Searching in image databases using the SESAM approach

Jon Olav Hauglid and Roger Midtstraum
Department of Computer and Information Science
Norwegian University of Science and Technology
{joh,roger}@idi.ntnu.no

Abstract

This paper presents a novel approach for content-based searching in image databases. Our system offers the user assistance in locating relevant images in query results by asking the user dynamically generated questions based on color features. The answers to these questions are used to retain only images with the indicated colors. This is an expansion of the previously published SESAM approach which only adresses textual information. We also present a World Wide Web-based prototype which has been successfully tested using a large, real-world database.

1. Introduction

Previously [1], we presented an approach entitled Searching Supported by Analysis of Metadata (SESAM). It assists users in searching in databases by making it possible for the user to remove irrelevant objects from query results. This removal not only decreases the size of the result, is also increases the average relevance of its objects. Using our dynamic and active interface, users specify what parts of the result they find relevant by answering dynamically generated questions. To generate these questions, we use the metadata contained in the data model to find the properties of the objects in the result.

In this paper, we examine how our approach can be expanded to include the use of content-based information extracted from images in an image database. This will allow us to complement questions based on textual properties to improve the overall usefulness of the SESAM approach.

The basics of the SESAM approach are presented in Section 2. In Section 3 and 4 we examine how features are extracted from images and subsequently used to generate questions. Section 5 and 6 describes evaluation and related work, while Section 7 contains conclusions and further work.

2. The SESAM approach

Traditionally, searching in databases has either been done using forms-based interfaces or by issuing queries using a query language (for instance SQL). Both these alternatives are characterized by a tight coupling with the underlying data model. As a result, some understanding of these models is almost a prerequisite for issuing queries. This works well as long as the user-base is restricted to persons who use the database regularly and have been given training in how to use it. However, when databases are published on the World Wide Web (WWW) for the general public, this is no longer the case. Therefore, users must be able to operate a WWW-based system with minimal background knowledge and training in order for it to be successful.

We have earlier [1] examined how a simple interface may be applied to a traditional, text-based, relational database system. Our goal was to design an interface that conformed better to the new requirements of the WWW than the traditional interfaces using forms or query languages. We also presented a World Wide Web- prototype based on this approach. In the remainder of this section, we briefly present the SESAM approach. In later sections we describe how this approach has been enhanced to handle content-based analysis of an image database.

2.1. Standard model of interaction

As observed by Hearst [5], most of the interfaces used for information seeking (including several of the most popular WWW search engines) are based on what he calls the standard model of interaction. Hearst's model is illustrated in Figure 1.

According to this model, users start by issuing a query to the system. The system then performs the search and sends the result back for evaluation. If the result does not contain the sought objects, the user can reformulate the initial query and start the query-process all over again.

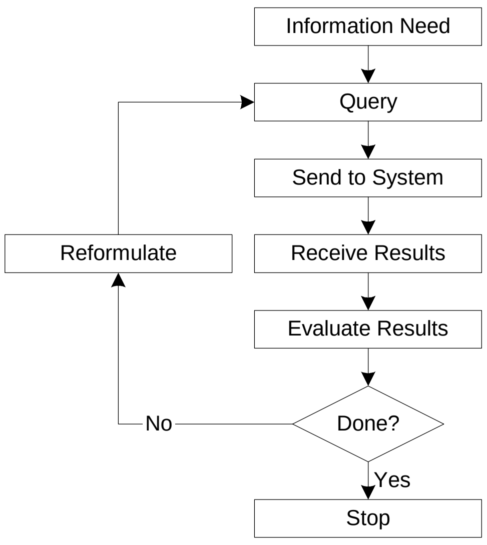

Figure 1: Hearst's standard model of interaction.

A consequence of this model is that the user is offered no help after the result has been presented. If the user is not satisfied with the result, the only options are to reformulate the query or abandon it altogether. Unfortunately, reformulation is often difficult, as the user is given no clues as to which reformulation of the initial query will produce the best result.

If a result does not contain the desired objects, nothing is lost by restarting the query process. On the other hand, if the relevant objects are buried beneath heaps of irrelevant objects, it is wasteful to throw them away. As databases and search results grow larger, this is a more and more likely situation. Few users have the time and motivation to manually browse several thousand objects just to find a handful. Consequently, such results are almost always useless.

We chose to focus on developing a method for handling of excessive search results. In the following section we describe how this can be accomplished by using the knowledge contained in a data model.

2.2. Using a data model

A data model (database schema) is a collection of concepts used to describe the static structure of a database. According to such a model, every object will have a number of properties, which typically includes both attributes and relations to other objects. Each of these properties provides us with a means, or a dimension, to classify the objects. This observation holds for both relational and object-oriented databases.

For each of the available dimensions, the objects in a result take on a number of different values. By counting the occurrences of each of these values, we can summarize the properties of a result in a set of frequency-tables. Suppose a database containing information about cars is searched for all cars whose name includes the word 'Ferrari'. For the objects returned, the describing dimensions might include year of production, color and model. For example, the system can count the number of cars of each color. This gives us the distribution of cars with regard to color as illustrated in Figure 2.

The resulting frequency-tables provide us with means to partition the objects in a result. Each of the partitions contains objects with a specific value in a given property. If we presume that the user is able to select one or several partitions, object not included in any of them can be removed. Thus, the interesting objects are exposed, lessening the need for manual browsing.

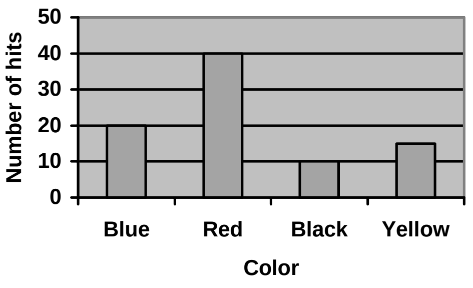

Figure 2: Color-distribution of cars.

In summary, the data model provides opportunities to give users information about properties of the objects in a result. This can in turn be used to retain only objects that conform to specific values of selected properties. This method can therefore greatly improve the handling of large search results.

2.2.1. Previous use of metadata

Using the data model to allow the user to narrow down the result has been investigated by a number of previous studies. In several papers Shneiderman et al. have described concepts called *dynamic queries* [2] and *query previews* [3] that allow users to specify queries using graphical widgets such as sliders while the result is constantly updated with regard to the values selected by the user.

A similar approach, *continuous querying* has been described by Shafer and Agrawal [4]. They present Eureka, a WWW-based database exploration engine. After Eureka has preformed an initial search, users can specify predicates on different attributes and immediately have the result updated. It is also possible to search using example records.

In order to reduce the size of a result, the user must be able to interact with the system to select the relevant properties and the preferred values. The systems mentioned above solve this by presenting all available options to the users – giving them total freedom. This is implemented by the means of lists, sliders, buttons and other graphical widgets that allow individual adjustment of each property value.

This approach is straightforward and well suited as long as the number of dimensions is fairly small. However, as every new dimension requires additional interface components, this approach does not scale well. In fact, a high number of dimensions easily results in a complex and confusing interface. Such interfaces might give experienced users no problems, but novice users are likely to suffer.

A somewhat related problem is that these user interfaces tend to require handcrafting by the designer. This makes modifications and adaptations to new databases more unwieldy. Also, as the type and number of dimensions are static, the interface does not take into account the unique properties of a given query result. If every object in a result is red, it is not useful to ask the user to specify color.

2.3. An active and dynamic interface

We have proposed a new interface where the system plays a more active role. To make interaction as simple as possible, the system presents questions that suggest alternative ways of partitioning the objects in the result. These questions will, if answered by the user, select both dimension and partition. For instance, if a user answers “yes” to a question like ‘Are you interested in red cars?’, the partition containing red cars is selected and cars of other colors can be removed. Conversely, a negative answer implies a selection of all but the red cars.

Our approach leads to a modified model of interaction, where answering questions is an alternative to rewriting the query. Users no longer need to be imaginative in finding a useful reformulation of the initial query. Thus, the process of generating a new and improved result is simplified compared to the method described in Section 2.1.

As our system makes a dynamic evaluation of the utility of the available dimensions, it is possible to keep the number of interface components low. This way, the interface can be uncomplicated even if the number of available dimensions is large.

Naturally, the usefulness of the presented questions becomes very important. If every question goes unanswered, our approach is worthless. It is therefore necessary to develop of a method to estimate their utility on the basis of a given search result. This argument is further strengthened by the fact that there is typically a large set of possible questions. Not only can each object have a great number of properties, it is also possible to imagine many different questions based on a single property. For instance, one could ask about red cars, blue cars etc. This concern is addressed in the following section.

2.4. Comparing questions

A method to compare the utility of questions is critically important. Naturally, it is impossible for the program to ensure that every question is relevant for any given user. Therefore, the best we can do is to apply reasonable heuristics.

We assume that as a given value becomes more frequent, the likelihood that the user knows about this value increases (everyone knows of red Ferraris). Users are more likely to have opinions regarding a known value. They are therefore more likely to answer a question based on such a familiar value. As a result, asking questions about values with high relative frequency, serves us well.

Further, we want as large a reduction the size of the search-result as possible. To attain this goal, we want questions that split the result in roughly equal halves – similar to the approach used in binary search algorithms. By having two halves, the size of the result is drastically reduced no matter which part is selected.

In order to use these two heuristics to compare the utility of different questions, we developed a set of *utility functions*. These functions return a number between 0 (bad) and 1 (good), indicating the usefulness of a question. Each question typically has at least one variable. The utility functions must therefore be applied to every value of each variable to find those that are most suited to be used.

Every object in a result should ideally be described in every dimension. However, in most real-world databases, this is not the case. Frequently, either the complete set of properties of every object is not known, or it makes no sense to describe an object in a particular dimension. Dimensions where many objects have a null-value are not well suited to be used in questions as we really do not know if the known values represent the result accurately. We have taken this into account by lowering utility values with increasing amount of null values.

3. Feature extraction from images

Having explained the basics of the SESAM approach, we now turn to how this approach can be expanded to include content-based questions for image databases.

In order for the SESAM system to be able to generate questions, there must be some means to categorize the different objects in a result. We therefore need to make sure it is possible to compare the different images based on their content. Aigrain et. al. [6] has previously listed the following five different ways of comparing images:

- Color similarity
- Texture similarity
- Shape similarity
- Spatial similarity
- Object presence analysis

In this paper we have chosen to focus on using color information to categorize images. This was done not only because color is easiest to extract, but also due to the importance color has to human perception.

The analysis required to extract color information from images is too time consuming to be made "on the fly" when generating questions. It is therefore necessary to extract and store this information in advance. With this in mind, the remainder of this section is structured shown in Figure 3.

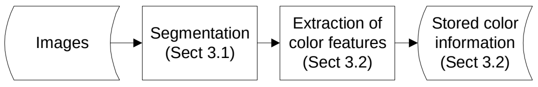
Images
Segmentation (Sect 3.1)
Extraction of color features (Sect 3.2)
Stored color information (Sect 3.2)

Figure 3: Feature extraction from images

Section 4 describes how the extracted color information is used to generate questions.

3.1. Segmentation

The images used in our prototype, is part of the PRIMUS database at the Norwegian Folk Museum. Most of these images show different artifacts shot against a uniform background. A typical example is shown in Figure 4.

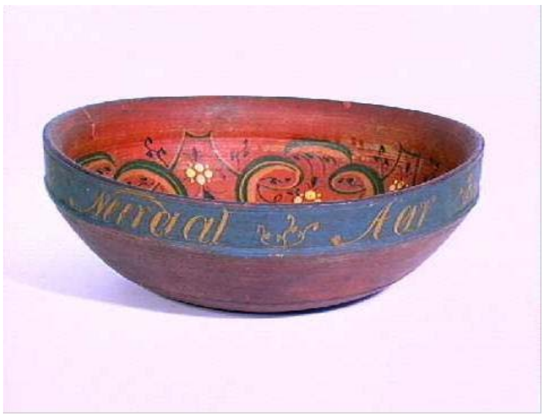

Figure 4: Test image before segmentation

Naturally, we are only interested in color information pertaining to the object and not the background. Therefore, we first have to segment the image to identify the parts of the image that contains the object.

Image segmentation has been the subject of a great deal of research interest. A survey of the different methods used, has previously been published by Skarbek and Koschan [7]. They have grouped the different methods into the following categories:

- Pixel based segmentation
- Area based segmentation
- Edge based segmentation
- Physics based segmentation

We have implemented a pixel-based method based on thresholding histograms. Our method is an adaptation of the method used by Hsu et al. [8].

Segmentation of a general image is very difficult – if not impossible. We simplify the task by exploiting the fact that our images have uniform background by first finding the background color and then assigning the remainder of the pixels to the object. To find the background color, we first quantize the palette of each image into just 64 different colors. Then the background color is identified as the most common color occurring in a border along the edges of the image. The result of applying this algorithm to the image in Figure 4 is shown in Figure 5.

We are fully aware of that more robust algorithms for color image segmentation have been published. However, our selected approach is both simple to implement and executes fast. This is a welcome feature as our database contains more than 235.000 images.

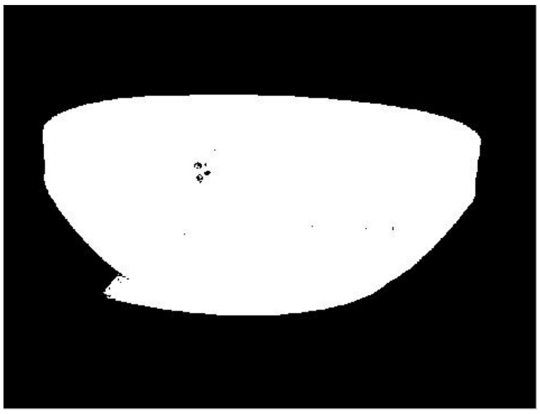
Figure 5: Test image after segmentation

3.2. Color extraction and storage

When the object of interest has been identified, the next task is to find a color representation that captures the color characteristics of the object. Smith and Chang [9] mention *mean color, dominant color* and *color histogram* as typical choices. In addition they propose a new approach called *binary color set*, which includes only those colors that are sufficiently present in the region.

Regardless of representation, there is also a choice of color model (RGB, HSV etc.). For a comparative study of different color models, with regards to image retrieval, we refer to Gevers and Smeulders [10].

Before we decided which representation to use in SESAM, we considered a number of alternatives. Color histograms were discarded as they typically contain more than 100 dimensions and are therefore too computationally expensive. Mean color was informally tested, but was found to be a poor representation of the perceived color of an object. In the end, we decided to first quantize the palette into 512 colors (RGB color space) and then extract the ten most dominant colors of the object. We also count the number of pixels for each of these colors.

When the color information extracted from an image is to be stored in a database, the representation should be as compact as possible. This is important as it improves the retrieval performance. With this in mind, we compress each RGB color into a single number by applying the following formula (R, G and B has values from 0 to 7):

$$color\_value = R \cdot 8^2 + G \cdot 8 + B$$

This formula also makes it simple to revert to the original RGB-values when the data has been read from the database.

4. Color-based questions

As stated in Section 3, our goal is to be able to categorize images based on their color. Having extracted and stored ten dominant colors for each image, we now turn to how this information is to be used to generate questions in SESAM.

The remainder of this section is structured as shown in Figure 6.

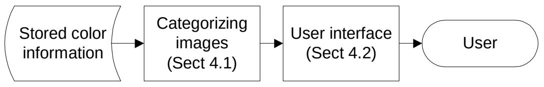

Figure 6: Color-based questions

4.1. Categorizing images

The SESAM approach is based on categorizing (or partitioning) objects in a search result according to a particular descriptor and allowing users to select categories that are of interest. In order to use color information extracted from images, we therefore need to develop a method for categorization. As described in Section 3.2, we have stored the ten most dominant colors for each image. However, in the first version of our prototype, we have selected to use only the most dominant of these. This allows us to keep the design relatively uncomplicated (The impact of using more than one color will be evaluated in Section 5.1).

For each the images present in a given result set, we thus have a single RGB-value or point in the three-dimensional RGB-space. To group images into categories with similar color features, we need to divide the RGB-space containing all the images into subspaces. An illustration of how a sample RGB-space might be divided is shown in Figure 7.

This division is performed by applying the following algorithm:

1. Available_dimensions = (R,G,B), Current_subspace = (0,0,0) – (255,255,255)
2. Select one of the available dimensions and remove this dimension from Available_dimensions.
3. Locate a value in the selected dimension where Current_subspace should be divided.
4. Divide Current_subspace in the selected dimension at the located value.
5. If Available_dimensions != Null, perform step 2-5 for each of the two new subspaces.

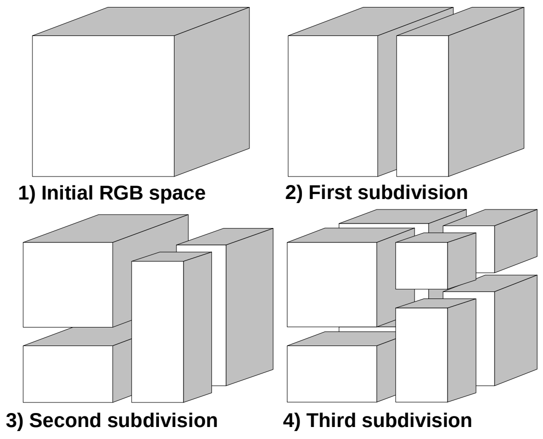

Figure 7: Categorizing images

By varying which dimension is selected from the available dimensions, a total of 12 different divisions of the original RGB-space are possible. As a result, we need both a method to compare different divisions and a method to determine the value selected in step 3.

The chosen solution uses the heuristic presented in Section 2.4 which states that groups (or partitions) of reasonably equals size are desirable as this promotes a large reduction of the result set no matter which groups the user selects. Further, we should try to group images such that the colors present in a given group are fairly uniform.

Suppose that the initial RGB-space is to be divided into to subspaces according to R-value. By counting the number of

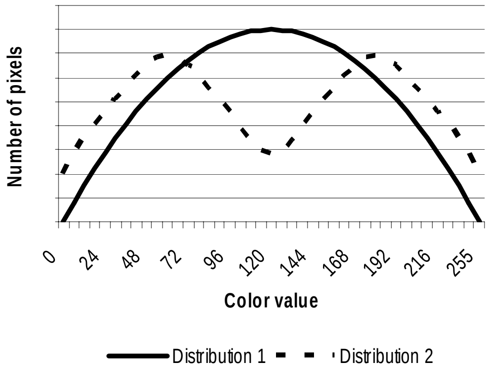

Figure 8: Different distributions of color value

images with each specific R-value, we get a frequency table. Figure 8 shows an illustration of the distribution of two such tables. Both these distributions are symmetrical and would therefore have resulted in the same R-value if we split the distribution in half. While distribution 1 has most of the hits in close proximity to the midpoint, distribution 2 has a larger spread. A division of distribution 2 into two halves will therefore give halves with more uniform images. Thus, it is preferable to split RGB-spaces at color values where few images are located.

To locate the most useful split-value, we used a combination of a weighting function which promotes distributions with most of the hits located far from the selected value and a function which promotes subspaces which contain an equal amount of images.

The overall utility of a subdivision of a RGB-space into eight subspaces (as shown in Figure 7), is the average of the utilities of each of the five divisions involved. This utility is used when the color-based question is compared with other questions as described in Section 2.4.

4.2. User interface

When the images in a result set have been categorized into groups, a representative image is selected for each group. This is done by selecting the image closest to the center of the corresponding RGB-subspace. The resulting set of images is then shown to the user as illustrated in Figure 9.

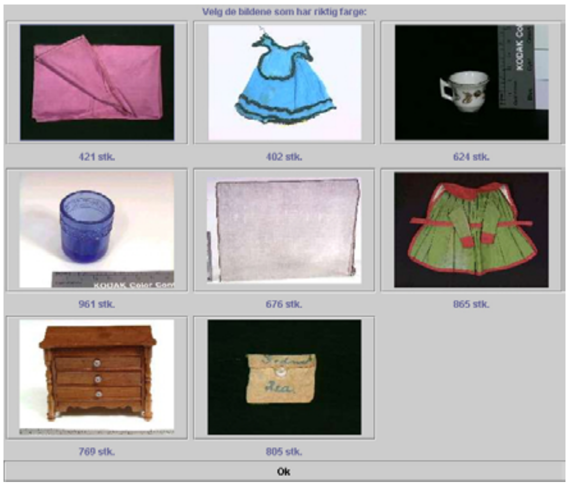

Figure 9: First set of generated images

Using this interface, the user can select those images that have objects with colors that are of interest. The images contained in the groups that are not selected, are removed from the result set. This produces a modified search result, which in turn is used to generate a new image-based question, as shown in Figure 10. New questions will continue to be generated until the user is satisfied with the result, the prototype is unable to generate a question, or the last presented question is left unanswered.

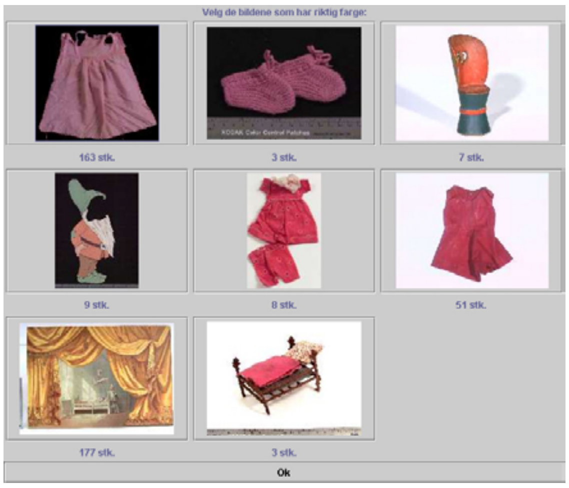

Figure 10: Second set of generated images

The inclusion of these kinds of questions is meant to complement questions based on textual properties. To illustrate how a typical search using SESAM might proceed, consider a user issuing an initial query for ‘Tablecloth’ to a database containing Norwegian artifacts. Conceivable questions generated by SESAM, might then concern material, place of origin (textual properties) or color (image feature). Answers to one or several of the presented questions would then lead to the generation of a new set of questions.

An overview of the different stages in a typical search is shown in Figure 11.

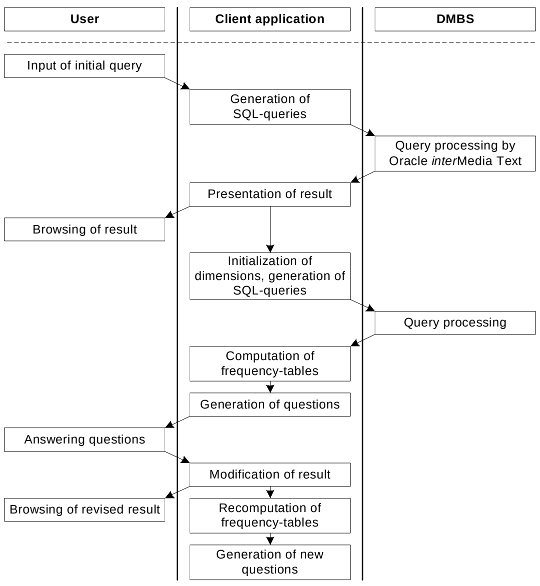
User
Client application
DMBS
Input of initial query
Generation of SQL-queries
Query processing by Oracle *inter*Media Text
Presentation of result
Browsing of result
Initialization of dimensions, generation of SQL-queries
Query processing
Computation of frequency-tables
Generation of questions
Answering questions
Modification of result
Browsing of revised result
Recomputation of frequency-tables
Generation of new questions

Figure 11: Stages in a typical search

5. Evaluation

The PRIMUS database used to evaluate the prototype database belongs to the Norwegian Folk Museum and contains information about more than 260.000 artifacts. In addition to textual descriptions, PRIMUS also contains about 235.000 images.

This section describes how prototype was evaluated with regards to the quality of the implemented color extractor and the performance of the question generator. User experiences with the general SESAM approach are briefly described in [1] and will be topic of further research.

5.1. Color extractor evaluation

The dominant color extractor described in Section 3 is not easily evaluated. This is in part because the objective way of determining the prevalent color (counting pixels with a specific RGB value) not necessarily matches users' subjective perception. After all, humans identify objects as red, blue, gray, etc, and not as having R=35, G=56 and B=128. Therefore, a surface consisting of pixels with slightly different color tone might be perceived as of uniform color. In the cases of such discrepancies between the objective measure and a user's subjective opinion, "the customer is always right". As a result, user testing is necessary in order to evaluate the implemented color extractor.

Further, as "Beauty is in the eye of the beholder", users' perceptions of the color of a given object are likely to differ. An object that is red to someone might seem maroon to others. Therefore it is important to not limit user evaluation to one or two persons. This inability to agree on which color is dominant of course also makes it impossible to achieve a perfect test result.

In the test performed, we evaluated the color extractor with regards to precision and recall [16]. Precision is defined as the degree in which the objects returned by a query system are relevant, while recall indicates how many of the relevant objects were returned. In our setting, a query consists of an image, while the set of images with similar dominant color constitutes the set of relevant objects. User intervention was used to establish the set of relevant images.

The test was implemented as a Java based client-server application. Upon startup of the client, the user is presented with a test image. The user is then asked to indicate which other images have the same dominant color. As the size of the database makes it impossible to browse through every image, the user is only asked to look through a set of 160 randomly drawn images. After the user has selected the set of relevant images, the result is sent to a server where it is compared with the result of the implemented color extractor. Three separate comparisons are done: The first is based on exact match – two images have the same dominant color if they have the same *color_value* (as defined in Section 3.2). The last two comparisons are fuzzy. Here two images have the same even if they have RGB values that are “one step apart” (R±1, G±1, B±1) or “two steps apart”. (R, G and B values still range from 0 to 7).

We tested using just the single most dominant color, as well as the weighted average of the 2, 3, 4 … 10 most dominant colors. In total, 201 tests were performed by 10 different users. The results are shown in Figure 12.

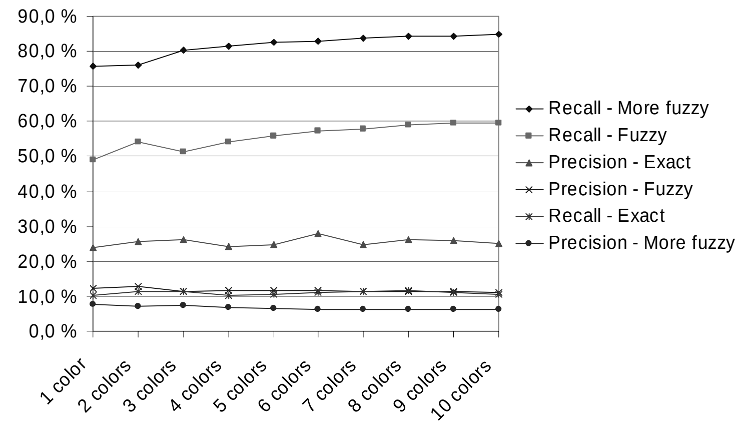

Figure 12: Result of color extractor evaluation

As this graph clearly illustrates, using more than one color had little effect. Further, the precision is too low to make the color extractor useful in anything but coarse classification of images. However, preliminary user evaluations of SESAM indicate that users seldom wish to answer more than one color based question. This is because the images presented in the second question already tend to be too similar.

Several contributing factors to the far from perfect result are mentioned in the start of this section. Another reason is that the color extraction method implemented assumes that images consist of an object displayed on a uniform background. This is only true for most of the images in the PRIMUS database. For the remaining images, the color extracted is often incorrect. A more in-depth study of the results also indicates that the RGB color-space is ill-suited for comparisons with human perception of color. This corresponds with the findings reported in [15]. Further, the implemented test introduces uncertainty both in the determination of the dominant color of the test image and in the determination of the dominant color of the 160 randomly drawn images.

5.2. Performance evaluation

Previously [1], we have presented a detailed performance evaluation of how SESAM performs on a text-only database. In this paper we therefore restrict ourselves to measuring the time spent on image related analysis.

To evaluate the performance of the prototype, we made a list of 180 typical search expressions which on average give a result of about 700 objects. The prototype was modified to randomly draw words from this list and log the time spent on the generation of questions (analysis). The database server used had a AMD Athlon 1,1 GHz processor and 1 GB RAM.

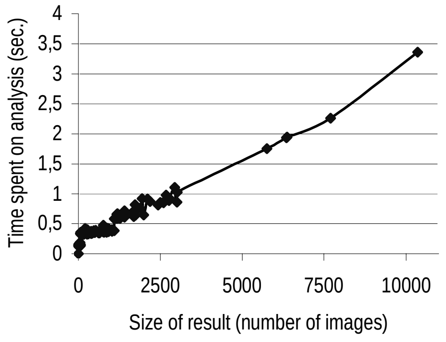

Figure 13: Time spent on analysis

To evaluate the performance, 20.000 queries were performed from a single client. The results are shown in Figure 12. The graph indicates that the time spent on analysis is linearly correlated to the size of the initial result. It also shows that analysis can be made in seconds even with results containing over 10 000 images.

Thus, the analysis can be made without compromising the response time of the system. This conclusion is further supported by the fact that users can browse the initial result while the analysis is performed. Thereby the actual time users spend idle, waiting for the questions, is further shortened. Another helpful feature with regard to the performance, is that the database is only contacted for the first set of questions in a particular search.

6. Related work

Content-based querying based on color features has been described in several earlier papers. One of the most prominent is the QBIC system described in [11] by Flickner et al. Using their system, users can search image databases based on color, shape, textures and sketches. The queries are constructed by graphical means – drawing, selecting colors from a color wheel or selecting a sample image.

Ogle and Stonebraker [12] have presented an approach which integrates traditional relational database retrieval with content analysis techniques. They find that the ability to express queries using this combination of techniques improves the overall image retrieval efficiency. In contrast to QBIC, the content-based query restrictions are specified using textual, not graphical input.

VisualSeek, an image database system which provides for color/spatial querying, has been described by Smith and Chang [13]. Their system allows users to sketch regions, position them in relation to other regions. Each region can be assigned properties of color, size and absolute position. Their studies indicate that usage of spatial information improves the performance of color-based query systems.

Another content-based query engine previously published is Virage [14]. Similarly to QBIC, Virage allows users to issue visual queries based on color, texture and object boundaries. However, Virage also supports user supplied weights for each of these image features.

7. Conclusions and further work

In this paper we have investigated how the SESAM approach can be expanded to consider color information extracted from images in an image database. The SESAM approach is an approach for assisting users in searching in databases. By making it possible for the user to remove irrelevant objects from query results, handling of large results is improved. This removal not only decreases the size of the result, is also increases the average relevance of its objects.

In order to utilize color information, we have implemented a straightforward feature extractor which uses domain knowledge to first segment images into object and background and then extract the ten most dominant colors. This information is stored in a relational database and is subsequently used to generate questions.

Questions are constructed by first categorizing the images of a given search result into groups which exhibits similar color features. The user is presented with a typical image from each of these groups. By selecting one or several of these groups, the images contained in other groups can be removed.

To test our approach, we have used a real-world database containing more then 235.000 images. Current, albeit limited, evaluation of SESAM indicates that the implemented color extractor is only suited for coarse classification of images. However, as users' perception of color differ, coarse classification could be all that is needed. Regardless, further improvements in the extraction of colors should be examined.

We have several other concrete plans for further development of SESAM. The color-based questions examined in this paper utilize the RGB-color space. We plan to evaluate other color spaces (HSV, Munsell etc.) as previous work [15] indicate that they might offer better results. Other image descriptors such as texture and shape will also be evaluated.

8. References

[1] J. O. Hauglid and R. Midtstraum, “SESAM – Searching Supported by Analysis of Metadata”, *Technical Report IDI 4/2001*, Norwegian University of Science and Technology, 2000.

[2] C. Alhberg, C. Williamson and B. Shneiderman, “Dynamic Queries for Information Exploration: An Implementation and Evaluation.”, *Proceedings of the CHI’92 Conference*, ACM Press, May 1992, pp. 619-626.

[3] K. Doan, C. Plaisant and B. Shneiderman, “Query Previews in Networked Information Systems: A Case Study with NASA Environmental Data”, *ACM SIGMOD Record, 6, 1*, March 1996, pp. 75-81.

[4] John C. Shafer and Rakesh Agrawal, “Continuous Querying in Database-Centric Web Applications”, *Proceedings of the WWW9*, Amsterdam, The Netherlands, May 2000

[5] M. A. Hearst, “*Modern Information Retrieval – Chapter 10: User Interfaces and Visualization*”, Addison Wesley Longman, 1999.

[6] P. Aigrain, H. Zhang and D. Petkovic, “Content-Based Representation and Retrieval of Visual Media: A State-of-the-Art Review”, *Multimedia tools and applications*, vol 3, pp. 179-202, Kluwer Academic Publishers, 1996

[7] W. Skarbek and A. Koschan, “Color Image Segmentation – A Survey”, *Techniscer Bericht 94-32*, Technische Universität Berlin.

[8] W. Hsu, Chua T. S. and Pung H.K., “An Integrated Color-Spatial Approach to Content-based Image Retrieval”, *Proceedings of ACM Multimedia ’95*, ACM Press, November 1995, pp. 305-313.

[9] J. R. Smith and S.-F. Chang, “Tools and Techniques for Color Image Retrieval”, *Storage & Retrieval for Image and Video Databases IV*, IS&T/SPIE Proceedings vol. 2670, San Jose, CA, USA, Mar 1996, pp. 426-437.

[10] T. Gevers and A.W.M. Smeulders, “A Comparative Study of Several Color Models for Color Image Invariant Retrieval”, *Proceedings of the 1st International Workshop on Image Databases & Multimedia Search*, Amsterdam, Netherlands, 1996, pp. 17.

[11] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang and B. Dom et al., “Query by Image and Video Content: The QBIC System”, IEEE Computer, 28(9), 1995.

[12] V. E. Ogle, M. Stonebraker, “Chabot: Retrieval from a Relational Database of Images”, IEEE Computer, September 1995.

[13] J. R. Smith and S.-F. Chang. “VisualSEEK: a fully automated content-based image query system”, *Proceedings of ACM Multimedia 96*, ACM Press, November 1996, pp. 87-98.

[14] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur et al., ”The Virage Image Search Engine: An open framework for image management”, *Storage and Retrieval for Still Image and Video Databases IV*, IS&T/SPIE Proceedings vol. 2670, San Jose, CA, USA, Feb 1996, pp.76-87.

[15] Y. Gong, C. H. Chuan, G. Xiaoyi, “Image Indexing and Retrieval Based on Color Histograms”, *Multimedia Tools and Applications*, vol 2, nr. 2, pp. 133-156, Kluwer Academic Publishers, 1996.

[16] V. S. Subrahmanian, *“Principles of Multimedia Database Systems”*, Morgan Kaufmann Publishers Inc, 1998.