabilities of current systems. So, this feature must be extracted manually or semi-automatically, meaning that a human must indicate to the system whether or not the image contains a tiger.

Having a human analyze every image contained in an MMDBMS takes a lot of time. By using specifications, we can reduce the number of images that have to be analyzed manually. We can determine whether or not the derived image in this example contains a tiger directly from its specification. So, we do not have to manually analyze the derived image as we would in a conventional MMDBMS. In addition, we again do not have to instantiate the derived image.

These examples illustrate that in some cases it is possible to perform content-based retrieval without extracting features from every object in an MMDBMS by using specifications. There are, however, several issues that must be resolved to implement this approach. The remainder of this section will discuss these issues.

4.1. Determine effects of transformations

In the examples presented in this section, we are able to answer the queries because we know how the operations in the specifications affect the features of the base image. Specifically, we know how the color histogram of an image is affected if all of the pixels of one color are changed into another. We also know that enlarging an image does not change the objects contained in it. So, one of the most important issues that need to be resolved to perform content-based retrieval using specifications is to determine the effects of the transformation operations contained in specifications on the features used for querying.

This requirement can be defined more formally. Call a base object X and an object derived from it Y. Let Y be stored as a specification. This means that Y is stored as a reference to X along with some list of operations used to transform it. Call this transformation T, and define T(X) as the result of performing the transformation on the image X. Therefore, Y = T(X). Now, let F represent a specific feature, and let F(n) represent its value extracted from some object n. We denote a feature vector representing an image, n, as $\vec{F}(n)$.

To perform content-based retrieval in an MMDBMS, we must know $\vec{F}(n)$ for all objects in the database. Therefore, we must know $\vec{F}(X)$ and $\vec{F}(Y)$ in our example. In a conventional MMDBMS, $\vec{F}(X)$ and $\vec{F}(Y)$ are determined through some feature extraction process performed on both X and Y. In an MMDBMS that uses specifications, the extraction process only needs to be performed on X. Since Y = T(X), $\vec{F}(Y) = \vec{F}(T(X))$. So, one method of finding $\vec{F}(Y)$ is to determine the composition of T and $\vec{F}$ [9].

There are several components of $\vec{F}(T(X))$ that we must identify to perform content-based retrieval. The first is that the possible values of X must be identified. This corresponds to identifying the set of images that will be stored in the application. This issue is the same for any DBMS. Database designers should know the properties of their data in order to select the best design.

Specifying the set of possible objects, {X}, will also lead to the next component, identifying all of the possible querying features, {F}. If the MMDBMS stores only faces, then the features may be eye color and hair color. If the MMDBMS stores only trademarks, then an important feature is the shape of the trademark. As stated earlier, since the users query the database using these defined features, completing this component determines the possible queries that can be entered by the user.

Intuitively, the next component is to specify all possible transformation operations, {T}. There are several issues to consider when performing this step. One is that the transformation operations are the set of editing operations a user can perform. So, if the transformations are not robust, the user will not be able to make elaborate changes to base images. Ideally, the set of operations {T} should be *complete*, meaning that it should be able to express all possible image transformations [8]. If too many operations are defined, however, it will be difficult to determine all possible compositions between the members of {T} and {F}.

Another issue that must be addressed is that the set of possible editing operations should be portable. Different editors should be able to display an image stored as a specification, so they must understand the operations contained within it. Therefore, the set of possible editing operations should be standardized [49, 31, 6].

The next component of $\vec{F}(T(X))$ to consider is T(X). T(X) should be expressed as succinctly as possible to reduce the computations needed to calculate $\vec{F}(T(X))$. This means that the operations given in T(X) should be optimized. For example, consider a transformation that first changes a color of a player's uniform from blue to red, then deletes the player from the image. This can be optimized by eliminating the first operation. This will not only speed up the computation of $\vec{F}(T(X))$, but also speed up the instantiation of T(X) when a user wants to access object Y. This optimization, then, should also be based on reducing the time it takes to instantiate a specification.

The final component of computing $\vec{F}(T(X))$ is to compute the composition of T and $\vec{F}$. This requires the development of rules regarding their composition. For example, one rule is that the rotation operation does not affect the color histogram of an image. So, if T is the rotation operation, and $\vec{F}$ is the histogram representing the percentage of different colors in an image, then we know that $\vec{F}(T(X)) = \vec{F}(X)$. This leads to the additional issues of identifying how to store the rules, determining which to apply, and accessing them efficiently.

An example of each of these components is presented in [1]. It uses the set of operations specified in [46]. The set of features, {F}, are *color* and *shape*. The value of each feature is a score from 0 to 100 that represents the confidence level that the image contains a specific instance of a feature such as the color red. The application of editing operations on images, T(X), is performed using defined regions. These are polygonal regions that are actually changed during the operations transforming X to T(X). These regions are then used to determine the final component, F(T(X)). Specifically, [1] uses the approach that each pixel in a defined region contributes equally to a feature. This means that after applying an operation on an image, the scores of its features are adjusted based on the number of pixels in the defined region that are altered.

4.2. Computing similarity

A common type of query that users frequently enter into MMDBMSs is called a *k-nearest-neighbor* query [13]. This refers to those queries in which the users request the database to return the k items that are the most similar to some user-defined image. The similarity of two objects is usually based on some function, D, measuring the distance between the feature vectors representing the images. Thus, D is a function mapping two feature vectors to a real number, $D(\vec{F}(X), \vec{F}(Y)) \rightarrow \boldsymbol{R}$. Formally, the k-nearest neighbor problem is the set of images Y such that no more than k-1 images in the database have a smaller value of D for the goal image X.

The set of k nearest images, K, can be calculated by computing the distance function on each feature vector in the database. The distance function used to compute the similarity between two multimedia objects often requires a large number of computations. For example, the time needed by many approaches to compute the similarity between two high-dimensional color histograms is often quadratic with respect to the number of dimensions [41]. So, one of the goals of an MMDBMS is to process nearest-neighbor queries while minimizing the number of times such expensive functions are evaluated.

In [41], an algorithm is proposed to accomplish that goal. Specifically, this algorithm processes nearest-neighbor queries while minimizing the number of times the actual, but expensive, distance function (d_o) is evaluated. Since nearest neighbor searching techniques often use an inexpensive function to approximate the distance between two objects [36, 42], this algorithm assumes the existence of such a function, d_f, such that $d_f(O_1,O_2) \leq d_o(O_1,O_2)$ for all multimedia objects O_1 and O_2. The algorithm begins by generating a candidate set of multimedia objects that may be the k-nearest neighbors of some query object, q. The set is determined initially by computing the less expensive function, $d_f(q,O)$ for all of the objects, O, in the MMDBMS. The algorithm then evaluates $d_o(q,O)$ for only those objects, O, where $d_f(q,O)$ is less than some value. This value is the actual distance of the k^{th} most distant object from the query object in the candidate set, which is adjusted as new candidates are added. The algorithm is argued to be optimal since $d_o(q,O)$ is evaluated for only those objects in the candidate set such that $d_f(q,O)$ is less than the actual distance of the k^{th} nearest object to q. Thus, no more evaluations of d_o will be performed than necessary [41].

In an MMDBMS that uses specifications, we will not compute the distance function, d_f, for derived objects since we are not storing their feature vectors. Instead, a more natural technique to measure the distance between the query and an object stored as a specification is the transformation approach [48]. In this approach, the distance between two objects is determined by the operations needed to transform one into the other. Each transformation operation, t_i, has an associated cost, $w(t_i)$, and the distance between the objects is the sum of the cost of the transformation operations, $\Sigma(w(t_i))$. Thus, the distance, *d*, between a derived object, T(X), and its base, X, is $\Sigma(w(t_i))$ where $T=<t_1, t_2, ..., t_m>$.

Once the actual distance between X and q is computed, we can use $\Sigma(w(t_i))$ to obtain an inexpensive lower bound on $d_o(q,T(X))$. Using the triangle inequality in a manner similar to [42], $d_o(q,T(X)) \geq |d_o(q,X) - \Sigma(w(t_i))|$. So, we can inexpensively compute a lower bound for the distance between a specification and the query object. This lower bound can then be used to avoid having to perform the more expensive distance computation, $d_o(q,T(X))$.

We must use caution in computing the distance between a derived object and its base, however. To illustrate, consider assigning a weight of $w > 0$ to the operation "*Rotate image 180°*". Applying the operation twice would produce a non-zero weight of 2w representing the distance of the resulting derived image to its base. The derived image, however, would be identical to its base, so it should have a distance of 0. The sum of the weights only produces an upper bound on the distance from a base to its derived objects. To be more accurate, the context of each operation must be considered.

4.3. Access method

The multidimensional indexes referenced in section 3 have been proposed for storing multimedia data represented by feature vectors. None of these indexes are appropriate in an MMDBMS that uses specifications. Again, this is because it is redundant to create and store feature vectors for images stored as specifications. Thus, alternative indexing strategies must be employed.

5. Summary and future work

There are many applications that allow users to derive new images by editing other existing ones. Examples of such applications include video games and comic strip databases. In applications such as these, space can be saved by storing the derived images as specifications. This storage format consists of the sequence of operations used to create the derived image along with a reference to the other existing image upon which it is based.

In addition to saving space, an advantage of an MMDBMS using this storage format is that it can perform content-based searching of images without extracting features from each object in the database. This is possible because the values of the features used in users' queries can be determined directly from the specifications. The result is that the process used for extracting features can be avoided for many of the images in the database. This will save time, especially when the feature extraction process requires a human to analyze the image. In addition, the MMDBMS does not have to maintain feature vectors for each image stored in the database.

In order to implement content-based retrieval using specifications, many issues must be considered. First, once the set of images in the database is defined, their important features should be identified as well as the list of editing operations that can be performed on them by the users. Once these sets are defined, the result of applying each operation on the various image features must be computed. This information is necessary to determine whether a derived image satisfies a query directly from its specification. In addition, this information represents the guidelines or rules used to perform content-based retrieval. These rules need to be stored in the MMDBMS, and techniques regarding their access must be determined.

Another issue that must be considered to perform content-based retrieval using specifications is the development of techniques regarding the measurement of similarity between two images when none, one, or both are stored as specifications. As presented in this paper, this may involve assigning a weight to each of the editing operations that can be applied on the images.

This paper also presented the issue of developing a technique for accessing derived images stored as specifications. Many conventional MMDBMSs compute and store feature vectors for each object in the database and use a multidimensional indexing technique for their access. This is insufficient for the approach presented in this paper since feature vectors may not be generated for derived images stored as specifications because they may contain redundant information. Therefore, alternative access methods must be developed.

Although our research has focused primarily on using specifications to store images, this method can be employed with any type of media data. Specifications can be used to enhance content-based retrieval in applications that store similar audio files or videos as well. In order to implement such applications, many of the issues discussed in this paper must be addressed for the audio and video data types as well.

6. References

[1] Aars, Michael, "*Automatic Feature Extraction Using Specifications of Images*", M.S. Thesis, Baylor University, 1999.

[2] Analyti, Anastasia and Starvos Christodoulakis, "*Content-Based Querying*", Multimedia Databases in Perspective, Apers, Blanken, and Houtsma (Eds.), Springer, 1997.

[3] Berchtold, Stefan, Daniel A. Keim, and Hans-Peter Kriegel, "*The X-Tree: An Index Structure for High-Dimensional Data*", Proc. of the 22nd Intl. Conf. on VLDB, 1996, pp. 28 - 39.

[4] Blanken, Hans, "*Introduction*", Multimedia Databases in Perspective, Apers, Blanken, and Houtsma (Eds.), Springer, 1997.

[5] Bozkaya, Tolga and Meral Ozsoyoglu, "*Distance-Based Indexing for High-Dimensional Metric Spaces*", Proc. of the 1997 ACM SIGMOD Intl. Conf. on Management of Data, May 1997, pp. 357-368.

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

[7] Brown, Leonard and Le Gruenwald, "*Tree-Based Indexes for Image Data*", Journal of Visual Communication and Image Representation, Volume 9, Number 4, 1998, pp. 300-313.

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

[9] Date, C. J., An Introduction to Database Systems, 6th Edition, Addison-Wesley, Reading, MA, 1995.

[10] Djeraba, Charbane et al., "*Retrieval and Extraction by Content of Images in an Object Oriented Database*", Proc. of the 2nd Conference on Multimedia Information Systems, April 1997, pp. 50-57.

[11] Eakins, John P., Jago Boardman, and Margaret E. Graham, "*Similarity Retrieval of Trademark Images*", IEEE Multimedia, Volume 5, Number 2, April-June 1998, pp. 53-63.

[12] Fagin, Ronald, "*Fuzzy queries in Multimedia Database Systems*", Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, June 1998, pp. 1-10.

[13] Faloutsus, Christos, Searching Multimedia Databases by Content, Kluwer Academic Publishers, Boston, 1996.

[14] Faloutsus, Christos, et al., "*Efficient and Effective Querying by Image Content*", Journal of Intelligent Information Systems, Volume 3, 1994, pp. 231-262.

[15] Flickner, Myron et al., "*Query by Image and Video Content: The QBIC System*", IEEE Computer, Volume 28, Number 9, September 1995, pp. 23-31.

[16] Freeston, Michael, "*A General Solution of the n-dimensional B-tree Problem*", Proc. of the 1995 ACM SIGMOD Intl. Conf. on Management of Data, 1995, pp. 80-91.

[17] Gibbs, Simon, Christian Breiteneder, and Dennis Tsichritzis, "*Data Modeling of Time-Based Media*", Proc. of the 1994 ACM SIGMOD Intl. Conf. on Management of Data, May 1994, pp. 91-102.

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

[19] Guttman, Antonin, "*R-trees: A Dynamic Index Structure for Spatial Searching*", Proc. of the 1984 ACM SIGMOD Intl. Conf. on Management of Data, 1984, pp. 47-57.

[20] Hafner, James et al., "*Efficient Color Histogram Indexing*

for Quadratic Form Distance Functions", IEEE Trans. on Pattern Analysis and Machine Intelligence, Volume 17, Number 7, July 1995, pp. 729-736.

[21] Hermes, Th. et al., "*Content-Based Image Retrieval*", Proc. of CASCON '95, November 1995.

[22] International Organization for Standardization, "*MPEG-7: Context and Objectives*", ISO/IEC JTC1/SC29/WG11, available http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm.

[23] Jagadish, H. V., "*Content-Based Indexing and Retrieval*", The Handbook of Multimedia Information Management, Grosky, Jain, and Mehrotra (Eds.), Prentice Hall, 1997.

[24] Katayama, Norio and Shin'ichi Satoh, "*The SR-Tree: An Index Structure for High-Dimensional Nearest Neighbor Queries*", Proc. of the 1997 ACM SIGMOD Intl. Conf. on Management of Data, May 1997, pp. 369-380.

[25] Kelly, Patrick M., Michael Cannon, and Donald R. Hush, "*Query by Image Example: The CANDID Approach*", SPIE Vol. 2420 Storage and Retrieval for Image and Video Database III, 1995, pp. 238-248.

[26] Klas, Wolfgang and Karl Aberer, "*Multimedia and its Impact on Database System Architectures*", Multimedia Databases in Perspective, Apers, Blanken, and Houtsma (Eds.), Springer, 1997.

[27] Klette, Reinhard and Piero Zamperoni, Handbook of Image Processing Operations, John Wiley and Sons, New York, 1996.

[28] Kumar, Akhil, "*G-Tree: A New Data Structure for Organizing Multidimensional Data*", IEEE Trans. on Knowledge and Data Engineering, Volume 6, Number 2, April 1996, pp. 341 - 347.

[29] Lin, King-Ip, H. V. Jagadish, and Christos Faloutsos, "*The TV-Tree: An Index Structure for High-Dimensional Data*", VLDB Journal, Volume 3, 1994, pp. 517-542.

[30] Mehrotra, Rajiv and James E. Gary, "*Similar Shape Retrieval in Shape Data Management*", IEEE Computer, Volume 28, Number 9, September 1995, pp. 57-62.

[31] Meira, S. R. L and A. E. L. Moura, "*A Scripting Language for Multimedia Presentations*", Proc. of the Intl. Conf. on Multimedia Computing and Systems, May 1994, pp. 484-489.

[32] Park, YoungChoon and Forouzan Golshani, "*ImageRoadMap: A New Content-Based Image Retrieval System*", Proc. of the 8th Intl. Conf. on Database and Expert Systems Applications, September 1997, Lecture Notes in Computer Science, Volume 1308, Springer, pp. 225-239.

[33] Ramakrishnan, Raghu, Database Management Systems, WCB McGraw-Hill, Boston Massachusetts, 1998.

[34] Ravela, S. and R. Manmatha, "*Image Retrieval by Appearance*", Proc. of the 20th Intl. Conf. on Research and Development in Information Retrieval, July 1997.

[35] Ritter, Gerhard X. and Joseph N. Wilson, Handbook of Computer Vision Algorithms in Image Algebra, CRC Press, Boca Raton, 1996.

[36] Roussopoulos, Nick, Stephen Kelley, and Frédéric Vincent, "*Nearest-Neighbor Queries*", Proceedings of the 1995 ACM SIGMOD Intl. Conf. on Management of Data, 1995, pp. 71-79.

[37] Schloss, Gerhard A. and Michael Wynblatt, "*Building Temporal Structures in a Layered Multimedia Data Model*", Proc. of the ACM Multimedia '94, October 1994, pp. 271-278.

[38] Schulz-Mirbach, H., H. Burkhardt, and S. Siggelkow, "*Using Invariant Features for Content-Based Data Retrieval*", 1st NOBLESSE Workshop on Nonlinear Methods in Model-Based Image Interpretation, September 1996.

[39] Sclaroff, Stan, Leonid Taycher, and Marco La Cascia, "*ImageRover: A Content-Based Image Browser for the World Wide Web*", Technical Report TR97-005, Boston University, Boston, 1997.

[40] Seidl, Thomas and Hans-Peter Kriegel, "*Optimal Multi-Step k-Nearest Neighbor Search*", Proc. of the 23rd VLDB Conf., August 1997, pp. 506-515.

[41] Seidl, Thomas and Hans-Peter Kriegel, "*Efficient User-Adaptable Similarity Search in Large Multimedia Databases*", Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data, 1998, pp. 154-165.

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

[43] Silberschatz, Abraham, Henry Korth, and S. Sudarshan, Database Systems Concepts, McGraw-Hill, New York, 1997.

[44] Speegle, Greg, "*Views as Metadata in Multimedia Databases*", Proc. of the ACM Multimedia '94 Conference Workshop on Multimedia Database Management Systems, October 1994, pp. 19-26.

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

[46] Speegle, Greg, Xiaojun Wang, and Le Gruenwald, "*A Meta-Structure for Supporting Multimedia Editing in Object-Oriented Databases*", Proc. of the 16th British Natl. Conf. on Databases, July 1998, Lecture Notes in Computer Science, Volume 1405, Springer, pp. 89-102.

[47] Stonebraker, Michael and Joseph Hellerstein, Readings In Database Systems 3, Morgan Kaufmann, San Francisco, California, 1998.

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

[49] Swartz, Jonathan and Brian C. Smith, "*A Resolution Independent Video Language*", Electronic Proceedings of ACM Multimedia '95, November 1995, available http:// www.cs.cornell.edu/zeno/Projects/rivl/Rivl-mm95/mm-95.html.

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

[51] Weiss, Ron, Andrzej Duda, and David K. Gifford, "*Content-Based Access to Algebraic Video*", Proc. of the Intl. Conf. on Multimedia Computing and Systems, May 1994, pp. 140-151.