Issue of resource usage in content-based image retrieval algorithms

Vedran Ljubovic, Haris Supic
vljubovic@etf.unsa.ba, hsupic@etf.unsa.ba
Elektrotehnicki fakultet Sarajevo
Zmaja od Bosne bb
71000 Sarajevo, Bosnia and Herzegovina

***Abstract*-Content-based image retrieval (CBIR) is a field of active research for almost 20 years. This timeframe has seen several generations of hardware and corresponding changes in computer usage patterns. It is therefore prudent to periodically reevaluate known methods in the context of modern hardware and usage patterns. Overall the issue of resource usage in CBIR is somewhat neglected. In this paper some extremes in this area are benchmarked and results presented. Specifically, paper is focused on usage scenario of indexing a personal image collection.**

***Keywords:* content-based image retrieval**

I. Introduction

Content-based image retrieval (CBIR) is a study of methods and algorithms used to extract certain features from raster still images, in order to be able to compare and search such images without dependence on accurate metadata. It's a special class of information retrieval (IR) problems.

The block diagram of a CBIR system is given in Figure 1. Typical CBIR system performs two operations: indexing of images (storing their features in a database) and querying this database for an image that presents a closest match. Both operations involve a feature extraction step where image data is analyzed and certain transformations applied to obtain a feature vector. This step is therefore a computational bottleneck in indexing.

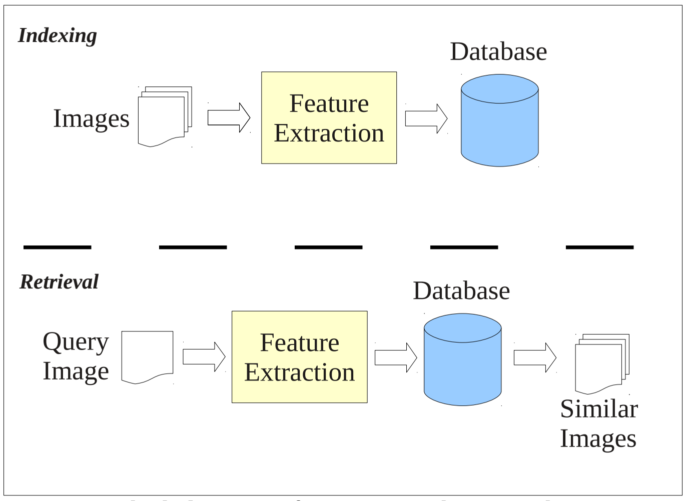


Figure 1: Block diagram of CBIR query-by-example processes

The issue of handling large amounts of traffic in CBIR systems is recognized in literature. [1] Several papers presenting new methods also discuss computational complexity and provide some benchmarks (see e.g. [4], [5], [6]). Some papers discuss use of hardware support to improve processing speed. [7] [1] However, no comprehensive comparative study of CBIR performance and resource usage is provided.

One possible reason for this could be that the problem of indexing a large number of images is an "embarrassingly parallel" problem in that extracting feature vectors from each image is computationally independent from the next image. However, literature [1] notes that a possible use-case for CBIR could be, for example, desktop applications enabling users to index and search their personal image collection. In such applications, responsiveness may prove to be a limiting factor in choice of CBIR approach.

Here it is important to note that many older papers in the area of CBIR are still relevant and useful. However, benchmarking data included in these papers is consistent with hardware abilities and usage patterns at the time of writing of these papers. This time period has seen development of several generations of hardware. It is further noted that even the cheapest digital cameras today provide resolutions in excess of 6 MP (6 million pixels), while popular phones have cameras with resolution between 4 and 8 MP. [8] [9] Also it is noted that most consumer computers today have multi-core processors that support hyperthreading feature.

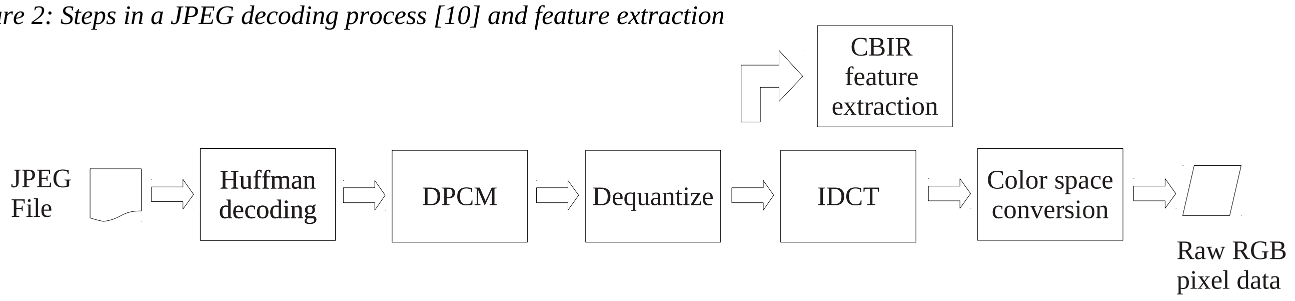
It is therefore prudent to periodically revisit methods presented in older papers that were proven to give useful results and benchmark them in the context of modern hardware and usage patterns.

For brevity, this paper will focus on indexing operation, and thus on feature extraction algorithm. Computational complexity of searching operation seems to be dominated by calculation of distance between feature vectors. [7] Several algorithms from literature are chosen to represent extremes in performance.

II. Compressed Domain Image Retrieval

Discrete Cosine Transform (DCT) is an algorithm closely related to Fourier transform, finding use in multimedia compression. Specifically, DCT is a core element of JPEG

Figure 2: Steps in a JPEG decoding process [10] and feature extraction



compression. [10] Researchers have long ago noted that DCT has certain properties that might be useful in CBIR. [11]

Most images on the Internet are in JPEG format. Specifically, most consumer digital cameras provide photos in JPEG format. Therefore, a number of papers address the possibility of using DCT coefficients from partially decompressed JPEG data as a starting point for feature extraction. These methods are promising to deliver fast and resource efficient feature extraction.

Naturally, such algorithms only work with images in JPEG format. Images in other formats need to first be transcoded into JPEG format.

Some of the more recent papers proposing CBIR schemes operating in JPEG compressed domain are [12], [13], [14] and [15].

Typically, the process of decoding a JPEG file for the purpose of, for example, display on a computer screen consists of the following steps (see Figure 2 and [10]):

1. decode incoming bitstream using Huffman coding algorithm;
2. differential coding of the first (DC) coefficient;
3. zig-zag transformation;
4. dequantization;
5. inverse DCT;
6. translation from Y'CbCr color space into a more common RGB space.

CBIR methods operating in compressed domain terminate this process after step 4 to obtain DCT coefficients pertaining to Y'CbCr color space. In that way, computationally intensive parts of JPEG decoding are skipped and a reduction in CPU usage is obtained compared to algorithms operating in RGB pixel domain. The Y'CbCr color space is a variant of YUV space which more closely models human vision and therefore is better suited for computer vision algorithms.

Most consumer and scientific software uses a reference open-source implementation of JPEG (de/en)coder developed by Independent JPEG Group (IJG), named libjpeg. [16] This decoder is a highly efficient library written in C. It is extremely difficult to match, let alone surpass the performance offered by this library. Therefore, for the purpose of testing image retrieval in compressed domain, a modified version of libjpeg was developed where IDCT and color space transformation are omitted. Such library provides raw DCT coefficients which can then be statistically analyzed and features extracted in an efficient way.

For this paper, algorithm presented in [5] is implemented. This scheme proposes extraction of two feature vectors: First vector provides a color histogram extraction from DCT coefficients representing a reduced resolution image (4x4 blocks). Second vector represents texture and direction information using certain coefficients as inspired by literature. Distance for both vectors is calculated and combined using a weighted formula. Mentioned calculations have very low computational complexity compared to certain other methods, and use a low amount of memory (48 values per image).

With minor modifications to the algorithm that are not relevant for the purpose of this paper, experimental results presented in [5] were reproduced using popular Wang SIMPLIcity dataset. [2]

In order to further stress the multiprocessing abilities of modern computers, a multithreaded version of this application was developed. Four threads are launched at start, each of them extracting features from separate JPEG images.

III. Personal Image Retrieval

As previously mentioned, this paper will focus on the use-case of indexing a personal image collection. To this purpose a number of solutions were evaluated that meet the following criteria: (1) there is a desktop application with a user-friendly interface, suggesting that indexing a personal collection is one potential use-case foreseen by the authors, (2) source code is open so the principle of operation can be verified, (3) the algorithms and methods are published in a peer-reviewed journal.

Also, very outdated systems were not evaluated.

This evaluation produced two viable tools: LIRe [4] and GIFT [17].

LIRe is a flexible Java library for CBIR implementing a range of state of the art methods for feature extraction and comparison. It uses a popular Lucene engine for indexing and search, enabling easy integration into a more general hybrid

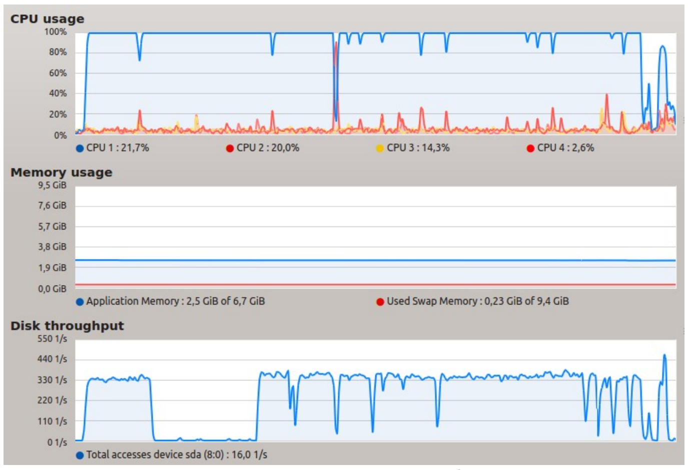


Figure 3: Resource usage in typical run of indexing a personal photo collection using method by Lu et al. (single thread)

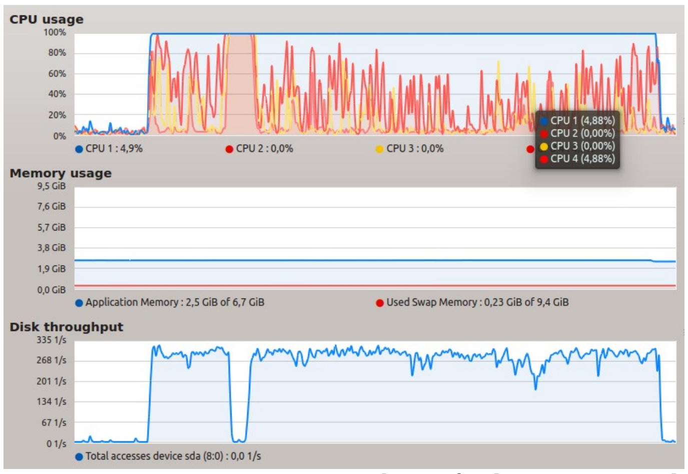


Figure 4: Resource usage in typical run of indexing a personal photo collection using method by Lu et al. (four threads)

multimedia retrieval system.

Through extensive testing of various methods and parameters, a subset of three features is chosen, labeled “fast”, “slow” and “recommended”.

The fastest method offered by LIRe (“fast”) was found to be color histogram in RGB space. This is a well documented and researched CBIR method. JPEG decoding is by far the most computationally intensive part of feature extraction, therefore experimental results for this method can be considered an approximation of a minimal overhead for working in RGB domain.

We further tested this hypothesis by developing a C program that uses an unmodified libjpeg to extract a simple histogram of RGB values with 512 bins from each image. Performance from this program is labeled as “C histogram”.

By default, LIRe uses Color and edge directivity descriptor (CEDD) described in [18]. This method gives overall the best performance in unspecific image search, while offering decent speed, therefore this method was labeled “recommended”.

Finally, color correlograms [19] are used as an example of a

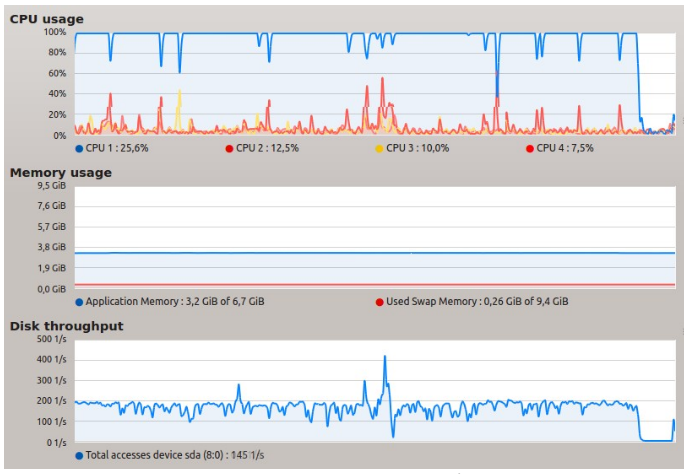


Figure 5: Resource usage in typical run of indexing a personal photo collection using simple color histogram method in C

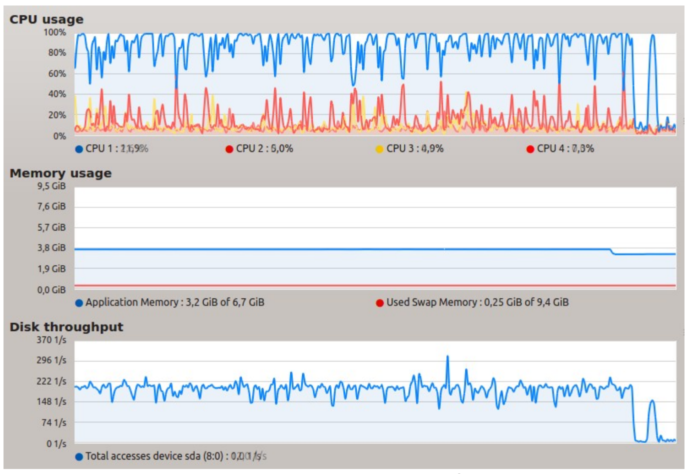


Figure 6: Resource usage in typical run of indexing a personal photo collection using simple color histogram in Java (LIRe)

highly sophisticated “slow” method for image retrieval.

In this paper, version 0.9 of LIRe was used.

The GIFT project is the result of work of the Vision Group (Viper) at CUI (computer science center) of the University of Geneva. This project was officially finished in 2002. and no new releases of GIFT were published since 2005. However the program is still usable and can be downloaded from the GIFT site [20] Version 0.1.14 of GIFT was used in this paper.

There is no configuration options to GIFT indexing part. To have more meaningful results, after each indexing in GIFT the folder gift-indexing-data was deleted.

IV. Testing Results

For the purpose of testing, two datasets were used. Wang SIMPLIcity dataset [2] is often cited in literature and used. It consists of 1000 images. However it must be noted that this dataset was developed in 2001. and therefore it features images of relatively low resolution (256x384 or 384x256 pixels). To simulate workload more representative of modern usage, a second dataset was created consisting of 1427 images, each of 10 MP (10 million pixels) resolution.

All tests were performed on a PC with Intel Core i5 (2400) CPU. This processor features 4 cores and 4 threads, has 6 MB cache and runs at 3.1 GHz. The computer further has 7GB RAM and a 7200 rpm hard disk. Operating system used was Ubuntu Linux 12.04.

Performance testing of LIRe was performed according to the instructions provided by its authors, [21] while C programs were compiled at highest optimization settings available for the platform (-O3) and executed in command line.

Each of the folders were fully indexed with given tools, and the times required are given in Table 1. Tests were repeated five times, mean and standard deviation calculated.

Another possible topic for discussion of LIRe benchmark results is the impact of using Java versus C. To this purpose we created a dummy Java image reader which simply reads images in folder into a BufferedImage object and then discards said object. Test results for Java dummy reader are given as “Java dummy” in Table 1.

Further, Figures 3, 4, 5 and 6 show CPU load (in percent of processor time for each of four cores), memory and swap usage (in GB) and disk throughput (in read/write operations per second). Figure 3 illustrates a typical run of single-threaded version of algorithm by Lu et al. while indexing our second dataset with 1427 10 MP images. Figure 4 uses the multi-threaded version of same algorithm, Figure 5 depicts a typical run of our plain histogram application in C, while Figure 6 depicts same method using LIRe indexer.

Please note that the disk throughput graph is scaled differently between figures.

Graphs for the smaller SIMPLIcity dataset are not provided because, apparently, the entire dataset gets loaded into disk cache and thus disk usage becomes negligible, while CPU usage is fairly constant, and the execution run is too short to provide valuable insights.

Dataset / Method	SIMPLIcity (1000 img @ 100kP)		Personal collection (1427 img @ 10 MP)	
	Mean time	σ	Mean time	σ
Lu et al. (single thread)	814 ms	5,5 ms	160516 ms	519 ms
Lu et al. (four threads)	306 ms	6,3 ms	142884 ms	4572 ms
C histogram	1580 ms	4,5 ms	308717 ms	1176 ms
Java dummy	2457 ms	21,4 ms	288599 ms	1424 ms
LIRe “fast”	4529 ms	140 ms	329804 ms	3921 ms
LIRe “recommend”	11508 ms	43 ms	394331 ms	2511 ms
LIRe “slow”	42059 ms	1047 ms	~14 min.	/
GIFT	196369 ms	421 ms	~half hour	/

Table 1: Mean time and standard deviation of time required to complete indexing operation using methods presented in text

V. Discussion

Table 1 demonstrates that, when faced with a modern usage situation of a personal photo collection, all of the tested CBIR applications delivered poor indexing time. That said, an algorithm in compressed domain (Lu et al.) significantly outperforms even the most primitive algorithm in RGB domain (color histogram). Also, the range between slowest and fastest application is several orders of magnitude.

This demonstrates that further research into algorithms operating in compressed domain is desirable.

The multi-threaded version, as expected, further outperforms all other algorithms. In a typical desktop use-case, user could be able to choose how many indexing threads to launch, enabling a trade-off between faster indexing and higher system responsiveness.

Image input-output in C outperforms Java when working with a large number of smaller files. However, as file size grows to a more realistic 10 megapixels, this difference wanes. Overall it can be concluded that choice of platform doesn't influence performance in a significant way.

Figure 3 shows that, even when working in compressed domain, CPU is still a bottleneck in indexing performance, although less so than with methods operating in RGB space (Figures 5 and 6). This indicates that further work in optimizing this method is desirable.

Figure 4 however suggests that, for the multi-threaded version, disk throughput is becoming a limiting factor. This is also indicated by higher standard deviation for the “Personal collection” dataset, since different disk caching states over a large folder of images gave varying effect on speed.

This can be explained by the fact that the tested CBIR implementation used a fairly naïve approach to multi-threading. An approach where disk reading thread is separated from decompression thread(s) should probably give better parallelization.

Both figures 3 and 4 show negligible memory usage due to very compact feature vectors used in the method by Lu et al. Java approach (LIRe) however features higher memory usage, as indicated by a depression in graph (corresponding to about 512 MB) at the point when program ends. This is due to JVM parameters given in [21] that are optimized for speed rather than memory usage.

Overall memory usage doesn't seem to be a concern given typical memory specifications of modern PCs.

Also it must be noted that JVM performance improves slightly with each subsequent run, which again is reflected in a higher standard deviation for all LIRe tests.

VI. Conclusions and Future Work

Content-based image retrieval (CBIR) is an old field, but is nevertheless facing new challenges. As computer performance increases, the usage patterns change as well, prompting a reevaluation of established methods and algorithms.

This paper demonstrates usefulness of continued research into area of image retrieval methods operating in compressed domain. A broad evaluation of methods operating in compressed domain should be made, from the aspect of precision and recall as well as computer resource usage.

Also, the issue of search performance must be further researched. The use-case of a busy server responding to a large number of queries must be evaluated as well. This requires further inquiry into methods for indexing and distance vector calculation. A number of papers exist on this topic as well that should be critically evaluated.

References

[1] Ritendra Datta, Dhiraj Joshi, Jia Li, James Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age", *ACM Computing Surveys (CSUR)*, Vol 40 Issue 2, April 2008.

[2] James Z. Wang, Jia Li, Gio Wiederhold, "SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 23, No 9, pp. 947-963, 2001.

[3] L. Fei-Fei, P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories", *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 524-531, 2005.

[4] Mathias Lux, Savvas A. Chatzichristofis, "LIRe Lucene Image Retrieval – An Extensible Java CBIR Library", *Proceedings of the 16th ACM International Conference on Multimedia*, pp. 1058-1088, 2008.

[5] Zhe Ming Lu, Su-Zhi Li, Hans Burkhardt, "A content-based image retrieval scheme in JPEG compressed domain", *International Journal of Innovative Computing, Information and Control*, Vol 2, No 4, pp. 831-839, August 2006.

[6] H. B. Kekre, Sudeep D. Threpade, Ashkay Maloo, "Image Retrieval using Fractional Coefficients of Transformed Image using DCT and Walsh Transform", *International Journal of Engineering Science and Technology*, Vol. 2(4), pp. 362-371, 2010.

[7] Rayan Chikhi, Steven Derrien, Auguste Noumsi, Patrice Quinton, "Combining Flash Memory and FPGAs to Efficiently Implement a Massively Paralel Algorithm for Content-Based Image Retrieval", *Reconfigurable Computing: Architectures, Tools and Applications*, Vol. 4419/2007, pp. 247-258, 2007.

[8] http://dpbestflow.org/camera/sensor#megapixels (Retrieved on: August 15th, 2012)

[9] http://www.flickr.com/cameras/ (Retrieved on: August 15th, 2012)

[10] *Information Technology – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines*, ITU CCITT Recommendation T.81, ITU 1993.

[11] Hee-Jung Bae, Sung-Hwan Jung, "Image retrieval using texture based on DCT", *Proceedings of the International Conference on Information, Communications and Signal Processing*, pp. 1065 – 1068, 1997.

[12] Vidya R. Khapli, Anjali S. Bhalchandra, "Fast Image Retrieval Using VQ for Compressed and Uncompressed Images", *Computer Vision And Information Technology: Advances and Applications*, pp. 149-156, 2010.

[13] Daan He, "Efficient image retrieval in DCT domain by hypothesis testing", *16th International Conference on Image Processing (ICIP)*, pp. 225-228, Nov. 2009.

[14] P. Poursistani, H. Nezamabadi-pour, R. Askari Moghadam, M. Saeed, "Image indexing and retrieval in JPEG compressed domain based on vector quantization", *Mathematical and Computer Modelling*, 2011.

[15] Yuanjian Zhou, Liu Wien, "Image retrieval method based on color feature of diagonal sub-image in DCT domain", *2nd International Conference on Information Science and Engineering (ICISE)*, pp. 1249-1251, 2010.

[16] http://www.ijg.org/ and http://www.jpeg.org/jpeg/ (Retrieved on August 15th, 2012.)

[17] David M. Squire, Wolfgang Müller, Henning Müller, Thierry Pun, "Content-based query of image databases: inspirations from text retrieval", *Pattern Recognition Letters*, Vol 21, Issues 13-14, pp. 1193-1198, December 2000.

[18] Savvas A. Chatzichristofis, Yiannis S. Boutalis, "CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval", *Computer Vision Systems, Lecture Notes in Computer Science*, Vol. 5008/2008, pp. 312-322, 2008.

[19] Jing Huang, S. R. Kumar, M. Mitra, Wei-Jing Zhu, R. Zabih, "Image indexing using color correlograms", *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR 97)*, pp. 762-768, 1997.

[20] http://www.gnu.org/software/gift/ (Retrieved on August 15th, 2012.)

[21] http://www.semanticmetadata.net/wiki/doku.php?id=lire:lire (Retrieved on August 15th, 2012.)