Speeding up Color-Based Retrieval in Multimedia Database Management Systems that Store Images as Sequences of Editing Operations

Leonard Brown
The University of Texas at Tyler
Department of Computer Science
Tyler, TX, 75799
lbrown@uttyler.edu

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

Abstract

Typically, multimedia database management systems process content-based image retrieval queries by extracting a set of features from each data object as it is inserted into the underlying database. By expressing queries that are based upon these features, users are able to retrieve the data objects back from the database. Previous research has demonstrated that one method of improving the effectiveness of similarity searches in such systems is to augment the underlying database with a set of edited images to allow more flexible matching. Space can be saved by storing the additional images as sequences of editing operations instead of as large binary objects. This paper proposes an approach for processing retrieval queries in such an environment and presents the results of a performance evaluation demonstrating the effectiveness of the approach.

1. Introduction

Due to the availability of faster and more powerful processors and the growth of the popularity of the Web, more and more computer applications are being developed that maintain collections of images and other types of multimedia data. Because multimedia data objects are different than traditional alphanumeric data, a Multimedia DataBase Management System (MMDBMS) has different storage and retrieval requirements from a traditional database management system. For example, images are typically much larger than traditional alphanumeric data elements, so an MMDBMS should utilize efficient storage techniques when it contains a large number of images and other types of graphics objects. In addition, users interpret the content of images when they view them, so an MMDBMS should facilitate searching utilizing that content, which is commonly called Content-Based Image Retrieval (CBIR) [1, 8].

To facilitate CBIR, systems typically extract a set of features and generate a signature for each image in the database, which is then used to represent the image's content. Subsequently, users can pose queries to the MMDBMS requesting images that have specific feature values. The types of features extracted from the database images, then, are dependent upon the properties that best allow users to search them. These properties should reflect the inherent nature of the domain of the application supported by the MMDBMS.

To illustrate how visual properties can support CBIR, consider an application that performs autonomous navigation while driving and therefore needs to recognize images of road signs. When considering such images, it should be noted that many countries around the world have adopted specific color and shape-based conventions for classifying different types of signs. The is because signs with recognizable symbols and colors are easier for people to use than signs with words, and the symbols and colors aid drivers and passengers that are not familiar with the local language [11]. An MMDBMS supporting road sign recognition, then, should provide searching using color and shape-based features since they provide information about the purpose of a sign.

**This material is based upon work supported by (while serving at) the National Science Foundation (NSF). Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.*

The above discussion indicates that a critical component of performing CBIR is the ability to extract features from images. The existing techniques for performing feature extraction are based upon the images being stored in a conventional binary format. Previous research [5, 7] has indicated that it may be possible to improve the effectiveness of a CBIR application by storing some of the images using an alternative format, which is as sequences of editing operations. The purpose of this paper is to present techniques for performing CBIR in this environment.

The rest of the paper is organized as follows. Section 2 discusses how adding images stored as editing operations can improve the effectiveness of a CBIR system. Sections 3 and 4 present two approaches for performing CBIR in such a system. Section 5 presents the results of a performance evaluation comparing the execution times of the two approaches. Finally, Section 6 summarizes this paper and provides directions for future work.

2. Database Augmentation

One issue when performing CBIR is that it is possible that features extracted from two similar images do not match. Many instances of this problem persist as open issues in the CBIR research community. For example, it is difficult to match images of the same object under varying lighting conditions or under varying settings such as outdoor environments [25]. Thus, if an image of an object taken outdoors at night is presented as a query image, it may fail to match an image in the database of the same object. This affects the accuracy of the MMDBMS when processing a similarity search.

One technique for improving the accuracy of CBIR systems is to replace a given query image by several query images as in [23]. Each query image is submitted to the database as a separate query, and the results from all of the individual queries are combined together to form one cumulative result. This technique is somewhat analogous to text retrieval systems that place additional terms in a user's query utilizing a manually produced thesaurus before searching a collection of documents. One problem with this approach is that the features must be extracted from each query image in order to search the underlying MMDBMS. Since feature extraction is a very expensive process, the time needed to respond to process each CBIR query would dramatically increase.

As an alternative to the above approach, an MMDBMS can address the problems of feature matching by augmenting the underlying database with new images derived by editing the original images already present. So, for each image object z in the database, the system will store z along with a set of images created by transforming z using sequences of editing operations. This approach can improve the accuracy of CBIR systems in any situation when a particular database image, say x, is expected to be retrieved in response to a query image q, but the features extracted from q and x do not sufficiently match. The central idea is that the features of q may sufficiently match op(x), where op(x) is created by applying a series of editing operations on image x. This means that if the database is augmented by the addition of op(x), then op(x) can be returned in response to the similarity search query. Furthermore, as long as the MMDBMS maintains a connection between images x and op(x), this connection can be used to determine that x should also be returned in response to the similarity search query even though their respective features do not sufficiently match. While this approach may increase the number of false positives produced by the system, it will decrease the number of false negatives. Avoiding false negatives is often more important than avoiding false positives since a user can filter out unwanted returned images but has no way of knowing a matching database image exists if the system fails to retrieve it [12].

One key benefit of database augmentation becomes apparent when comparing it to traditional CBIR research. Such research typically focuses on either identifying techniques for improving the features extracted from images or improving the process used to compare or classify those features. To implement these techniques in an existing CBIR system, it is necessary to change its internal procedures used to extract and compare features. These changes can be expensive since it often requires purchasing new software or hiring developers to modify the existing source code. In contrast, augmenting databases can improve CBIR without developing new feature extraction techniques. Thus, database augmentation can be an inexpensive method for improving the accuracy of a CBIR system.

A disadvantage of database augmentation is that it increases the number of images stored in the underlying database. This disadvantage is magnified because, as stated earlier, one of the features that distinguish multimedia data from traditional alphanumeric data is that multimedia data objects are much larger. Thus, adding edited images to the database will create a nontrivial increase in the storage required by the MMDBMS.

To minimize the effects of the above disadvantage, an MMDBMS can adopt the technique of storing the additional images as sequences of operations [19, 20] instead of storing them in a conventional binary format such as JPEG [24]. The reason is that an image stored as a set of editing operations will consume much less space than the same image stored in a conventional binary format. Specifically, if an image e is created by editing an original base image object, say b, the edited image is stored as a reference to b along with the sequence of operations used to change b into e. Such an image can be instantiated by accessing the referenced base image and sequentially executing the associated editing operations.

To summarize, in order to improve the accuracy of CBIR, it may be necessary to add edited versions of existing images to the underlying database. So, when an image x is inserted into such a CBIR system, several edited versions of image x should be added to the underlying database as well. These edited images should be stored as sequences of operations to save space. Thus, an MMDBMS that uses database augmentation will store images conventionally and as sequences of editing operations. The remainder of this paper discusses processing queries in this environment.

3. Searching Edited Images by Color

The current methods for extracting features from images require that the images are stored in a binary format. So, in an augmented database, any images stored as editing operations must first be instantiated for the system to use the current methods of feature extraction. Since instantiation is an expensive process in terms of execution time, it should be avoided. Ideally, then, an augmented database management system needs to be able to identify the values of the features in the edited images directly from the sequence of operations used to create it. As a first step toward this goal, we developed a method of determining the color-based features that are contained within an image represented as a sequence of editing operations [4]. To present this method, it is first necessary to describe the conventional method for searching images by color.

3.1. Extracting Color-Based Features

When extracting color features, one common method used by existing systems is to generate a histogram for each image stored in the database where each histogram bin contains the percentage of pixels in that image that are of a particular color. These colors are usually obtained by uniformly quantizing the space of a color model such as RGB, HSV, or Luv into a system-dependent number of divisions. Numerous CBIR systems utilize similar histogram methods to either directly represent or to generate alternative representations for color-based features including BIC [21], DISIMA [16], MARS [17], and RECI [6].

Since each image is represented using a signature computed based on a color 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 at least 25% blue.*" In addition, to search for images that are similar to a query image q, the MMDBMS can extract a signature from q and then compare that signature to the signatures stored in the database. Common functions used to evaluate the similarity between two n-dimensional histograms $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$ include the (1) Histogram Intersection [22] and (2) the L_p-Distances [15]. Additional functions for comparing histograms can be found in [6]. In order to reduce the query processing time, the histograms can be organized in multidimensional indexes such as the R-tree [13] and its numerous variants [3, 10].

$$\sum_{i=1}^{n} \min(x_i, y_i) \tag{1}$$

$$\sqrt[p]{\sum_{i=1}^{n} (x_i - y_i)^p} \tag{2}$$

3.2. Retrieving Edited Images by Color

From the above discussion, the key component of performing color-based retrieval is being able to extract a color histogram from each of the images stored in the database. This involves identifying the values in each bin of the histogram of a given image. We presented a Rule-Based Method (RBM) for identifying these values in edited images without having to instantiate them [4]. RBM uses a set of rules that describe the effects of specific editing operations on the histogram of an image. Specifically, it uses the rules to compute minimum and maximum bounds on the percentage of pixels in an edited image whose colors map to some specified histogram bin, say bin HB.

The above rules are dependent upon the image editing operations that may be used by the system to create the edited images. In our work [4], these operations are restricted to the following set from [2,

20], which contains five operations called Define (DR), Combine (C_1, …, C_9), Modify (RGB_{old}, RGB_{new}), Mutate (M_{11}, …, M_{33}), and Merge (target_image, coordinates). The Define operation selects the group of pixels that will be edited by the subsequent operations in the list, and the parameters to the operation specify the coordinates of the desired group of pixels, called the Defined Region (DR). The Combine operation is used to blur images by changing the colors of the pixels in the DR to the weighted average of the colors of the pixels' neighbors, and the parameters to the operation are the weights (C_1, …, C_9) applied to each of the neighbors C_1 through C_9. The Modify operation is used to explicitly change the colors of the pixels in the DR that are of a certain color, RGB_{old}, into a new color, RGB_{new}. The parameters of the Modify operation specify both RGB_{old} and RGB_{new}. The Mutate operation is used to rearrange pixels within an image, and the parameters specify the matrix (M_{11}, …, M_{33}) used to change the locations of the pixels. This operation can be used to perform rotations, scales, and translations of items within an image. Finally, the Merge operation is used to copy the current DR into a target image, and the parameters specify the target image and the coordinates specifying where to copy the DR. This set of five operations is used because it has the property that its operations can be combined to perform any image transformation by manipulating a single pixel at a time [2].

The purpose of each rule is to determine how its associated editing operation can change a given histogram bin HB. Thus, each rule is expressed as an adjustment to the minimum and maximum bounds on the percentage of pixels that may be in bin HB if the edited image was instantiated. The percentages are adjusted by repeatedly updating the total number of pixels that are in the image as well as the minimum and maximum number of pixels that are in bin HB for each editing operation used to create the edited image. Table 1 summarizes the formulae for computing these adjustments based on the parameters of each operation. In the table, |E| represents the number of pixels in the edited image, |T| represents the number of pixels in the target image of the Merge operation, $|T_{HB}|$ represents the number of pixels in the target that are in bin HB, $|HB|_{min}$ represents the minimum number of pixels in bin HB, and $|HB|_{max}$ represents the maximum number of pixels in bin HB.

Consider using the rules to determine if an edited image E satisfies the given query. A system could access the value of the histogram bin for the referenced base image given in the storage format of E, and then use the above rules to determine how the associated editing operations modify that value. After applying the rules, let the minimum number of pixels that are in bin HB be represented by $BOUND_{min}$, let the maximum number of pixels that are in bin HB be represented by $BOUND_{max}$, and let the size of the image be represented by imageSize. The range [$BOUND_{min}$/imageSize, $BOUND_{max}$/ imageSize] represents the bounds on the percentage of pixels in image E that map to bin HB. If this range does not overlap the desired query range, image E cannot satisfy the given query. Thus, the above rules can be used to eliminate images that do not satisfy a given query without producing false negatives by computing the range [$BOUND_{min}$/imageSize, $BOUND_{max}$/imageSize].

4. Reducing Execution Time

Systems that use conventional approaches such as histograms to retrieve images by color are able to process submitted retrieval queries without having to access each image in the underlying database. This is frequently accomplished by using an index or other type of access method that clusters the data elements into sections of the multidimensional data space of the histograms. Searching is then performed by accessing nodes in the data structure that represent those sections. By quickly identifying sections of the multidimensional space that cannot contain any histograms of images that satisfy the given query, the query processing algorithm can avoid accessing the data elements contained in those sections.

Using a similar idea of reducing query processing time by eliminating data accesses, this section presents a method for speeding up the RBM approach described in the previous section. When using RBM for determining if an edited image satisfies a given color-based query, it is necessary to access each of the image's editing operations and apply the corresponding rules. Thus, this approach must access every edited image in a database as well as every editing operation within each image description to process the submitted query. Following the idea of indexes for multidimensional data objects, this section presents an approach for producing the same query results while reducing the execution time by avoiding having to apply rules for some of the editing operations in the images in the database.

Table 1. Rules for adjusting bounds on numbers of pixels in histogram bin HB

Editing Operation	Conditions	Minimum Number in bin HB	Maximum Number in bin HB	Total Number of Pixels in Image
Combine ($C_{11}, \ldots, C_{33}$)	All	No change	No change	No change
Modify (RGB_{old},RGB_{new})	If RGB_{new} maps to HB	No Change	Increase by |DR|	No Change
	Else if RGB_{old} maps to HB	Decrease by |DR|	No Change	No Change
	Else	No Change	No Change	No Change
Mutate (M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}, M_{31}, M_{32}, M_{33})	DR contains image	Multiply by $\|M_{11}\times M_{22}\|$	Multiply by $\|M_{11}\times M_{22}\|$	Multiply by $\|M_{11}\times M_{22}\|$
	Rigid Body	Decrease by |DR|	Increase by |DR|	No Change
Merge (*Target*, x_p, y_p)	Target is NULL	$\|DR\| - (\|E\| - \|HB\|_{min})$	$MIN[\|HB\|_{max}, \|DR\|]$	|DR|
	Target is Not NULL	$\|DR\| - (\|E\| - \|HB\|_{min}) + \|T_{HB}\| - \|DR\|$	$MIN(\|HB\|_{max}, \|DR\|) + MIN(\|T_{HB}\|, \|T\| - \|DR\|)$	$[MAX((x_p+x_2-x_1), \text{height of Target}) - MIN(x_p,0)+1] \times [MAX((y_p+y_2-y_1), \text{width of Target}) - MIN(y_p,0)+1]$

To present the proposed technique, it is first necessary to consider the characteristics of the rules that are applied for each operation. Each rule produces new maximum and minimum bounds on the percentage of pixels that may be in a given histogram bin for an edited image. The algorithm produces these bounds by computing the maximum number of pixels that are in the histogram bin, the minimum number of pixels that are in the histogram bin, and the total number of pixels in the edited image. So, these three values can be used to identify certain characteristics of the proposed rules.

Several of the proposed rules only increase the maximum bound, $BOUND_{max}$, and decrease the minimum bound, $BOUND_{min}$, on the number of pixels in the bin, while they keep the total number of pixels, imageSize, in the edited image constant. The result is that these rules will only widen the range specified by the minimum bound and maximum bounds. Rules that exhibit this characteristic are called *bound-widening* rules, and they can be used to avoid having to apply rules for some of the operations. Specifically, consider an edited image E that only contains operations with bound-widening rules. During the process of computing the range [$BOUND_{min}$/imageSize, $BOUND_{max}$/imageSize], let its initial values form a range that intersects the range formed by the desired query range. Since all of the rules for the operations in E only increase the range formed by [$BOUND_{min}$/imageSize, $BOUND_{max}$/imageSize], we know that the final computed range will intersect the desired query range [PCT_{min} and PCT_{max}] without having to apply the rules.

The above discussion implies that two conditions are necessary in order to avoid having to apply the rules for an edited image E. First, all of the editing operations in E must have bound-widening rules. Second, the initial value of the range [$BOUND_{min}$/imageSize, $BOUND_{max}$/imageSize] must intersect the desired query range. Note that the initial values of the bounds are taken from the referenced image listed in the description of E. This means that the second condition can be simplified to requiring that the referenced base image of E must satisfy the user's query. If both conditions hold, the query processor does not have to access the rules associated with the operations in E.

So, to reduce the time needed by the RBM query processing approach presented in Section 3, it is necessary to identify which rules are bound-widening, and which images only contain those rules. The rules for the Modify, Combine, and Mutate operations are bound-widening, and the rule for the

Merge operation is bound-widening when the target parameter is null. The system needs to store those images that only contain those operations inside a data structure that used for future query processing. We describe such a data structure in the next section.

```
/* Identify referenced base image of the newly created input edited image E. */
1. Identify the referenced base image B of the edited image E
2. Access the histogram corresponding to B

/* Analyze all of the operations in E to determine if they are all bound-widening */
3. While (E has more ops and E is unmarked)
3.1     Access rule for the next operation in E
3.2     If the rule is not bound-widening
3.2.1       Mark E as unclassified

/* If all operations in E are bound-widening, add E to Main, else add E to Unclassified */
4. If E has been marked as unclassified
4.1     Append identifier of E to Unclassified
5. else
5.1.    Find loc in Main referring to base image
5.2     Append identifier of E to the list of edited images at the above location
```

Figure 1. Insertion algorithm for proposed data structure

4.1. Proposed Data Structure

The proposed data structure consists of two different components called the Main Component and the Unclassified Component. The Main Component contains a list of the edited images which contain operations that have only bound-widening rules proposed for it. These edited images are clustered together based upon the referenced base images that are listed in their respective descriptions, meaning that two edited images are clustered together if and only if they have the same referenced image. Each element of the Main Component is composed of a tuple <*B_id, E_list*> where B_id is the identifier of referenced base image and E_List is the list of identifiers of edited images that were created from modifying B_id. Some of the edited images may have descriptions that contain at least one editing operation, the corresponding rule of which is not bound-widening. The identifiers of such edited images are stored in the Unclassified Component. To process these edited images, the system will still have to use the approach presented in Section 3.

The proposed data structure can be constructed as images are inserted into the database. Each time an image stored in a traditional binary format is inserted, the identifier for its corresponding histogram should be added to the Main Component. The list of identifiers should be kept sorted to make it easier to search for a specific binary image. Once a binary image B is added to the MMDBMS, the system should insert the descriptions of the edited versions of B into the system as well. Each time an edited image is inserted into the database, the system needs to determine whether it should be added to the Main Component or the Unclassified Component. To make this determination, the edited image must be analyzed in order to check if it contains any rules that are not bound-widening. If so, then the identifier of the edited image is added to the Unclassified Component. If all of the rules are bound-widening, then the identifier is added to the cluster in the Main Component corresponding to image B. An algorithm for performing this insertion is displayed in Figure 1.

```
/* Initialize the parameters of the given query */
1. Set results = ∅
2. Input query from user
3. Analyze query to determine parameters query range [PCT_min, PCT_max] and histogram bin HB

/* Identify edited images in Main Component that satisfy the query */
4. For each element <B_id, E_list> in Main
4.1   pixels = the value in bin HB of image E_id

/* If the binary image does satisfy the query, all elements of E_list satisfy the query as well */
4.2   If ((pixels > PCT_min) and (pixels < PCT_max))
4.2.1     Add B_id to results
4.2.2     Add the elements in E_list to results

/* If the binary image does not satisfy the query, compute boundary range for each image given in E_list to determine if it satisfies the query */
4.3   else
4.3.1     For each E in E_list
4.3.1.1       Execute BOUNDS algorithm for E
4.3.1.2       If bounds overlap [PCT_min, PCT_max]
4.3.1.2.1         Add E to set results

/* The boundary range must be computed for each edited image in Unclassified to determine if it satisfies the query as well. */
5. For each element E in Unclassified
5.1   Execute the BOUNDS algorithm for E
5.2   If bounds overlap [PCT_min, PCT_max]
5.2.1     Add E to results
```

Figure 2. Query processing algorithm using proposed data structure

The above data structure can be used in the Bound-Widening Method (BWM) for processing range queries in an augmented MMDBMS without having to ever instantiate the edited images. First, the algorithm, displayed in Figure 2, computes the query parameters HB, PCT_{min}, and PCT_{max}. Next, the algorithm sequentially accesses each cluster in the Main Component and checks if the histogram of the corresponding binary image satisfies the given query. If so, then its identifier along with all of the identifiers of the edited images within the cluster are added to the query's resultant set. If the binary image's histogram does not satisfy the query, then the rules for each operation of the edited images within the cluster will have to be applied as indicated in Section 3. The final step in the algorithm is to apply the rules for each operation of the edited images listed in the Unclassified Component.

5. Performance Evaluation

We implemented RBM and BWM and compared them using various data sets. The implementations were performed using the Perl language on a SUNsparc workstation, and they do not use any commercial software for managing the databases. They do use utilities from the pbmplus [18] package, however, to convert binary images between the text-based ppm format and more commonly used formats such as gif and jpeg. The data sets were obtained from various sites on the Internet. The first data set contains a collection of images of flags around the world [9], and the second contains a collection of images of college football helmets [14]. These data sets were selected because color-based features are extremely important in recognizing both flags and logos. Table 2 lists the parameters of the test for each data set.

The tests compared the average execution time of the algorithms for processing range queries in augmented databases with and without using the data structure presented in Section 4. The average execution time is measured against the percentage of edited images stored in the database. The results of the tests are displayed in Figures 3 and 4 for the flag and helmet data sets, respectively. They indicate that the average execution time of BWM is smaller than the average execution time of RBM. BWM allows the system to process the queries an average of 33.07% faster for the helmet data set and an average of 22.08% faster for the flag data set. Both tests demonstrated, however, that the reduction in time decreased as more images were stored as editing operations. The reason is that the proposed data structure improves execution time when images contain only operations with bound-widening rules. Each edited image containing a non bound-widening operation requires the same processing cost as the algorithm of Section 3. If many of the edited images fall into this category, the added cost of the data structure actually hurts the performance of the query processor.

6. Summary and Future Work

When focusing on improving the methods used by CBIR components of MMDBMSs to search and retrieve images, existing research focuses on developing techniques for improving the feature extraction or comparison functions. Unfortunately, implementing those techniques in an existing system requires changing the internal procedures used by that system to perform those functions. As an alternative, the approach of database augmentation provides the ability to improve the retrieval accuracy of an existing CBIR application without modifying its internal procedures. Implementing the database augmentation approach, however, means that the system will need to store many edited versions of its data objects. In order to save space, these additional object versions can simply be stored as sequences of editing operations.

Table 2. Default values of parameters used in performance evaluation

Description	Helmet	Flag
Number of images in database	551	817
Number of binary images in database	391	466
Number of edited images in database	160	351
Average number of operations within an edited image	4.56	4.99
Number of edited images that contain only operations with bound-widening rules	14	207
Number of edited images that have an operation whose rule is not bound-widening	146	144

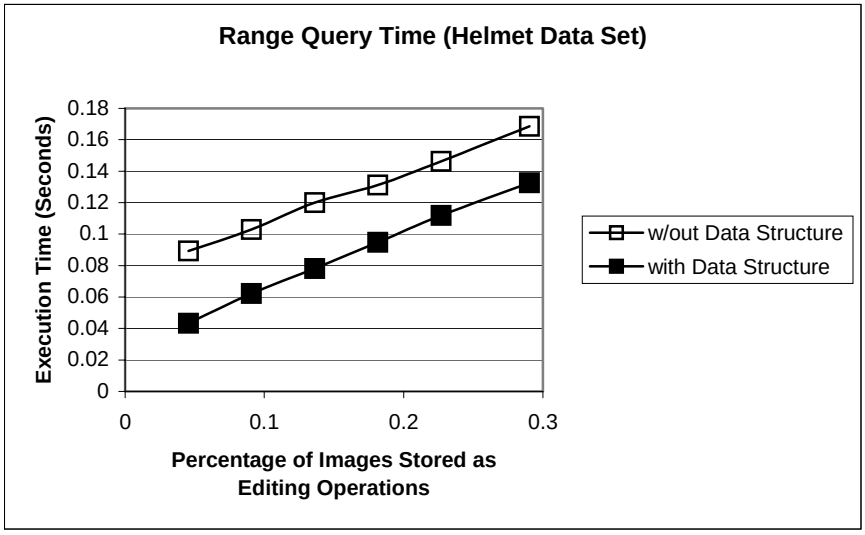

Figure 3. Execution time vs. percentage of images stored as editing operations (helmets)

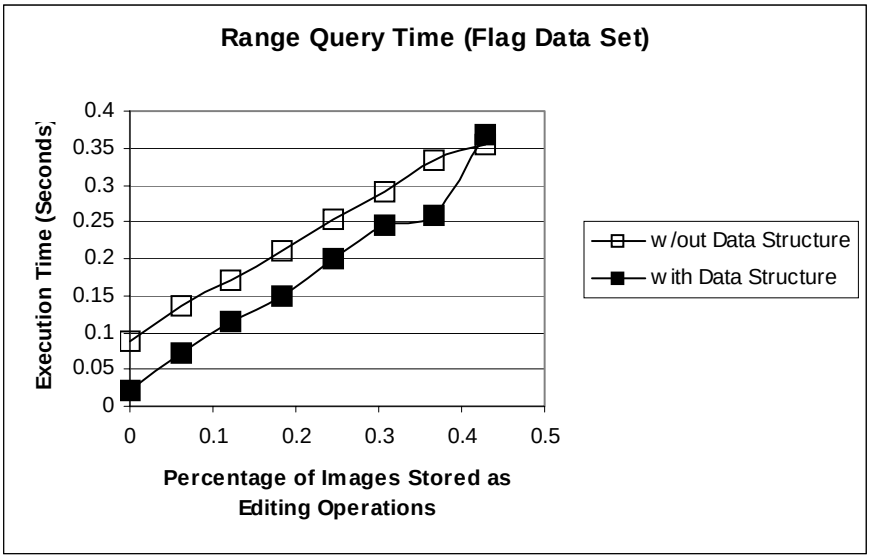

Figure 4. Execution time vs. percentage of images stored as editing operations (flags)

Assuming the edited objects are stored as sequences of editing operations, another research question involves how to search those objects without instantiating them. The approach presented in this paper is another step in answering that question. Specifically, this paper presented an approach for reducing the execution time when processing range queries in a system that uses rules to determine the color features of images stored as editing operations. The approach is based upon identifying whether each rule is a bound-widening rule.

This research focused on searching image applications that are distinguished using color features. It tested the approach on prototype systems that used color histograms to represent images and permitted users to submit range queries in order to retrieve the data. These tests were performed because they serve as the foundation for many different representations of color and many different types of queries. Still, more testing is needed to verify the effects of the proposed data structure on systems that represent color features without histograms and systems that permit other types of

queries including nearest neighbor searches. In addition, in order to allow this approach to be effective for a broader collection of applications, it will be necessary to develop approaches for other common features besides color, such as texture and shape.

7. References

[1] Aslandogan, Y. A. and C. T. Yu, "Techniques and Systems for Image and Video Retrieval", IEEE Transactions on Knowledge and Data Engineering, Volume 11, Number 1, January/February 1999, pp. 56-63.

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

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

[4] Brown, L. and L. Gruenwald, "Performing Color-Based Similarity Searches in Multimedia Database Management Systems Augmented with Derived Images", Proceedings of the 21st British National Conference on Databases, Lecture Notes in Computer Science, Volume 3112, Springer, July 2004, pp. 178-189.

[5] Brown, L., "Issues in Augmenting Image Databases to Improve Processing Content-Based Similarity Searches", Proceedings of the 20th Annual ACM Symposium on Applied Computing, March 2005, pp.1254-1255.

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

[7] Dukkipati, P. and L. Brown, "Improving the Recognition of Geometrical Shapes in Road Signs By Augmenting the Database", Proceedings of the 3rd Intl. Conf. on Computer Science and its Applications, June 2005, pp. 8-13.

[8] Dunckley, L., Multimedia Databases: An Object-Relational Approach, Addison-Wesley, London, 2003.

[9] images from URL http://www.flags.net, last accessed on January 7, 2003.

[10] Gaede, V. and O. Günther, "Multidimensional Access Methods", ACM Computing Surveys, Volume 30, Number 2, June 1998, pp. 170-231.

[11] URL http://www.geocities.com/jusjih/roadsigns.html#d , last accessed May 26, 2005.

[12] Gupta, A. and R. Jain, "Visual Information Retrieval", Comm. of ACM, Vol. 40, No. 5, May 1997, pp. 71-79.

[13] Guttman, A., "R-trees: A Dynamic Index Structure for Spatial Searching", Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 1984, pp. 47-57.

[14] Images obtained from Web, URL http://inside99.net/Helmet_Project/index.htm, last accessed on January 7, 2003.

[15] Jagadish, H. V., "Content-Based Indexing and Retrieval", The Handbook of Multimedia Information Management, Chapter 3, Grosky, Jain, and Mehrotra (Eds.), Prentice Hall, 1997.

[16] Oria, V., et al., "Similarity Queries in the DISIMA Image DBMS", Proceedings of the 9th ACM International Conference on Multimedia, October 2001, pp. 475-478.

[17] Ortega, M. et al., "Supporting Similarity Queries in MARS", Proceedings of the 5th ACM International Conference on Multimedia, 1997, pp. 403-413.

[18] Software obtained from URL http://www.acme.com/software/pbmplus/, last accessed January 7, 2003.

[19] Speegle, G., X. Wang, and L. Gruenwald, "A Meta-Structure for Supporting Multimedia Editing in Object-Oriented Databases", Proceedings of the 16th British National Conference on Databases, July 1998, Lecture Notes in Computer Science, Volume 1405, Springer, pp. 89-102.

[20] Speegle, G. et al., "Extending Databases to Support Image Editing", Proceedings of the IEEE International Conference on Multimedia and Expo, August 2000.

[21] Stehling, R. O., M. A. Nascimento, and A. X. Falcão, "A Compact and Efficient Image Retrieval Approach Based on Border/Interior Pixel Classification", Proceedings of the 11th International Conference on Information and Knowledge Management, November 2002, pp. 102-109.

[22] Swain, M. J. and D. H. Ballard, "Color Indexing", International Journal of Computer Vision, Volume 7, Number 1, 1991, pp. 11-32.

[23] Tahaghoghi S. M. M., J. A. Thorn, and H. E. Williams, "Are Two Pictures Better Than One", Proceedings of the 12th Australasian Conference on Database Technologies, Queensland, Australia, Jan. 2001, pp. 138-144.

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

[25] Zhao, W. et al., "Face Recognition: A Literature Survey", ACM Computing Surveys, Volume 35, Number 4, December 2003, pp. 399-458.