Micro Level Attacks In Real-Time Image Processing For An On-Line CBIR Systems

Wichian Premchaiswadi, Anucha Tungkatsathan, *Member, IAENG*

***Abstract*— A cluster computing is popularly used in parallel computing for images retrieval approaches. However, it only attacks this problem at the macro level. In contrast, this paper presents the micro level aspect of the problem by using multi-threading. The framework of multi-threaded processing for an on-line CBIR application is proposed. It enhances the capability of the application including a process of extracting low-level features, feature comparison and image downloading from diverse sources. Although, this paper's main focus on parallel computing techniques for image retrieval. However, we also proposed an efficient color descriptor technique for image feature extraction, namely, Auto Color Correlogram and Correlation (ACCC) to improve efficiency of the image retrieval system and reduce the processing time. Based on the results of the experiment, the use of multiple threads and ACCC algorithm can significantly improve the performance of image indexing and retrieval computation**

***Index Terms*— Mobile Search and Retrieval, Mobile Image Search, Content-Based Image Retrieval, Tourist Information, ACCC Algorithm.**

I. Introduction

The image indexation and similarity measure computation of images are complex processes and they are an obstacle for the development of a practical CBIR system. Especially, when it is developed based on a real-time process optimization approach. There are many researchers are trying to solve this problem by using distributed computing, for instance cluster computing to reduce the computational time [1] [2] [3] [4] [5] [6]. For instance Yongquan Lu, et al [1] presented a parallel technique to perform feature extraction and a similarity comparison of visual features, developed on cluster architecture. The experiments conducted show that a parallel computing technique can be applied that will significantly improve the performance of a retrieval system. Kao, et al [2] proposed a cluster platform, which supports the implementation of retrieval approaches used in CBIR systems. Their paper introduces the basic principles of image retrieval with dynamic feature extraction using cluster platform architecture. The main focus is workload balancing across the cluster with a scheduling heuristic and execution performance measurements of the implemented prototype. Ling and Ouyang [3] proposed a parallel algorithm for semantic concept mapping, which adopts two-stage concept searching method. It increases the speed of computing the low-level feature extraction, latent semantic concept model searching and bridging relationship between image low-level feature and global sharable ontology. Kao [4] presents techniques for parallel multimedia retrieval by considering an image database as an example. The main idea is a distribution of the image data over a large number of nodes enables a parallel processing of the compute intensive operations for dynamic image retrieval. However, it is still a partitioning of the data and the applied strategies for workload balancing. Although, cluster computing is popularly used in images retrieval approaches, it only attacks this problem at the macro level. Especially, to design a distributed algorithm and program it with cross-platform capability is difficult [7]. In contrast, this paper is concerned with the micro level aspect of the problem by using multi-threading. Multi-threading is not the same as distributed processing. Distributed processing which is sometimes called parallel processing and multi-threading are both techniques used to achieve parallelism (and can be used in combination) [8].

Fortunately, with the increasing computational power of modern computers, some of the most time-consuming tasks in image indexing and retrieval are easily parallelized, so that the multi-core architecture in modern CPUs and multi-threaded processing may be exploited to speed up image processing tasks. Moreover, it is possible to incorporate an image analysis algorithm into the text-based image search engines such as Google, Yahoo, and Bing without degrading their response time significantly [7]. We also presents modify advanced algorithm, namely auto color correlogram and correlation (ACCC) [9] based on a color correlogram (CC) [10], for extracting and indexing low-level features of images. The framework of multi-threaded processing for an on-line CBIR application is proposed. It enhances the capability of an application when downloading images and comparing the similarity of retrieved images from diverse sources.

Section II presents the framework of an on-line image retrieval system with multithreading. Section III discusses the proposed indexing technique in older to speed up image

W. Premchaiswadi, Graduate School of Information Technology in Business, Siam University, Bangkok 10163, Thailand (email: wichian@siam.edu)

A. Tungkasthan, Graduate School of Information Technology in Business, Siam University, Bangkok 10163, Thailand (email: aimdala@hotmail.com).

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

processing tasks. The experimental study is presented in Section IV and concluding remarks are set out in Section V.

II. The Proposed Framework of Multithreading for an On-line CBIR System

Before introducing our framework of multi-threading for an on-line CBIR application, we will briefly examine the properties of the queries to be answered. We have developed a novel framework of real-time processing for an on-line CBIR application, using relevance images from Yahoo images. Our method uses the following major steps: (a) Yahoo Images is first used to obtain a large number of images that are returned for a given text-based query; (b) The users select a relevance image and a user's feedback is automatically collected to update the input query for image similarity characterization; (c) A multi-threaded processing method is used to manage and perform data parallelism or loop-level parallelism such as downloading images, extraction of visual features and computation of visual similarity measures; (d) If necessary, users can also change a keyword before selecting a relevance image for the query; (e) The updated queries are further used to adaptively create a new answer for the next set of returned images according to the users' personal preferences. The overview of on-line CBIR application is shown in Fig. 1.

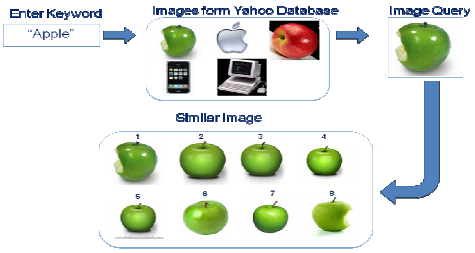


Figure 1. Basic principles of the proposed system

In this section, a multi-threaded processing method is used to carry out parallel processing of multiple threads for a specific purpose. Multi-threading is a way to let programs do more than one thing at a time, implemented within a single program, and running on a single system. In order to utilize the threads more efficiently, the number of threads should be considered and they must technically be assigned to the correct parts of the program. The development of functions, classes, and objects in the program should logically be designed as a sequence of steps. In this research, first, we use the threads to improve the downloading speed for images from various sources according to the locations specified in the xml file that are returned from Yahoo BOSS API [11]. Second, they increase the speed of computing the image feature extraction and similarity measure of feature vectors. The framework of multi-thread processing is presented in Fig. 2. The thread control and the tasks insight of a thread for retrieving images are presented in Fig. 3. and Fig. 4.

An image list control receives the xml files that are returned from Yahoo BOSS API. The Lists of URL can be obtained from the xml files. They are further displayed and used for downloading images from the hosts. An image download module is designed to work in a multithreaded process for downloading images from diverse sources. It is controlled by an image search control module. The image search control module performs a very important function in the management of the system. It fully supports and controls all modules of the online CBIR system. It checks for errors, and the input/output status of each module. Most importantly, it efficiently supports the synchronization of multiple threads that perform image download and similarity measurement by the associated modules. The similarity measurement module performs the computation of the feature vectors and distance metrics of all images that are obtained from the image download module. The image download and similarity measurement modules work concurrently. The query results are recorded into a session of an array in sequential order. The image list object is responsible for the arrangement of all displayed images on the application.

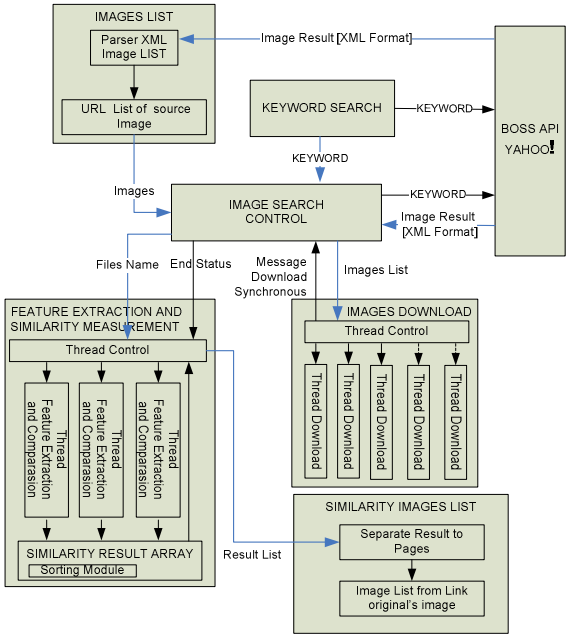


Figure 2. The framework of in Real-Time multi-threaded processing for an on-line CBIR application

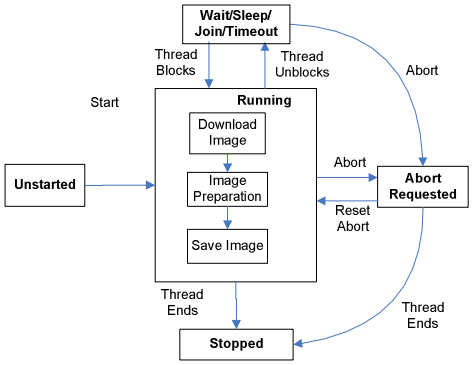


Figure 3. The thread control and the tasks insight of a thread for downloading

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

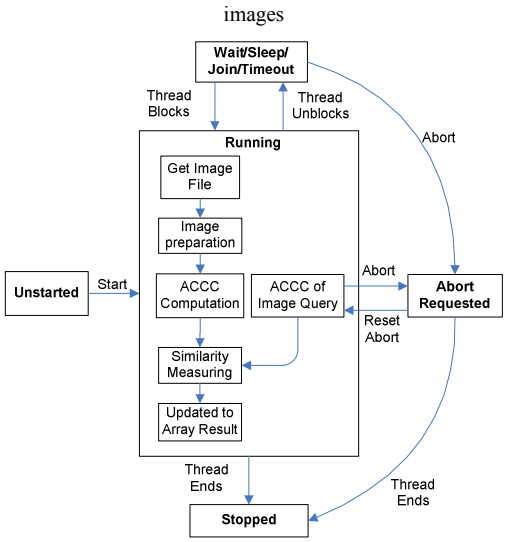


Figure 4. The thread control and the tasks insight of a thread for retrieving images.

III. Feature Computation

This paper's main focus is on parallel computing techniques for image retrieval. The main objective is to reduce the processing time of real-time a CBIR system. However, an efficient color descriptor technique for image feature extraction is still required to reduce the processing time. In this section, we present an efficient algorithm that used in this framework. It is a modifying of the correlogram technique for color indexing. An auto color correlation (ACC) [9] expresses how to compute the mean color of all pixels of color C_j at a distance k-th from a pixel of color C_i in the image. Formally, the ACC of image {I(x,y), x = 1,2,…,M, y = 1,2,…,N } is defined as

$$ACC(i,j,k) = MC_j\gamma_{C_i,C_j}^{(k)}(I) = \left\{r_{mc_j}\gamma_{C_i,C_j}^{(k)}(I),\ g_{mc_j}\gamma_{C_i,C_j}^{(k)}(I),\ b_{mc_j}\gamma_{C_i,C_j}^{(k)}(I) \mid c_i \neq c_j\right\} \quad (1)$$

Where the original image *I(x,y)* is quantized to *J* colors $C_1, C_2, \ldots, C_J$ and the distance between two pixels $d \in [\min\{M,N\}]$ is fixed a priori. Let MC_j is the color mean of color C_j from color C_i at distance d in an image *I*. The mean colors are computed as follows:

$$r_{mc_j}\gamma_{C_i,C_j}^{(d)}(I) = \left(\prod_{i=1}^{N}\Gamma_{C_i,rC_j}^{(d)}(I)\right)^{1/N} \mid c_i \neq c_j$$

$$g_{mc_j}\gamma_{C_i,C_j}^{(d)}(I) = \left(\prod_{i=1}^{N}\Gamma_{C_i,gC_j}^{(d)}(I)\right)^{1/N} \mid c_i \neq c_j$$

$$b_{mc_j}\gamma_{C_i,C_j}^{(d)}(I) = \left(\prod_{i=1}^{N}\Gamma_{C_i,gC_j}^{(d)}(I)\right)^{1/N} \mid c_i \neq c_j \quad (2)$$

where $C_j \neq 0$ and N is the number of accounting color C_j from color C_i at distance k , defined by:

$$N = \Gamma_{C_i,C_j}^{(k)}(I) = \begin{Bmatrix} P(x_1,y_1) \in C_i \mid P(x_2,y_2) \in C_j \\ k = \min\{|x_1 - x_2|, |y_1 - y_2|\} \end{Bmatrix} \quad (3)$$

We propose an extended technique of ACC based on the autocorrelogram, namely Auto Color Correlogram and Correlation (ACCC). It is the integration of Autocorrelogram [6] and Auto Color Correlation techniques [9]. However, The size of ACCC is still O(md). The Auto Color Correlogram and Correlation is defined as

$$ACCC(i,j,k) = \left\{\gamma_{C_i}^{(k)}(I),\ MC_j\gamma_{C_i,C_j}^{(k)}(I)\right\} \quad (4)$$

Let the ACCC pairs for the m color bin be (αi,βi) in I and (α'i,β'i) in I'. The similarity of the images is measured as the distances between the AC's and ACC's d(I,I') and is applied from [12] as follows:

$$d(I,I') = \lambda_1 \sum_{\forall i} \frac{|\alpha_i - \alpha_i'|}{1+\alpha_i+\alpha_i'} + \lambda_2 \sum_{\forall i} \frac{|\beta_i - \beta_i'|}{1+\beta_i+\beta_i'} \quad (5)$$

Where λ_1 and λ_2 are the similarity weighting constants of autocorrelogram and auto color correlation, respectively. In the experiments conducted, $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$. α_1 and α_1 are defined as follows:

$$\alpha_i = \gamma_{C_i}^{(k)}(I)$$

$$\beta_i = \left\{r_{mc_j}\gamma_{C_i,C_j}^{(d)}(I),\ g_{mc_j}\gamma_{C_i,C_j}^{(d)}(I),\ b_{mc_j}\gamma_{C_i,C_j}^{(d)}(I) \mid c_i \neq c_j\right\} \quad (6)$$

The detail of ACC and ACCC algorithms are presented in [9].

IV. Experiment and Evaluation

The experiment focused on studying how multithreads that run on single-core and multi-core CPUs affect the processing time in parallel computing for an on-line image retrieval task and determining the number of suitable threads in single, and multi core processors. However, the experiments that were performed are divided into two groups: In group 1, we studied the performance of multi-thread processing on single-core and multi-core in term of data parallelism for real-time image retrieval tasks. In group 2, we evaluated the retrieval rate for on-line Yahoo image data sets in term of user relevance.

A. Experiment 1: Performance of Multithreading in the Image Retrieval tasks

In the experimental settings, we used one keyword for downloading two hundred images and performed the image search in the same environment (internet speed, time for testing, hardware and software platform). We tested by using forty-nine test keywords in heterogeneous categories including animal, fruit, sunset, nature, and landscape. We performed the image search three times for each keyword and calculated the average

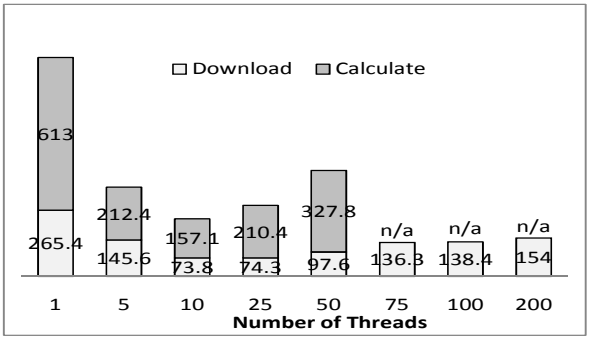
ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

processing time of the whole process for an on-line image retrieval task. The number of downloaded images for each keyword must have a maximum error value less than ten percent of total images. The threads are tested and run on two different hardware platform specifications such as single-core and multi-core CPUs. They are described as follows:

TEABLE I THE AVERAGE TIME IN SECOND OF A WHOLE PROCESS, IMAGE DOWNLOADING, FEATURE EXTRACTION, AND IMAGE COMPARISON AT SUITABLE NUMBER OF THREADS IN EACH PLATFORM (MEAN ± STD-DEV)

w	S-Core 25 threads	Q-Core 50 threads	w	S-Core 25 threads	Q-Core 50 threads
1	159.6±3.3	57.0 ±5.7	26	153.6±9.2	61.0 ±5.1
2	165.6±13.5	57.3 ±6.9	27	156.6±4.9	64.3 ±5.8
3	165.7±15.1	65.7 ±3.9	28	157.6±13.1	68.3 ±3.9
4	148.7±18.9	70.0 ±6.2	29	159.0±11.4	66.0 ±7.8
5	155.6±4.9	75.3 ±7.1	30	161.3±11.8	56.7 ±1.7
6	150.0±13.1	53.3 ±1.7	31	160.3±7.6	51.3 ±4.5
7	159.3±8.5	60.7 ±1.2	32	165.6±16.9	63.3 ±4.2
8	158.0±21.4	65.7 ±2.5	33	153.0±7.5	69.3 ±3.9
9	148.3±17.6	61.3 ±5.4	34	153.3±12.6	59.0 ±4.5
10	160.0±21.9	67.0 ±3.6	35	150.0±10.7	63.6 ±6.0
11	158.3±5.7	71.0 ±2.2	36	158.0±11.3	63.7 ±2.1
12	162.7±10.9	66.3 ±4.0	37	159.6±11.4	60.7 ±6.8
13	155.3±3.4	63.7 ±1.2	38	154.0±5.9	61.0 ±2.9
14	157.7±3.1	62.0 ±1.2	39	157.0±13.4	65.7 ±2.5
15	149.3±4.5	58.7 ±7.5	40	156.6±8.9	62.3 ±4.2
16	152.0±2.3	61.3 ±2.5	41	165.0±11.1	53.7 ±4.8
17	167.7±18.6	66.0 ±7.8	42	164.3±9.7	56.7 ±2.6
18	174.7±15.1	66.0 ±4.9	43	149.3±11.1	56.3 ±3.1
19	171.3±7.6	58.0 ±4.1	44	138.3±6.2	64.3 ±7.8
20	162.0±10.0	71.0 ±9.2	45	148.6±4.0	58.0 ±0.8
21	163.7±3.7	59.3 ±4.5	46	150.0±9.9	57.7 ±6.8
22	162.3±8.3	58.0 ±7.1	47	146.0±7.5	57.0 ±4.3
23	162.0±4.2	56.0 ±5.1	48	145.0±8.5	60.7 ±5.7
24	156.3±16.0	72.3 ±10.2	49	150.0±10.4	54.0 ±3.3
25	154.7±8.9	64.3 ±11.6	**Avg.**	**157.1 ±8.4**	**62.0 ±7.3**

1) Pentium IV single-core 1.8 GHz, and 1 GB RAM DDR2 system. 2) Quad-Core Intel Xeon processor E5310 1.60 GHz, 1066 MHz FSB 1 GB (2 x 512 MB) PC2-5300 DDR2, respectively. The number of threads versus time on single-core and multi-core for an image retrieval process that includes image downloading, feature extraction and image comparison are shown in Fig. 5, Fig. 6, Fig. 7, and Fig. 8. We can conclude that the processing time for the same number of threads in each platform for an image retrieval task is different. However, we selected the most suitable number of threads from the tests on each platform to determine the assumptions underlying a hypothesis test. The results are shown in Table 1. We formulated the hypothesis based on the experiment by computing and using the statistical t-test. In order to measure the significance of the complete processing time obtained by our scheme, we did a t-test on the forty-nine keywords for retrieving images, as shown in Table 1. The mean processing times of single-core and multi-core platforms are 157.12 ±8.4 and 62.0 ±7.3 respectively. Using the t-test to compare the means of two independent CPU platform specifications, the P values obtained from the t-test of single-core versus multi-core is 1.98e-25. A statistical test shows that a multi-core platform significantly consumes less processing time than that of the single-core platform.



n/a = number of downloading images that has an error more than 10% of total images

Figure 5. Number of threads versus time on single-core for image downloading and feature extraction

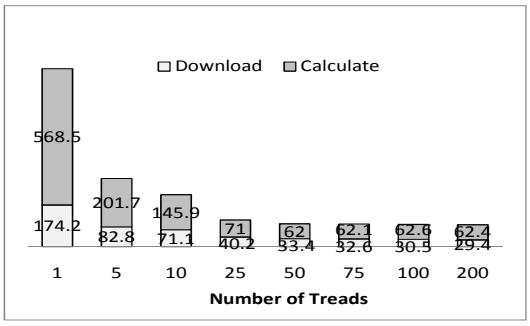


Figure 6. Number of threads versus time on multi-core for image downloading and feature extraction.

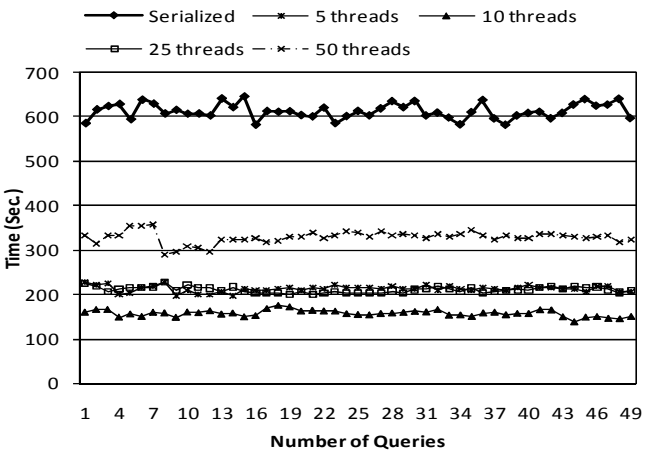


Figure 7. Number of trreads versus time on multi-core in all processes for online CBIR system.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

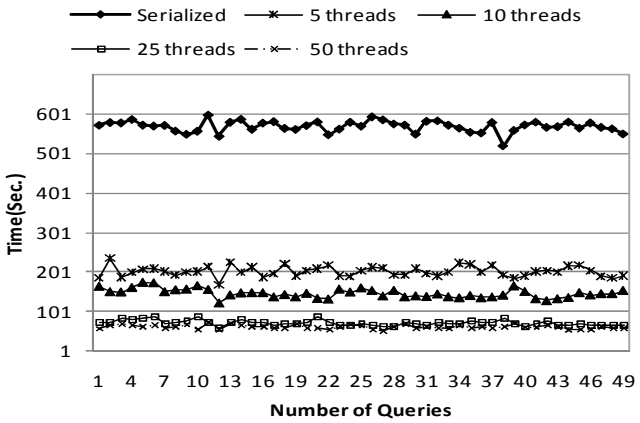


Figure. 8 Number of threads versus time on multi-core in all processes for online CBIR system.

B. *Experiment 2: Performance of Similarity Search*

The goal of this experiment is to show that relevant images can be found after a small number of iterations, the first round is used in this experiment. From the viewpoint of user interface design, precision and recall measures are less appropriate for assessing an interactive system [13]. To evaluate the performance of the system in terms of user feedback, user-orientation measures are used. There have been other design factors proposed such as relative recall, recall effort, and coverage ratio [14]. In this experiment the coverage ratio measure is used. Let R be the set of relevant images of query q and A be the answer set retrieved. Let $|U|$ be the number of relevant images which are known to the user, where $U \subseteq R$. The coverage ratio is the intersection of the set A and U, $|R_k|$ be the number of images in this set. It is defined by equation 10.

$$Coverage\ (C_q) = \frac{|R_k|}{|U|} \tag{7}$$

Let $W_{(q)}$ is the number of keyword used. The average of coverage ratio is by equation 11.

$$Avg\ C_{(\mu)} \frac{1}{N_q} \sum_{i=1}^{N_q} \frac{|R_k|}{|U|} \tag{8}$$

To conduct this experiment, Yahoo Images is first executed to obtain a large number of images returned by a given text-based query. The user selects a relevant image, specific to only one interaction with tzhe user. Those images that are most similar to the new query image are returned. The retrieval performance in term of coverage ratio of the proposed system is compared to the traditional Yahoo text-based search results. The average coverage ratio is generated based on the ACCC algorithm using over 49 test keywords in five categories including animal, fruit, sunset/sunrise, nature, and landscape. The results are presented in table II and Fig. 9.

TABLE 2. COVERAGE RATIO AVERAGE OF THE TOP 24 OF 200 RETRIEVED IMAGES

Sample images	Coverage Ratio				
	Animal	*Fruit*	*Sunset/ Sunrise*	*Nature*	*Lands -cape*
Sample 1	0.71	0.79	0.62	0.64	0.69
Sample 2	0.65	0.71	0.65	0.59	0.65
Avg.	0.68	0.75	0.63	0.62	0.67

The data in table II shows that a proposed scheme using a keyword with the ACCC algorithm can increase the efficiency of image retrieval from the Yahoo image database. Using the combination of text and a user's feedback for an image search, the images that do not correspond with the category are filtered out. It also decreases the opportunity of the images in other categories to be retrieved. In the experiment, we used two sample images obtained from the keyword search to test querying images for evaluating the performance of the system.

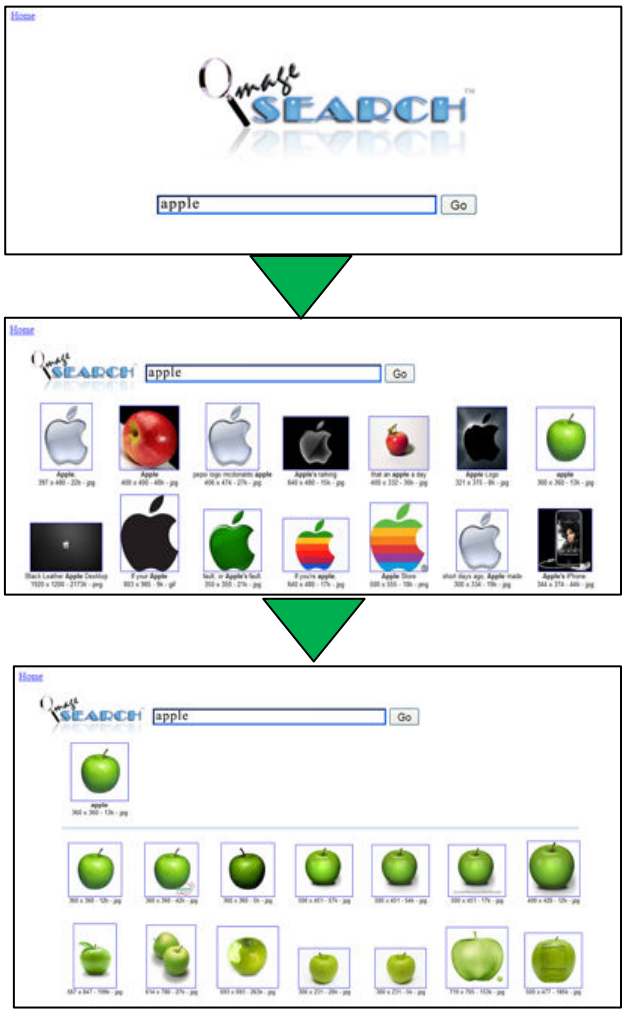


Figure 9. The proposed on-line CBIR system, where the keyword "apple" is used and image results

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. Conclusions

This paper presents a method to bridge the gap between the theory and practice of CBIR by introducing multi-thread processing. In order to reduce the processing time of feature computation, the modified ACCC algorithm was proposed. The experimental results show that multi-threaded processing can improve the speed of the processing time for feature extraction and the measurement of image similarity as well as downloading images from various hosts. The use of multiple threads can significantly improve the performance of image indexing and retrieval on both platforms. However, the number of threads is limited on a single-core platform but has a significant impact on a multi-core platform in this experimental setting. The most suitable number of threads is 10 and 25 to 50 for overall processes (including image downloading, feature extraction and image comparison) of an on-line CBIR System based on single-core and multi-core platform respectively. Additionally, the performance measure of retrieving image accuracy was considered. In the future work, the distributed processing and multi-threading will be used in combination to achieve parallelism.

References

[1] Y. Lu et al. Study of content-based image retrieval using parallel computing technique. Proceedings of the ATIP's. :186–191, 2007
[2] O. Kao et al. Scheduling aspects for image retrieval in cluster-based image databases. Proceedings of First IEEE/ACM. Cluster Computing and the Grid : 329 - 336, 2001
[3] Y. Ling, Y. Ouyang, "Image Semantic Information Retrieval Based on Parallel Computing," CCCM, vol. 1, pp.255-259, 2008.
[4] O. Kao, "Parallel and Distributed Methods for Image Retrieval with Dynamic Feature Extraction on Cluster Architectures," dexa, pp.0110, 2001.
[5] G. Pengdong et al., "Performance Comparison between Color and Spatial Segmentation for Image Retrieval and Its Parallel System Implementation," Proceeding of ISCSCT: 539-543, 2008.
[6] Chris Town and Dr Karl Harrison, "Large-scale Grid Computing for Content-based Image Retrieval," Proceeding of ISKO: 2009
[7] Yuli G, Jianping F, Shinichi S. A novel Approach for Filtering Junk Images from Google Search Results. Springer LNCS 4903 : 1 – 12, 2008.
[8] Multi-Threading in IDL. http:// www.ittvis.com/
[9] T. Anucha el al. Spatial Color Indexing using ACC Algorithms. Proceeding of the ICT&KE: 113-117, 2009.
[10] Huang Jing. et al. Spatial Color Indexing and Applications. Proceeding of Sixth International Conference on Computer Vision. : 606 – 607, 1998.
[11] http://developer.yahoo.com/search/boss/
[12] Lee H. Y., Lee H. K., Ha Y. H., Senior Member, IEEE. Spatial Color Descriptor for Image Retrieval and Video Segmentation. IEEE Trans. Multimedia, 5(3): 358–367, 2003.
[13] B.-Y. Ricardo, and R.-N. Berthier, Modern Information Retrieval, ACM Press Book, 1999.
[14] R. K. Robert, Information Storage and Retrieval, John Wiley & Sons, Inc. 1993.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)