Content-Based Sub-Image Retrieval Using Relevance Feedback

Jie Luo
Dept. of Computing Science
University of Alberta, Canada
jieluo@cs.ualberta.ca

Mario A. Nascimento
Dept. of Computing Science
University of Alberta, Canada
mn@cs.ualberta.ca

ABSTRACT

This paper presents the use of relevance feedback to the problem of content-based sub-image retrieval (CBsIR). Relevance feedback is used to improve the accuracy of successive retrievals via a tile re-weighting scheme that assigns penalties to each tile of database images and updates the tile penalties for all relevant images retrieved at each iteration using both the relevant (positive) and irrelevant (negative) images identified by the user. Performance evaluation on a dataset of over 10,000 images shows the effectiveness and efficiency of the proposed framework. Using 64 quantized colors in the RGB color space, the system can achieve a stable average recall value of 70% within the top 20 retrieved (and presented) images after only 5 iterations, with each such iteration taking about 2 seconds.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*image databases*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*abstracting methods, indexing methods*.

General Terms

Algorithms, Experimentation.

Keywords

Content-Based Sub-Image Retrieval, Relevance Feedback, CBIR

1. INTRODUCTION

Most of the content-based image retrieval (CBIR) systems perform retrieval based on a full image comparison, i.e., given a query image the system returns overall similar images. It is not unusual that users are also interested in images from the database that *contain* an image (perhaps an object) similar to a query image (note that there is no restriction as to where the query (sub)image may be within a relevant image). This so-called content-based sub-image retrieval (CBsIR) problem has been defined as follows [6]: given an image query Q and an image database S, retrieve from S those images Q' which *contain* Q according to some notion of similarity. The sub-image retrieval problem we consider is similar to region-based image retrieval (RBIR), e.g. [4][7], since the goal can also be to retrieve images at object-level. However, the difference between these two problems stands out as the CBsIR problem is to search for an image, given as a whole, which is to be contained within another image, whereas in RBIR one is searching for a region, possibly the result of some image segmentation. For instance, in the Blobworld project [4] the query image is segmented into regions and the retrieval task is that of finding database images that have a region similar to a give region from the query image. The CBsIR is more intuitive since users can provide a query image as in traditional CBIR, and unlike RBIR, it does not rely on any type of segmentation preprocessing.

Most early researches on CBIR have been focused on developing effective global features [5], which are not suitable for representing images at object-level. To solve the CBsIR problem, we proposed in [13] an approach called HTM (Hierarchical Tree Matching) which used a tree to model a hierarchical decomposition of an image, encoding the color feature of image tiles which are in turn stored as an index sequence. The retrieval of relevant images is accomplished effectively and efficiently by comparing the query's tree structure with all subtrees of the tree structure for the database images.

The main contribution of this paper is to improve the retrieval performance obtained in [13] by applying two techniques: a more powerful yet compact representation for the tile features to embody the user's perceptions of image content, and relevance feedback (RF), to learn the user's intentions. For the first, we use a recent alternative for CBIR called BIC (Border/Interior pixel Classification) [12]. It depends on a simple yet powerful image analysis algorithm, whose result can be efficiently stored and compared. RF is an interactive learning technique which has been demonstrated to boost performance in CBIR systems [8][9][10]. Despite the great potential of RF shown in CBIR systems using global representations, and also in RBIR systems, to the best of our knowledge there is no research that uses it within CBsIR.

The remainder of this paper is organized as follows. In the next section we briefly review the CBsIR framework introduced in [13], summarize the BIC method for CBIR and

MMDB'04, November 13, 2004, Washington, DC, USA.

how we adopt it for the CBsIR system. The tile re-weighting scheme and the RF strategy using query refinement as well as the incorporation of user's judgement in the image similarity measure are discussed in Section 3. In Section 4, we present and discuss experimental results. Finally, Section 5 concludes the paper and offers directions for future work.

2. BACKGROUND: HTM AND BIC

Region-based image retrieval systems with a similar retrieval goal, e.g., [4, 7], use automatic image segmentation algorithms which usually lead to a super segmentation of the image when trying to achieve homogeneous visual properties. Sometimes the obtained regions are only part of a real object which a user would likely identify by looking at the image and should be combined with some neighbor regions so as to represent a meaningful object. Complex distance functions are generally used to compare poorly segmented images at query time. Also, the number and size of regions per image are variable and a precise representation of the obtained regions may require substantial storage overhead.

Aiming to overcome the issues above, we proposed in [13] a new method called HTM (Hierarchical Tree Matching) for the CBsIR problem. It has three main components: (1) a tree structure that models a hierarchical partition of images into tiles using color features, (2) an index sequence to represent the tree structure (allowing fast access during the search phase), (3) a search strategy based on the tree structures of both database images and the query image.

To model an image, a grid is laid on it yielding a hierarchical partition and tiles. Although granularity could be arbitrary, we have obtained good results using a 4×4 grid resulting in a three-level multiscale representation of the image (similarly to what was done in [3] and [6]). The hierarchial partition of an image and its resulting tree structure are illustrated in Figure 1. There are three levels in the hierarchical structure. The highest level is the image itself. For the second level the image is decomposed into 3×3 rectangles with each side having half the length of the whole image, yielding 9 *overlapping* tiles. The lowest level consists of 4×9=36 rectangles, since each tile of the second level is partitioned into 4 non-overlapping sub-tiles. Note that, to exclude redundance in the CBsIR system, only the indices of the 4×4=16 unique tiles in the lowest level are stored with a small structure for relationship information. The average color of the image tiles in the RGB color space is associated to the nodes in the tree stuctures for images. Thus, every database image is represented as a series of tiles, each of which is mapped to a subtree of the tree modeling the image. It should be noted that although similar, the tree model of the hierarchical partition is *not* the well-known Quadtree [2]. Our tree structure models the *overlapping* tiles at intermediate levels from the hierarchical partition, while the Quadtree is used to describe a class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of *non-overlapping* tiles.

An index sequence representing the predefined parent-child relationship (given by the predefined order of sequence in the index) for the tree structure is stored on secondary storage and used for fast retrieval. Details about the index sequence structure can be found in elsewhere [13]; in short, it resembles a priority tree where the relative order among the tree nodes reflect the relative order of the entries and which can be efficiently mapped onto an array structure. Such an structure allows one to efficiently traverse the necessary indices for computing (sub)image similarity.

The searching process is accomplished by "floating" the tree structure of the query image over the full tree structure of the candidate database image, shrinking the query's tree structure so that it is comparable with the candidate database image's trees at each level of the hierarchical structure. The minimum distance from tree comparisons at all hierarchical levels, indicating the best matching tile from a database image, is used as the distance between the database image and the query. Different from [6], the HTM search strategy considers local information of images' tiles represented by leaf nodes in the subtree structures. The average of distance values among the corresponding leaf nodes is taken for the distance between the tree structures of query image and a certain tile of the database image at any hierarchical level. Experiments detailed in [13] show that this yields better retrieval accuracy compared to related work at the cost of small storage overhead.

The image analysis algorithm of BIC [12] classifies pixels as either border or interior, and two normalized histograms are computed considering only the border pixels and the interior pixels respectively. That is, for each color two histogram bins exist: one in the border pixel histogram and one in the interior pixel histogram. This allows a more informed color distribution abstraction and captures implicitly a notion of texture.

For histogram comparison, the *dLog* distance function is used to diminish the effect that a large value in a single histogram bin dominates the distance between histograms, no matter the relative importance of this single value [5][11]. The basic motivation behind this is based on the observation that classical techniques based on global color histograms treat all colors equally, despite of their relative concentration. However, the perception of stimulus, color in images in particular, is believed to follow a "sigmoidal" curve [11]. The more relative increment in a stimulus is perceived more clearly when the intensity of the stimulus is smaller than when it is larger. For instance, a change from 10% to 20% of a color is perceived more clearly than a change from 85% to 95%. Indeed it has been a well observed phenomena regarding many other phenomena involving how sensitive one is (including animals) to different stimuli [1]. Thus, the distance function is defined as: $dLog(a,b) = \sum_{i=0}^{M} |f(a[i]) - f(b[i])|$ where $f(x) = 0$ if $x = 0$; 1 if $0 < x \leq 1$ or $\lceil log_2 x \rceil + 1$ otherwise, and $a[i]$ and $b[i]$ represent the i^{th} bin of histograms a and b respectively. Note that if we normalize the histograms bins in the [0, 255] range of integer values, instead of usual [0, 1] continuous range, the $f(x)$ function will return integers in the range [0, 9], requiring only 4 bits of storage per histogram bin. This allows substantial reduction in storage, and yet a quite fine discretization of the bins.

The BIC approach was shown to outperform several other CBIR approaches and, as such, we adopt it in the CBsIR system to extract and compare the visual feature of each tile with the goal of improving the retrieval accuracy when compared with the simpler approach adopted in [13], where for each tile only the average color was recorded and used for image indexing. Next, we introduce RF into our CBsIR system (combined with the HTM schema and the BIC method) in order to improve retrieval performance.

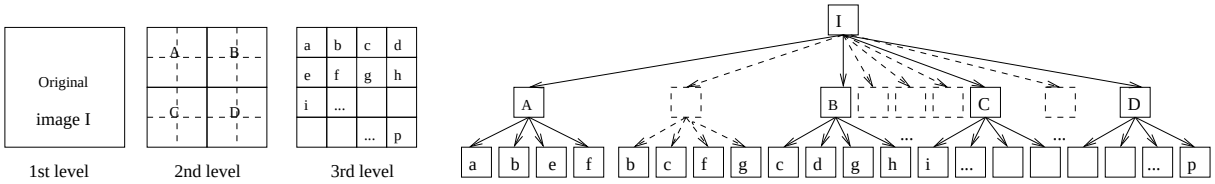

Figure 1: Hierarchical partition of an image and the resulting tree structure.

3. RELEVANCE FEEDBACK FOR CBSIR

Researches in RBIR [9][14] have proposed a region weighting scheme for relevance feedback (RF). In this paper we propose a tile re-weighting scheme that specializes the technique presented in [9] to accomodate our tile-oriented (not region-oriented) HTM approach for CBsIR. It should be emphasized that instead of considering all the images in the database to compute the parameters for region weight [14] (which is computationally expensive), our tile re-weighting scheme uses only the positive and negative examples identified by the user to update the *tile penalty* of the positive images only, which is much more efficient. Moreover, the region re-weighting scheme in [9] uses a predefined similarity threshold to determine whether the region and the image is similar or not, otherwise the comparison of region pairs would become too expensive since images might consist of different and large number of regions. This threshold is sensitive and subject to change for different kinds of image datasets. Thus, how to obtain the right threshold is yet another challenge for the RF method in RBIR. However, our RF method for the CBsIR problem does not need any threshold because the number of obtained tiles is the same and small for each database image and there exists implicit relationship between the tiles, which makes it easier to compare them.

In our system, the user provides feedback information by identifying positive and negative examples from the retrieved images. The basic assumption is that important tiles should appear more often in positive images than unimportant tiles, e.g., "background tiles" should yield to "theme tiles" in positive images. On the other hand, important tiles should appear less often in negative images than unimportant tiles. Following the principle of "more similar means better matched thus less penalty", we assign a *penalty* to every tile that represents the database image for the matching process. The user's feedback information is used to estimate the *tile penalties* for all positive images, which also refines the ranking of images. Note that during the RF iterations, users do not need to specify which tile of a certain positive image is similar to the query[1].

Next, we introduce some definitions used to formalize the overall RF process.

Definition 1: The distance between two tiles T_a and T_b from images I_a and I_b respectively, is:

$$DT(T_a, T_b) = \frac{\sum_{i=1}^{m} Dist(Feature(T_{a_i}), Feature(T_{b_i}))}{m}$$

where m is the number of unique leaf nodes in the tree structures at any hierarchical level (if already at the leaf level, $m = 1$). The $Dist$ function is to be instantiated with some particular distance measure based on the result of the feature extraction done by the $Feature$ function on the tiles, e.g., BIC's $dLog()$ function defined in the previous section. ■

Definition 2: The penalty for a certain tile i from a database image after k iterations is defined as: $TP_i(k), i = 0, \cdots, NT$, where $NT+1$ is the number of tiles per database image, and $TP_i(0)$ is initialized as $\frac{1}{NT+1}$. ■

For instance, in Figure 1, $NT+1 = 1+9+16$, i.e., is equal to the number of nodes in the tree structure representing the hierarchical partition of a database image; for the lowest level, only unique nodes count.

Definition 3: For each tile from a positive image, we define a measure of the distance DTS between tile T and an image set $IS = \{I_1, I_2, \cdots, I_n\}$. This reflects the extent to which the tile is consistent with *other* positive images in the feature space. Intuitively, the smaller this value, the more important this tile is in representing the user's intention.

Assuming that I_i^0 denotes the whole image of image I_i and that I_i^j denotes the j^{th} tile of image I_i, DTS is defined as: $DTS(T, IS) = \sum_{i=1}^{n} exp(DT(T, I_i^0))$, if T is at full tree level; if T is at subtree level, then $DTS(T, IS) = \sum_{i=1}^{n} exp(min_{j=1..NT} DT(T, I_i^j))$, where NT in this case is the number of tiles at the current subtree level. ■

Assuming that I is one of the identified positive example images, we can compute the tile penalty of image I which consists of tiles $\{T_0, T_1, \cdots, T_{NT}\}$. The user provides positive and negative examples during each k^{th} iteration of feedback, denoted respectively as $IS^+(k) = \{I_1^+(k), \cdots, I_p^+(k)\}$ and $IS^-(k) = \{I_1^-(k), \cdots, I_q^-(k)\}$, where $p + q$ is typically much smaller than the size of the database.

Based on the above preparations, we now come to the definition of tile penalty.

Definition 4: For all images (only being positive), the tile penalty of T_i after k iterations of RF is computed (and normalized) as:

$$TP_i(k) = \frac{W_i \times DTS(T_i, IS^+(k))}{\sum_{j=0}^{NT}(W_j \times DTS(T_j, IS^+(k))}$$

where $W_i = 1 - \frac{DTS(T_i, IS^-(k))}{\sum_{j=0}^{NT} DTS(T_j, IS^-(k))}$, acts as a penalty, reflecting the influence of the negative examples. ■

This implies the intuition that a tile from a positive example image should be penalized if it is similar to negative examples. Basically, we compute the distances DTS between a particular tile T and the positive image set IS^+ as well as the negative image set IS^- respectively to update the penalty of that tile from a positive example image. The inverse of the tile's distance from the negative image set is used to weight its corresponding distance from the positive image set.

[1]This would only make the problem simpler at an additional cost to the user. Nonetheless, we plan to address this in the future.

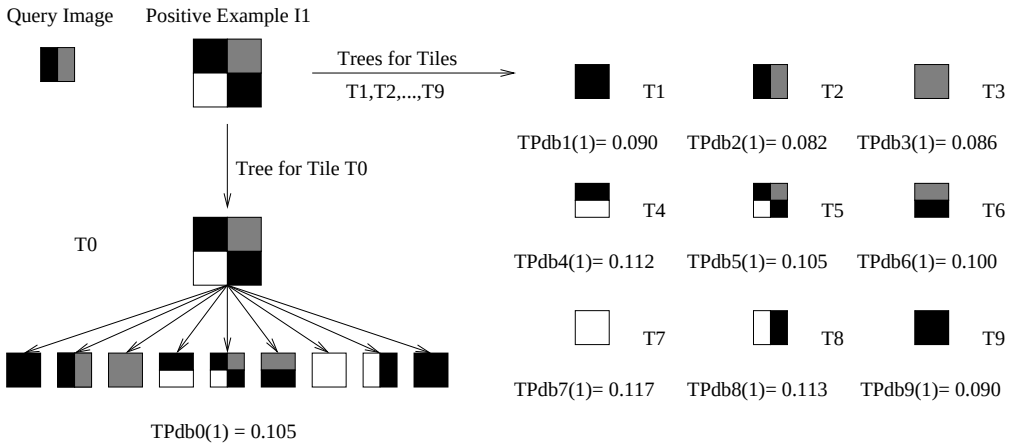

Figure 3: Comparison of tile penalty for database image I_1 before and after feedback.

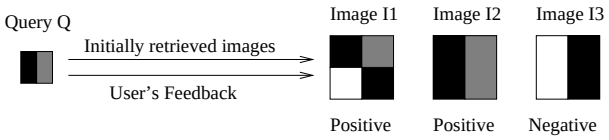

Figure 2: Initial set of retrieved images with user's feedback.

Let us now illustrate the above methodology with a simple example, which also motivates the notion of tile penalty. For simplicity, assume that the color palette consists of only three colors: black, gray and white. Figure 2 shows the top 3 retrieved images and the users' judgement. Image I_1 is marked as a positive example since it actually contains the query image, which exactly represents the sub-image retrieval problem we are dealing with. Image I_2 is also marked as a positive example because it is the enlargement of the query image (and therefore containing it as well).

For the sake of illustration, assume a two-level multiscale representation of database images is used as in Figure 3. The tile penalties for tiles per database image are initialized as 0.1 for the 10 tiles, i.e., $TP_i(0) = 0.1, i \in [0, 9]$. Now, take tile T_1 for example. According to Definition 3, we need to compute the distances DTS between T_1 and the positive/negative image set. In order to do this, firstly, the distances between T_1 and all tiles at the corresponding subtree levels of all the images in the positive/negative image set should be obtained by Definition 1. Then, using Definition 4 the new penalty of T_1 is updated from 0.1 to 0.090 accordingly. The penalties for other tiles are updated in the same way during each feedback iteration. We illustrate the new values of all tile penalties for database image I_1 as a positive example after one feedback iteration in Figure 3. We can see that after the user provides feedback information, some tiles lose some weight while others gain. For instance, T_1, T_2, T_3 and T_9 receive less penalties now because they only contain the color of grey and/or black which is/are also in the query. T_0, T_4, T_5, T_7 and T_8 are penalized more since they all contain the color white. The new weights for these tiles generally follow the trend that more percentage of white color more penalty. T_6, which is a rotation of the query image maintains its weight for this iteration. This means that our system is to some extent also capable of perceiving changes such as rotation.

The RF process using query refinement strategy is based on the tile re-weighting scheme and all positive and negative example images. The main concern is that we need to maintain as much as possible the original feature of query image while introducing new feature elements that would capture more new relevant images. Considering the hierarchical tree structure of the query image, we use the most similar tile (with minimum tile penalty) at every subtree level of each positive image to update the query feature at the corresponding subtree level.

Definition 5: The updated query feature after k iterations is:

$$qn_l^k[j] = \frac{\sum_{i=1}^{p}(1 - TPmin_{i_l}(k)) \times Pos_{i_l}^k[j]}{\sum_{i=1}^{p}(1 - TPmin_{i_l}(k))}$$

where qn_l^k is the new feature with M dimensions for a subtree (tile) at the l^{th} level of the tree structure for the query image after k iterations, $TPmin_{i_l}(k)$ is the minimum tile penalty for a subtree (tile) found at the l^{th} level of the tree structure for the i^{th} positive image after k iterations, $Pos_{i_l}^k$ is the feature for the subtree (tile) with minumum tile penalty at the l^{th} level of the i^{th} positive image's tree structure after k iterations, and p is the number of positive images given by the user at this iteration. ■

Intuitively, we use the weighted average to update the feature for a subtree (tile) of the query, based on the features of those tiles that have minimum tile penalties within the respective positive images. In this way, we try to approach the optimal query that carries the most information needed to retrieve as many relevant images to the query as possible.

With the updated query feature and tile penalties for positive images, we can now define the distance between images and the query for ranking evaluation at each feedback iteration. In order to locate the best match to the query subimage, our image similarity measure tries to find the minumum from the distances between the database image tiles and the query (recall that both the database image and the query sub-image have been modeled by the tree structure in the same way) at corresponding hierarchical level in the tree structure, weighted by the tile penalty of corresponding database image tiles.

Definition 6: The distance between the (updated) query image Q and a database image I at the k^{th} iteration is:

$$DI_k(I,Q) = min_{i=0..NT} TP_i(k-1) \times DT(I_i, Q_j)$$

where $NT+1$ is the number of all subtrees in the tree structure (tiles) of a database image, and $TP_i(k-1)$ is the tile penalty for the i^{th} tile of image I after $k-1$ iterations. ■

For the comparison of full tree structures, $i = 0$ and $j = 0$, indicates both the full tree structure of the database image and the query image. For the comparison of subtree structures, $i = 1..N_l$ for each $1 \leq j \leq (L-1)$, where N_l is the number of subtree structures at the l^{th} level of the tree structure and L is the number of levels of the tree structure, mapped from the hierarchical partition. j indicates the subtree structure at a particular level of the query image's tree structure, as a result of shrinking the original query tree structure to make the comparison with the subtree structures of database images comparable.

Thus, the overall RF process for the CBsIR system can be summarized in the following pseudo algorithm:

1. The user submits a query (sub)-image with no concern about whether the query is a tile or similar to any tile of any database image;
2. The system retrieves the initial set of images using a similarity measure, which consists of database images containing tiles similar to the query sub-image;
3. The system collects the positive and negative feedback examples identified by the user;
4. For each positive image, update the tile penalties of those tiles representing this image using positive examples and negative examples;
5. Update the query using positive images and their newly updated tile penalties;
6. Use the revised query and new tile penalties for database images to compute the ranking score for each image and sort the results;
7. Show the new retrieval results and go to step 3.

4. PERFORMANCE STUDY

We test the proposed relevance feedback approach for the CBsIR system using a broad-domain image dataset. It consists of 10,150 color JPEG images: a mixture of the public Stanford10k[2] dataset and some images from one of COREL's CD-ROMs, each of which falls into a particular category – we use 21 such categories[3]. Some categories do not have rotated or translated images, but others do. On average, each answer set has 11 images, and none of the answer sets has more than 20 images, which is the amount of images we present to the user for feedback during each iteration. We manually crop part of a certain image from each of the above categories to form a query image set of 21 queries (one for each category). Images of the same categories serve as the answer sets for queries—one sample query and respective answer set are shown in the Appendix (Figure 7). The size of the query image varies, being on average 18% the size of the database images. The following performance results are collected from the online demo[4] on a computer running Linux 2.4.17 with two Pentium III CPUs and 256MB of main memory.

For certain applications, it is more useful that the system brings new relevant images (found because of the update of query feature from previous feedback) forward into the top range rather than keeping those already retrieved relevant images again in the current iteration. For other applications, however, the opposite situation applies and the user is more interested in obtaining more relevant images during each iteration including those s/he has already seen before. Besides, it is more helpful that the system learn the user's intention within as fewer iterations as possible. Given these observations, we use two complementary measures for precision and recall as follows:

- *Actual Recall*: the percentage of relevant images retrieved at each iteration over the number of relevant images in the answer set.
- *Actual Precision*: the percentage of relevant images retrieved at each iteration over the number of retrieved images at each iteration.
- *New Recall*: the percentage of relevant images that were not in the set of the relevant images retrieved during previous iterations over the number of relevant images. (Measured only after the first iteration, i.e., after the first feedback cycle.)
- *New Precision*: the percentage of relevant images that were not in the set of the relevant images retrieved during previous iterations over the number of retrieved images. (Also measured after the first iteration.)

The new recall and precision explicitly measure the learning aptitude of the system; ideally it retrieves more new relevant images as soon as possible.

Moreover, we also try to measure the total number of distinct relevant images the system can find during all the feedback iterations. This is a history-based measure that implicitly includes some relevant images "lost" (out of the top presented images) in the process. We call them *cumulative recall* and *cumulative precision* defined as follows:

- *Cumulative Recall*: the percentage of distinct relevant images from all iterations so far (not necessarily shown at the current iteration) over the number of relevant images in the predefined answer set.
- *Cumulative Precision*: the percentage of distinct relevant images from all iterations so far over the number of retrieved images at each iteration.

Table 1 exemplifies the measures mentioned above, assuming the answer set for a query contains 3 images A, B, C and the number of images presented as the answer is 5.

In our experiments, the maximum number of iterations explored is set to 10 (users will give feedback 9 times by pointing out which images are relevant (positive)/irrelevant (negative) to the query) and we present the top 20 retrieved images at each iteration. Note that in our system the series of feedback iterations between queries are independent, i.e.,

[2]http://www-db.stanford.edu/∼wangz/image.vary.jpg.tar.
[3]The union of http://db.cs.ualberta.ca/mn/CBIRone/ and http://db.cs.ualberta.ca/mn/CBIRtwo/
[4]http://db.cs.ualberta.ca/mn/CBsIR.html

Table 1: Illustration of Cumulative/New/Actual Recall and Precision (assuming that 5 images are returned per iteration and that A, B and C are relevant).

Iteration	Retrieved Relevant Images	Cumulative Recall/Precision	New Recall/Precision	Actual Recall/Precision
1	A	33.33%/20%	–/–	33.33%/20%
2	A	33.33%/20%	0%/0%	33.33%/20%
3	B,C	100%/60%	66.67%/40%	66.67%/40%

the information collected from the user is not integrated into the search for the next queries, even if the very same query is submitted to the system again. This consideration is based on the observation of the subjectivity of human perception and the fact that even the same person could perceive the same retrieval result differently at different times.

For the retrieval accuracy of relevant images using 64 quantized colors in the *BIC* method, the results are shown in Figure 4 and Figure 5 by the measures proposed above. As it can be seen from both figures, after only 5 iterations the actual recall and precision values increased by 55% and 60% respectively, then reaching very stable values (also reflected by the new recall and precision curves). It is also noteworthy to mention that the stable actual precision value of nearly 40% is not as low as it may seem at first. The answer sets have an average of 11 images and since the user is presented with 20 images, the maximum precision one could get (on average) would be about 50% as almost half of the displayed images could not be considered relevant by construction. This interpretation leads to the proposal of the following measure:

- *Normalized Precision*: the actual precision over the maximum possible actual precision value.

Interestingly enough, careful consideration of such a measure however shows that is equivalent to the usual notion of (actual) recall. Indeed, consider R and A to be the sets of relevant answers and the retrieved answers with respect to a given query. The actual precision is then defined as $|R \cap A|/|A|$. The maximum precision value one can obtain is $|R|/|A|$. When the former is divided by the latter one obtains $|R \cap A|/|R|$ which is precisely the very definition of actual recall.

This leads to the argument that precision-based measures are not well suited for this type of scenario, where non-relevant images are very likely to be included in the answer set regardless of their relevance. The actual recall, being concerned only with the relevant images is a more realistic measure. Under this argument, 70% of stable actual recall (or normalized precision) seems to be quite reasonable.

We also obtained about 85% for cumulative recall and about 50% for cumulative precision. The reason for the higher values than those for actual recall and actual precision is because some relevant images that may be "lost" in subsequent iterations are always accounted for in these measures.

Figure 6 shows the average cost, measured in seconds, to process a query during each iteration, i.e., to access all disk-resident data, complete the learning from the user's feedback at the current iteration (not applicable to the first iteration), obtain the distance between the query image and database images and sort them by their resulting ranks. When using 64 quantized colors, the first iteration takes, on average,

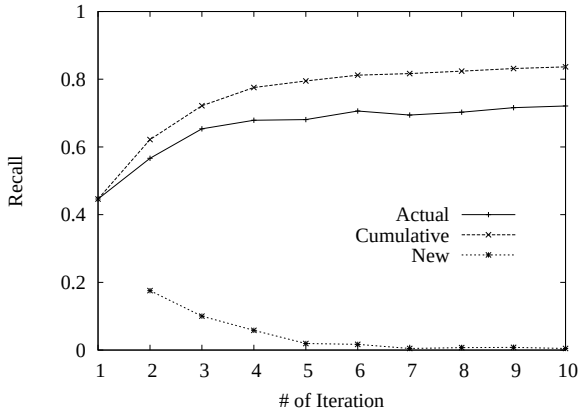

Figure 4: Recall measures.

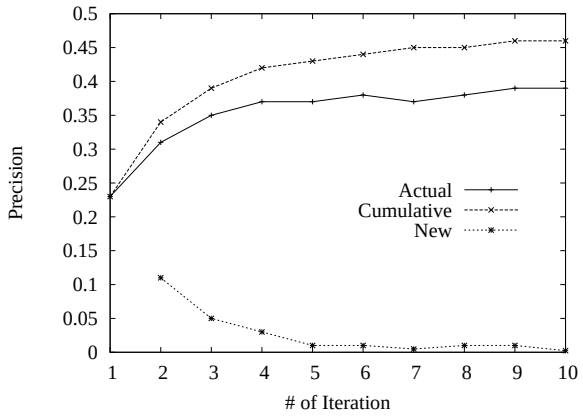

Figure 5: Precision measures.

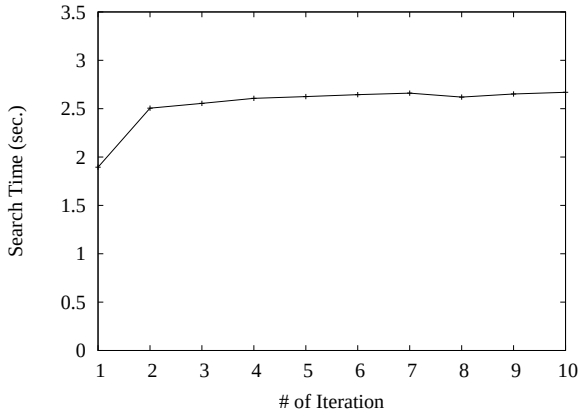

Figure 6: Average processing time of each iteration.

slightly less than 2 seconds, whereas each subsequent iteration requires about 2.5 seconds. This slight increase is due to the overhead for computing and updating the tile penalties.

In order to extract image features from the image database applying the BIC method and generate the metadata file, the use of 64 quantized colors requires about 25 minutes on a computer runing Linux 2.4.20 with AMD Athlon XP 1900+ CPU and 1GB of main memory. This procedure can nevertheless be done off-line. The storage cost for the disk-resident metadata is 10.5 MB (only about 20% the size of the image database).

In summary, our proposed relevance feedback-based approach for content-based sub-image retrieval was able to achieve a very good retrieval accuracy with small space cost and fast retrieval time including the overhead due to the feedback learning.

5. CONCLUSIONS AND FUTURE WORK

In this paper, the query refinement method in relevance feedback is integrated with the CBsIR system by applying a tile re-weighthing scheme to assign penalties to tiles that compose database images so as to better approach the user's intention. The tile penalties of positive images based on both the positive and negative examples identified by users are used to update the query for an improvement of retrieval accuracy of the next iteration. Also, a compact and efficient CBIR approach was used to boost the power of HTM strategy for CBsIR. Experimental results demonstrate the clear performance improvement by this framework compared to that of the previous CBsIR system [13], which used only average color as the feature representation for image tiles and allowed only one iteration of retrieval.

Some venues for future work include integrating other learning algorithms into CBsIR, handling the difference in image resolution between possible queries and target images, and accomplishing a more friendly user interface that allows real time query definition/refinement. In order to address the scalability of the proposed techniques efficient access structures, likely metric ones, for the hierarchical trees are also necessary.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC), Canada, the Alberta Informatics Circle of Research Excellence (iCORE), and the Canadian Cultural Content Management Research Network, a Network financed through Heritage Canada's New Media Research Networks Fund.

6. REFERENCES

[1] J. C. Falmagne. Psychophysical measurement and theory. In *Handbook of Perception and Human Performance, Vol. I*, chapter 1. Willey Interscience, 1986.

[2] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-wesley Publishing Company, Inc, 1990.

[3] K-S. Leung, R. Ng. Multiresolution Subimage Similarity Matching for Large Image Databases. In *Proc. of SPIE - Storage and Retrieval for Image and Video Databases VI*, pages 259–270, 1998.

[4] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proc. of 3rd Intl. Conf. on Visual Information Systems*, pages 509–516, 1999.

[5] G. Lu. *Multimedia Database Management Systems*. Artech House, 1999.

[6] N. Sebe, M. S. Lew, D. P Huijsmans. Multi-Scale Sub-Image Search. In *Proc. of the 7th ACM Intl. Conf. on Multimedia (Part II)*, pages 79–82, 1999.

[7] J. Li, J. Z. Wang, G. Wiederhold. IRM: Intergrated Region Matching for Image Retrieval. In *Proc. of ACM Intl. Conf. on Multimedia*, pages 147–156, 2000.

[8] Y. Rui, T.S. Huang. Optimizing Learning in Image Retrieval. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 236–245, 2000.

[9] F. Jing, B. Zhang, F. Lin, W. Ma, H. Zhang. A Novel Region-Based Image Retrieval Method Using Relevance Feedback. In *Proc. of the 3rd Intl. ACM Workshop on Multimedia Information Retrieval*, pages 28–31, 2001.

[10] T.S. Huang, et al. Learning in Content-Based Image Retrieval. In *Proc. of the 2nd Intl. Conf. on Development and Learning*, pages 155–162, 2002.

[11] M. A. Nascimento, V. Chitkara. Color-Based Image Retrieval using Binary Signatures. In *Proc. of ACM Intl. Symp. on Applied Computing*, pages 687-692, 2002.

[12] R. O. Stehling, M. A. Nascimento, A. X. Falcao. A Compact and Efficient Image Retrieval Approach Based on Border/Interior Pixel Classification. In *Proc. of the 11th Intl. Conf. on Information and Knowledge Management*, pages 102–109, 2002.

[13] J. Luo, M. A. Nascimento. Content Based Sub-Image Retrieval via Hierarchical Tree Matching. In *Proc. of the 1st ACM Intl. Workshop on Multimedia Databases*, pages 63–69, 2003.

[14] F. Jing, M. Li, L. Zhang, H. Zhang, B. Zhang. Learning in Region-Based Image Retrieval. In *Proc. of the 2nd Intl. Conferenence on Image and Video Retrieval*, pages 206–215, 2003.

APPENDIX

Figure 7 shows one example query image and the relevant answer set, i.e., any image outside this set retrieve when this query is posed is considered non-relevant. Note that it is possible that others could find other images also relevant. Unfortunately this is very difficult to take into account unless a larger scale experiment involving several different subjects is deployed.

Figures 8 shows which images of such an answer set, and their respective ranks, were retrieved in the first iteration, i.e., before any feedback was given. (Recall that only the 20 top ranked images are shown to the user.) When these three images are provided as positive examples, the answer set becomes much better. Figure 9 shows now that not only all images previously obtained were ranked higher but also more new images were found and also ranked high. Note that the the actual precision for these two iterations would be 15% and 25% respectively, whereas the actual recall would be 37.5% and 62.5% respectively. We believe the latter to be a more realistic measure of effectiveness since, in the case of this query, actual precision could be not be higher than 40% regardless of how well once could retrieve the relevant images.

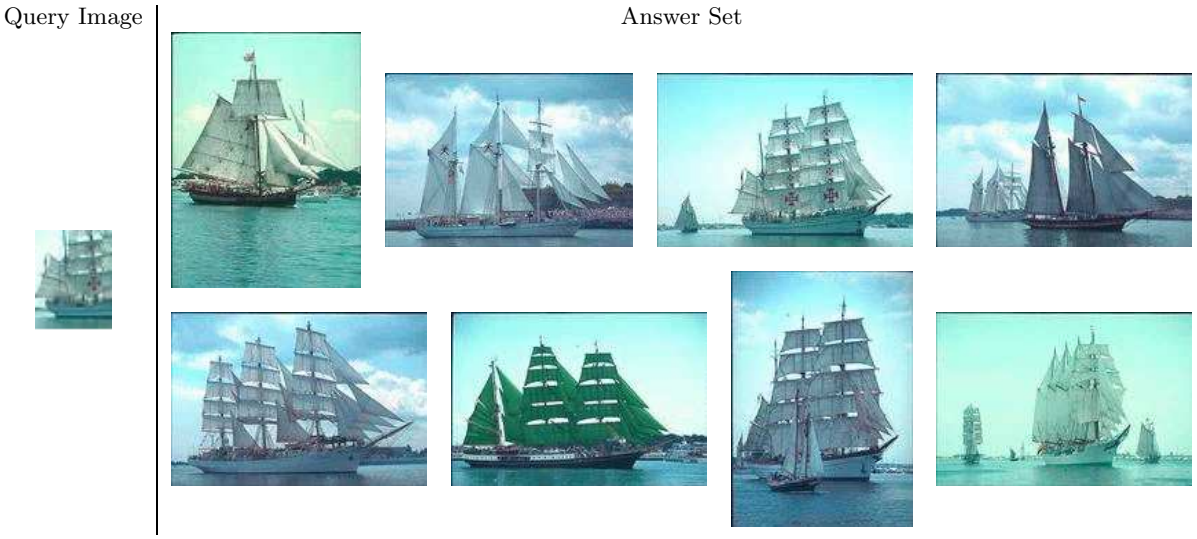

Figure 7: A sample query (sub)image and its relevant answer set.

Figure 8: Rank of the relevant images obtained after the first iteration (no feedback given).

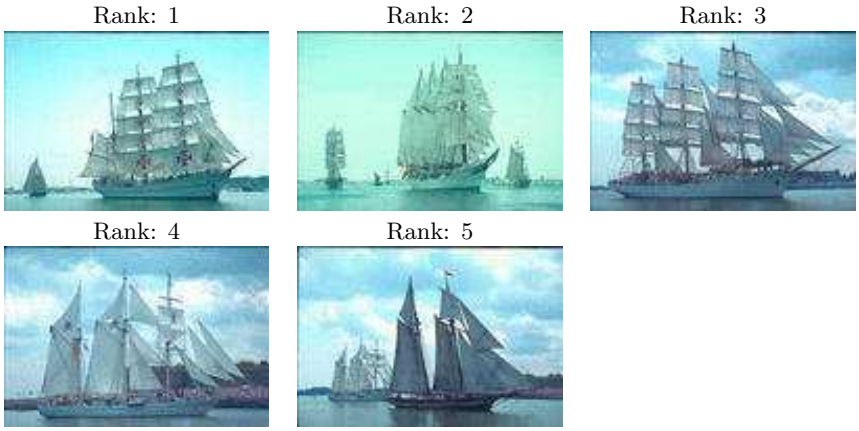

Figure 9: Rank of the relevant images obtained after the second iteration (feedback given once).