An Overview and Evaluation of JPEG Compressed Domain Retrieval Techniques

David Edmundson and Gerald Schaefer
Department of Computer Science
Loughborough University
Loughborough, U.K
E-mail: d.edmundson@lboro.ac.uk, gerald.schaefer@ieee.org

***Abstract*—While content-based image retrieval (CBIR) has been an active research area over many years, most CBIR techniques operate in the pixel domain even though images are typically stored in compressed form. Consequently, image decoding is required prior to feature calculation leading to a computational overhead that is prohibitive in particular for the case of online retrieval. However, as has been shown by several authors, well performing image features can also be extracted directly from the compressed domain of the images.**

In this paper, we focus on JPEG compression since it represents the dominant image format, and provide an overview of JPEG compressed domain CBIR algorithms. We furthermore compare these algorithms on a common benchmark database, the MPEG-7 Common Colour dataset. We demonstrate conclusively that working in the compressed domain is consistently faster than in the pixel domain, typically requiring around 15% of the processing time. We also show that in several cases the retrieval performance is comparable to that of methods in the pixel domain whilst maintaining this speed gain.

***Keywords*—Content-based image retrieval, image compression, compressed domain image retrieval, JPEG.**

I. Introduction

Content-based image retrieval (CBIR) has been a well explored area of research [1], [2] for a number of years. The majority of existing CBIR methods operate in the pixel domain which in turn requires images to be fully decompressed before feature extraction can begin. In order to achieve faster feature extraction, several methods have been introduced that work directly on the compressed image data without the need of (full) decoding [3], thus allowing mid-stream content access [4].

JPEG [5] is the most commonly used image format, accounting for up to 95% of images on the web [6]. JPEG is a lossy image compression format that uses the discrete cosine transformation (DCT) on blocks of 8×8 pixels in order to discard high frequency and consequently visually less noticeable information.

Several CBIR methods have been introduced in the literature that allow calculation of effective image features directly in the compressed domain of JPEG, and in this paper we present an overview of the most common techniques [7], [8], [9], [6], [10], [11], [12]. We furthermore benchmark these algorithms on a common dataset, namely the MPEG-7 Common Colour database [13]. For comparison purposes, we also implemented several standard CBIR methods in the pixel domain, namely colour histograms [14] and three of the MPEG-7 visual descriptors [15]. Our experiments demonstrate conclusively that working in the compressed domain is consistently faster than in the pixel domain, typically requiring around 15% of the processing time. We also show that in several cases the retrieval performance is comparable to that of methods in the pixel domain whilst maintaining this speed gain.

II. JPEG Image Compression

JPEG [5] is currently the most popular image compression technique and has been adopted as an ISO standard for still picture coding. It is based on the discrete cosine transform (DCT), a derivative of the discrete Fourier transform. First, an (RGB) image is usually converted into the YCbCr space. The reason for this is that the human visual system is less sensitive to changes in the chrominance (Cb and Cr) than in the luminance (Y) channel. Consequently, the chrominance channels can be downsampled by a factor of 2 without sacrificing too much image quality, resulting in a full resolution Y and downsampled Cb an Cr components.

The image is then divided (each colour channel separately) into 8×8 pixel sub-blocks and DCT applied to each such block. The 2-d DCT for an 8×8 block $f_{xy}, x, y = 0 \dots 7$ is defined as

$$F_{uv} = \frac{C_u C_v}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} f_{xy} \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right) \quad (1)$$

with $C_u, C_v = 1/\sqrt{2}$ for $u, v = 0$, $C_u, C_v = 1$ otherwise.

DCT has energy compactification close to optimal for most images which means that most of the information is stored in a few, low-frequency, coefficients (due to the nature of images which tend to change slowly over image regions). Of the 64 coefficients, the one with zero frequency (i.e. F_{00}) is termed "DC coefficient" and the other 63 "AC coefficients". The DC term describes the mean of the image block, while the AC coefficients account for the higher frequencies. As the lower frequencies are more important for the image content, higher frequencies can be neglected. The way JPEG is doing this is by applying a quantisation step that crudely quantises higher frequencies while preserving lower frequencies more accurately.

In particular, 8×8 quantisation tables are used (one for luminance, one for chrominance channels) and the DCT

coefficients are simply divided by the factors in the tables. This is the lossy part of the compression as it is not uniquely reversible. The compression ratio (and consequently the image quality) is governed by a "q-factor" which essentially scales the quantisation tables.

After quantisation, the DC terms are differentially coded since they change slowly over the image. The AC coefficients are ordered in a zig-zag fashion for each block and are run-length coded. Finally, both parts are entropy (Huffman) coded to maximise the compression efficiency.

III. Overview of JPEG Compressed Domain CBIR

While most CBIR algorithms [1] operate in the pixel domain, various approaches have been introduced that perform compressed domain retrieval of JPEG images. All these techniques operate on DCT data; therefore the later stages of the compression algorithm (Huffman coding, run-length/differential coding and quantisation) still need to be reversed, however there is no need to perform the inverse DCT which is by far the most expensive operation when decoding JPEG images. In the following, we give an overview of the most popular algorithms that perform JPEG compressed domain CBIR.

A. Shneier and Abdel-Mottaleb [7]

One of the earliest attempts of JPEG image retrieval is that of Shneier and Abdel-Mottaleb [7] who divide an image into a set of sub-images, or windows. For each window, a key is constructed from the average of each DCT component in every block across that window. These windows are then paired with other windows on the same image. For each pair of windows (we used 4×4 windows per image), if the difference between the average DCT coefficients in each window exceeds a certain threshold α (in our experiments $\alpha = 20$), a 0 is assigned, otherwise a 1. This gives a feature vector of 64 bits for each pair of windows. Two images are compared using the Hamming distance between their feature vectors.

B. Lay and Guan [8]

Lay and Guan [8] investigated the use of low energy JPEG coefficients. In order to increase efficiency, only a select few coefficients were chosen. The authors tested various different combinations of components and concluded that the best results were achieved using the set $F_{00}, F_{01}, F_{10}, F_{11}$, i.e. the top four coefficients. The sum of these components is calculated, and a histogram generated. Histograms are then compared using the L_1 norm [14].

C. Schaefer [9]

Schaefer [9] proposed a JPEG retrieval method that uses both colour and texture descriptors extracted from only the DC terms. A colour histogram [14] is constructed from chromaticity DC data, while a texture descriptor is obtained by applying the LBP operator [16] on the DC data of the luminance (Y) channel and generating a histogram of LBP descriptors. Both colour and texture histograms are compared using the L_1 norm, and the resulting scores weighted and added. In our experiments we use 0.65 and 0.35 as colour and texture weights respectively.

D. Jiang, Armstrong and Feng [6]

Jiang, Armstrong and Feng [6] also exploited the fact that the average colour of an 8×8 block can be obtained directly from the DC component of the block. They also showed that it is possible to extract the average value for 4×4 blocks through simple operators involving the first 3 AC components (i.e., F_{01}, F_{10} and F_{11}). However, their experiments showed this to be less effective than just using the DC component. A colour histogram is hence built based only on DC data; in our experiments we used both RGB and YCbCr based histograms.

E. Feng and Jiang [10]

Feng and Jiang [10] looked at a way of describing an image using statistical analysis of the DCT blocks. By analysing the distribution of moments, their method essentially describes a rough estimation of texture information. For each block, the mean is obtained directly from the DC component, whereas the variance is calculated from all AC components. A histogram of the two features is built, where the mean values are quantised into 4 intervals, and the variance quantised into 7 non-uniform intervals.

F. Eom and Choe [11]

Eom and Choe [11] built an edge histogram detector that resembles the MPEG-7 non-homogeneous texture descriptor [15]. Image blocks are categorised as containing an edge based on the variance of all AC coefficients. For edge blocks, the difference between the first two AC coefficients (i.e., F_{01} and F_{10}) is used to determine if the edge is vertical, horizontal, 45°or 135°. An image is divided into 4×4 sub-images, and a histogram is built for each of the sub-images. The used distance measure is based on a combination of global, semi-global and local edge histograms.

G. Lu, Li and Burkhardt [12]

The algorithm by Lu, Li and Burkhardt [12] extracts colour, texture and basic edge information. Colour information for 4×4 pixel blocks is approximated using the first 3 AC coefficients (i.e., F_{01}, F_{10} and F_{11}) to build a colour histogram.

Two texture features are obtained. For the first one an energy histogram is extracted for each block. Three features are calculated using different bands of energy coefficients. The second texture feature is based on the edge information in the block and extracted using the sum of the coefficients which represent horizontal, vertical and diagonal energy shifts. The mean and standard deviation of these features across the images are then used, resulting in a 12 key descriptor for image texture. Each component of the descriptor is then normalised with respect to the entire dataset.

TABLE I

RESULTS OF OUR RETRIEVAL EXPERIMENTS. FOR EACH ALGORITHM, WE REPORT (FROM LEFT TO RIGHT): THE FEATURE VECTOR LENGTH (IN 32-BIT INTEGERS); THE RETRIEVAL PERFORMANCE (IN TERMS OF ANMRR) ON THE MPEG-7 COMMON COLOUR DATASET; THE (AVERAGE) NUMBER OF CPU INSTRUCTIONS REQUIRED TO EXTRACT THE FEATURES FROM AN IMAGE; AND THE (AVERAGE) NUMBER OF CPU INSTRUCTIONS REQUIRED TO COMPARE TWO FEATURE VECTORS.

	feature length [int]	retrieval performance [ANMRR]	feature extraction [CPU instructions]	feature comparison [CPU instructions]
Colour histogram [14] (RGB)	512	0.106	28593418	5091
MPEG-7 Scalable colour [15]	192	0.132	122932088	4328
MPEG-7 Colour layout [15]	92	0.276	44098479	1111
MPEG-7 Edge histogram [15]	80	0.352	37750697	3087
Shneier & Abdel-Mottaleb [7]	96	0.543	5902296	2420
Lay & Guan [8]	192	0.168	4293377	1497
Schaefer [9]	320	0.268	4305271	4229
Jiang *et al.* [6] (RGB)	512	0.116	3910487	5091
Feng and Jiang [10]	28	0.492	6491164	2466
Eom and Choe [11]	80	0.380	5932545	9428
Lu *et al.* [12]	584	0.235	4894941	7218

IV. IMAGE RETRIEVAL EXPERIMENTS

In order to compare the different algorithms, we use the MPEG-7 Common Color Dataset [13] which is a well established image set commonly used for CBIR benchmarking. The database consists of 5466 JPEG images (of different quality settings and sizes) taken from a range of different sources, and comes with 50 queries with a pre-defined ground truth.

As performance measure we use the suggested metric of average normalised modified retrieval rank (ANMRR) [13] which is defined as the average, over all queries, of

$$NMRR(q) = \frac{AVR(q) - 0.5(1 + NG(q))}{1.25K(q) - 0.5(1 + NG(q))}, \qquad (2)$$

where $NG(q)$ is the number of correct matches in the dataset for query image q and

$$AVR(q) = \sum_{k=1}^{NG(q)} \frac{Rank^*(k)}{NG(q)}, \qquad (3)$$

with

$$Rank^*(k) = \begin{cases} Rank(k) & \text{if } Rank(K) \leq K(q) \\ 1.25K(q) & \text{if } Rank(K) > K(q) \end{cases}. \qquad (4)$$

$Rank(k)$ denotes the rank a correct match was found, while $K(q) = min(4 * NG(q), 2 * GTM)$, where GTM is the maximum of $NG(q)$ over all query images, is a threshold defining a window of results, and 1.25 the penalty factor for any matches that fall outside of this window. ANMRR results are in $[0; 1]$ with 0 indicating that the whole ground truth was found as the highest ranking images, and 1 reporting that none of the ground truth images were found within K.

Retrieval results for all algorithms discussed in Section III are presented in Table I. We furthermore implemented several pixel-domain CBIR algorithms, namely colour histograms [14] and three MPEG-7 visual descriptors, scalable colour, colour layout and edge histograms [15] for comparison; results for these are also given in Table I.

From Table I, it is obvious that that there are large advantages to working in the compressed domain with feature extraction requiring typically only about 15% of the number of instructions of the faster methods in the uncompressed domain.

While all compressed domain methods are in general comparable in terms of feature extraction speed, not surprisingly those that employ simpler (or fewer) features are computationally more efficient. Jiang *et al.*'s direct content access is the fastest of these as it relies solely on the DC components.

Where the compressed domain technique is similar to a pixel-domain algorithm, such as Jiang *et al.*'s colour histograms and Eom and Choe's edge histograms, the retrieval performance is near identical to that of the original technique, whereas the feature extraction is reduced from 28 to 3.9 and 37 to 5.9 million instructions respectively which represents a significant saving.

As the name suggests, the Common Colour dataset is heavily tailored towards colour retrieval methods, and it can be seen that simple colour histograms give the best retrieval performance. Consequently, it is not surprising to find that the best performing methods are those that extract colour features rather than texture characteristics or a combination of the two. In fact, incorporating texture aspects may actually lead to reduced retrieval performance on this dataset.

The best performing compressed domain method is Jiang *et al*'s colour histograms giving both the best results as well as the shortest feature extraction time. This is followed by Lay and Guan's, Lu *et al*'s and Schaefer's algorithms all of which combine colour features and texture.

Jiang and Feng's statistical features and Eom and Choe's edge histograms give somewhat lower results, again due to their emphasis on texture based on AC coefficients, while Shneier and Abdel-Mottaleb's algorithm performs rather poorly.

V. CONCLUSIONS

In this paper, we have provided an overview of the most popular JPEG compressed domain image retrieval algorithms. Furthermore, we have presented a comparative analysis of these methods on a common benchmark database. We have demonstrated that performing image retrieval directly in the compressed domain leads to a significant decrease in terms of computational complexity with compressed domain methods requiring typically less than 15% of the time for feature

extraction compared to common pixel-domain algorithms. On the other hand, retrieval performance has been shown to remain comparable. On the MPEG-7 Common Colour dataset, colour histogram based image retrieval where the histograms are calculated directly based on the DC data in the JPEG stream was shown to give the best retrieval results.

REFERENCES

[1] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1249–1380, 2000.

[2] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, pp. 1–60, 2008.

[3] F. Idris and S. Panchanathan, "Storage and retrieval of compressed images," *IEEE Trans. Consumer Electronics*, vol. 41, no. 3, pp. 937–941, 1995.

[4] R. Picard, "Content access for image/video coding: The fourth criterion," MIT Media Lab, Tech. Rep. 195, 1994.

[5] G. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, pp. 30–44, 1991.

[6] J. Jiang, A. Armstrong, and G.-C. Feng, "Direct content access and extraction from JPEG compressed images." *Pattern Recognition*, vol. 35, no. 11, pp. 2511–2519, 2002.

[7] M. Shneier and M. Abdel-Mottaleb, "Exploiting the JPEG compression scheme for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, pp. 849–853, 1996.

[8] J. A. Lay and L. Guan, "Image retrieval based on energy histograms of the low frequency dct coefficients," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, 1999, pp. 3009–3012.

[9] G. Schaefer, "JPEG image retrieval by simple operators," in *2nd International Workshop on Content Based Multimedia and Indexing*, 2001, pp. 207–214.

[10] G. Feng and J. Jiang, "JPEG compressed image retrieval via statistical features," *Pattern Recognition*, vol. 36, no. 4, pp. 977 – 985, 2003.

[11] M. Eom and Y. Choe, "Fast extraction of edge histogram in DCT domain based on MPEG7," in *International Conference on Enformatika, Systems Sciences and Engineering*, 2005.

[12] Z. Lu, S. Li, and H. 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, 2006.

[13] Moving Picture Experts Group, "Description of core experiments for MPEG-7 color/texture descriptors," Tech. Rep. ISO/IEC JTC1/SC29/WG11/ N2929, 1999.

[14] M. Swain and D. Ballard, "Color indexing," *Int. Journal of Computer Vision*, vol. 7, pp. 11–32, 1991.

[15] T. Sikora, "The MPEG-7 visual standard for content description - an overview," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 696–702, 2001.

[16] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study for texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, pp. 51–59, 1996.