Exploring Content-Based Image Indexing Techniques In the Compressed Domain

A. R. McIntyre and M. I. Heywood
Faculty of Computer Science, Dalhousie University, Halifax NS, Canada
{ armcnty@cs.dal.ca | mheywood@cs.dal.ca }

Abstract

A method is proposed for performing clustering and offline indexing on the signal representations of compressed JPEG images. The current system clusters on Discrete Cosine Transform (DCT) blocks of JPEG images using the Potential Function clustering algorithm, storing indices of varying length for a posteriori comparison and processing of query images. Results presented indicate the appropriateness of using clusters derived from JPEG DCT blocks for content-based indexing. In particular, we are able to provide image summaries based on index features defined from a reference texture database, which significantly speed the search process.

Keywords: *Content-based indexing; image clustering; compressed domain.*

1. INTRODUCTION

1.1. Motivation

Pixel-based graphic formats comprise the vast majority of images on the World Wide Web (WWW) today, in fact the JPEG format for image compression alone accounts for more than 95% of all images on the Web [1]. Although the Web and a significant portion of Multimedia rely on JPEG, there exist few robust, unsupervised techniques for deriving index information based on actual image content [2]. By unsupervised it is implied that it is not possible to provide *a priori* labels for image content. A label typically restricts the set of queries that can be made against the image.

The main problem with indexing images stems from the inherent difficulty involved in automatically attaching meaning to the content of an image. On the other hand, even relatively straightforward frequency based methods are capable of producing index keys that perform relatively well on text documents [5].

Very few image indexing and query by example utilities exist on the Web today, and none has emerged as an efficient solution that is robust. That is to say, it may be easily applied to a variety of pixel-based media and operate primarily in the compressed domain. Such a tool would add significant value and popularity to search engines and other applications which are, to date, essentially blind to image data.

1.2 Compressed domain background

Given the widespread popularity of the JPEG image format on the Web, the method here makes use of properties of the JPEG compressed domain coefficients for improved performance. Operating directly on Discrete Cosine Transform (DCT) coefficients is hardly a new concept. Sin et al. [3] have verified that JPEG based DCT information is capable of acting as features for indexing texture databases, but have not looked at the appropriateness in a wider application context. Shneier and Abdel-Mottaleb [4] describe a general system for content-based querying of images based on the JPEG DCT data, but base their keys on windowed versions of the original image and average DCT block content. In this work we base our approach on DCT blocks derived from the off-line clustering of a texture database. The ensuing features are then used to compute a weighted histogram that summarizes image content. It is this information that is utilized during the query process.

In the next section, the basics of the JPEG algorithm are reviewed and its role in the system is discussed; section 3 summarizes the three main components of the indexing system; section 4 details the algorithms used; section 5 reviews the results collected to date and section 6 concludes the work, making comments on results and recommendations on future work.

2. JPEG COMPRESSION BASICS

2.1 Algorithm summary

The JPEG algorithm for image compression can be briefly summarized as follows [6]:

1. The original pixel-based image is tiled into 8×8 blocks, B_i, of 64 pixels each, with each pixel value

shifted from unsigned integer in the range $[0, 2^p - 1]$ to signed integers in the range $[-2^{p-1}, 2^{p-1} - 1]$

2. Each pixel block (B_i) is processed through the 2D Discrete Cosine Transform function (producing DCT Blocks (B_i^*), each with one AC coefficient (corresponding to the average intensity of the whole block) and 63 DC coefficients):

$$F(x,y) = \frac{1}{4} C(u)C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} C(u)C(v) f(x,y)$$
$$\left[\cos\left(\frac{(2x+1)u\pi}{16} \right) \cos\left(\frac{(2y+1)v\pi}{16} \right) \right]$$

   Where: *u, v* vary over column and row directions; $C(u), C(v) = 1/\sqrt{2}$ for *u, v* = 0; or 1 otherwise.

3. Uniform quantization related to 64 element quantization table entries:

$$F^Q(u,v) = \frac{F(u,v)}{Q(u,v)}$$

4. Zigzag ordering of DCT and quantized block coefficients.

5. Entropy (Huffman or arithmetic) encoding.

2.2 DCT characteristics

Following application of the DCT operation, each block from the input image produces a set of 64 signal amplitudes (in terms of 64 basis signals), which make up the signal's 2D 'frequency spectrum'. Since the DCT transform is a one-to-one mapping, the overall size of the DCT result obtained is equal in size and dimension to the input image, but can be viewed as a block by block signal representation of the original image, with a number of interesting properties [6]:

- The first coefficient of a given block (DC) represents the average pixel intensity for that block;

- Low frequency contributions are represented in the upper left section of the DCT coefficients;

- High frequency contributions appear later in the zigzag ordering (lower right) of the DCT coefficients;

- Typical 8 × 8 blocks do not vary greatly over such small distances, therefore the higher frequency contributions (amplitudes appearing later in the coefficient sets) tend to be negligible and are particularly well suited for compression.

2.3 Partial decoding process

The system presented makes use of direct access to the JPEG compressed domain. This description requires some clarification: the main (and most computationally expensive) compression technique in JPEG is the DCT function, which transforms the image blocks into what we consider the compressed domain of an image. Operations in the fully compressed image domain are clearly infeasible since the fully compressed JPEG image is a collection of quantized, reordered and Huffman coded DCT blocks.

In the context of this work we are interested in preserving certain properties and block orderings of the original image. The following assumptions are therefore made when making reference to the 'compressed domain':

- Huffman (entropy) decoding of the image blocks has been completed;

- Image blocks have been de-zigzagged and dequantized to their original format.

The objective of the system proposed is therefore to utilize the DCT block information to extract generic indexing keys suitable for content-based queries of JPEG images without recourse to features in the original, 'uncompressed' image.

3. SYSTEM DESCRIPTION

3.1 Overview

As mentioned in sections one and two, the general objective of this work is to test the suitability of basing a content-based image query system on the DCT blocks produced as part of the JPEG compression algorithm. The rationale being that if it is possible to form generic indices describing the content of images using a key of 'n' features, then rather than requiring the decompression of images, the extraction of the necessary features and then the summarization of image content, we merely measure the distance between query key and candidate image index keys. Such index keys could be incorporated into the JPEG image header at compression time, the most expensive part being to calculate the Euclidean norms over n features between the query key and the image database / images within a document. The proposed system performs this general process in three stages.

The most difficult aspect of such a system is identifying the set of '*n*' features comprising the keys, which is stage one. These remain the same irrespective of the original image (query), but are required to provide sufficient between-class discrimination – where such a concept is very much in the eye of the beholder.

Given the generic nature of this work (no *a priori* information regarding the query domain is available to guide the selection of features), the approach taken here is to determine '*n*' features from an independent 'texture' database. Moreover, the texture database is considered as a flat file (section 3.2) as opposed to independent images. Thus each DCT block extracted from the texture database is unique. The objective being to cluster the 'space' of textures as opposed to the frequency of appearance of specific textures. The top '*n*' clusters taken are across this space of DCT blocks representing the '*n*' features of each index key and are then used to summarize the content of the candidate images.

Having defined the contents for the generic index key, stage two calculates a weighted histogram for each image in the system. This summarizes the content of each image in terms of the index key features (section 3.3).

Stage three is the query process and merely computes the Euclidean distance between query key and candidate image index keys (section 3.4).

3.2 Reference key construction

Within the context of this work, content-based querying is treated as a requirement to produce a set of 'generic' features, applicable to summarizing the content of a broad range of image topics. To do so, a texture based approach to feature extraction is taken for constructing the index key features (expansion to include other features - e.g. color – is possible but not considered in this work). The work reported here uses 112 images in the Brodatz texture collection, reformatted to grayscale and with 100 × 100 pixel subimages taken.

To begin, texture images are decoded to the level of raw DCT blocks (see section 2.3). Each DCT block is appended to a data structure which holds a complete 'library' of DCT blocks from all the images in the Brodatz texture collection. The library is then scanned for exact DCT duplicates, which are eliminated as a preparation for the clustering phase. This is computationally the most costly process, on the order of $\vartheta(n^2)$, but is a one time, off-line process.

Once the unique set of DCT blocks has been constructed, a clustering algorithm is applied. For this preliminary work the widely studied, unsupervised clustering algorithm known as the Potential Function method [7] is used. The method provides *cluster centers* and therefore partitioning of data points without *a priori* knowledge of the number of clusters required; other unsupervised clustering methods may, however, be equally suitable for the task and are certainly worthy of further study.

One approach to conceptualizing the role of the clustering algorithm is to imagine the DCT blocks as a set of unique points in 64-dimensional space. Clustering, then, associates points which are naturally 'close' in proximity based on a set of four function variables (which are defined *a priori*), and ignores the outliers. Clustering thus identifies a subset of the original blocks which are most suitable representatives for the entire collection. This set is referred to as the *cluster centers*.

Following the identification of cluster centers, a subset consisting of the centers of the largest clusters is chosen as the features of the 'reference key'. The number of clusters, therefore, determines the size of the index 'keys' and has implications for the quality of matching images (note that these features are not orthogonal). The results section of this work presents results using keys with 64 features.

3.3 Assembling an image database

Having constructed the index key, we evaluate performance on a suitable image database. In this case, grayscaled versions of the images available from University of California at Berkley's Digital Library Project are employed [8]. There are nearly 28,000 images in the database; each having dimensions 192 × 128 pixels each. Of the total set, a cross-section of about 640 of the images was used in the initial experiments reported here. These represent several types of image *categories* including fungi, plants, flora, construction, aerial, mountain, and leaf images.

To assemble an index for a set of images, each image in the set is treated separately (in contrast to the processing of the texture set, mentioned above). That is, for any image, the set of DCT blocks are extracted without removing duplicate blocks, and in each case the Euclidean distance between it (in 64-dimensional space) and each feature from the index is calculated. The closest matching feature from the index key – the feature with the minimum nearest neighbor distance δ, $\delta = \min(d_i)$ where *i* indexes features of the key; and *d* is the Euclidean norm – is then recorded using an exponentially weighted histogram function:

$$C_n^1 = C_n + \exp\left(-\beta\|\delta\|\right)$$

where β is the *a priori* radius (section 4.1).

In summary, each image is represented by a set of features (the index key) and may be imagined as an exponentially weighted (clustered) texture histogram of the image. The set of all histograms is compiled and stored as an index to the database of images.

3.4 Querying the image database

To query the image index, it is first necessary to convert the query image (presented by the end-user) into the index key summary using exactly the same process as used to summarize the original image database (section 3.3). Querying the contents of the image database now takes the form of calculating the Euclidean distance between query index key ('*n*' features) and those in the image database.

Naturally, the best 'matches' correspond to the histograms that are nearest in the Euclidean sense. These images are returned to the user as candidates for a match, along with their respective distance from the query image. Though there may be few (or no) suitable 'matches' in the database for a particular query image, the distance metric returned can convey this concept, indicating that although the top *k* images returned are the best matches, they are not necessarily good matches.

It is anticipated that as the user ranks returned images it will be possible to weight the features of the index key to provide a mechanism for supervising the content query process. This, however, is outside the scope of the current study.

4. ALGORITHM DETAILS

4.1 The Potential Function

As mentioned previously, the initial step in this system is to cluster a series of basic, natural textures to form a reference key. The clustering process identifies natural relationships between proximities of DCT blocks in 64 dimensional space, allowing a set of representative texture features to be derived in an unsupervised learning process. The clustering algorithm chosen for this initial work is the Potential Function method, which allows the system to proceed without *a priori* knowledge of the number of required cluster centers.

The Potential Function is a four step iterative process, which proceeds as follows [7]:

1. Identify each point's candidate potential with respect to all other points in the training set (section 3.2) using a suitable distance metric.

2. Select the point with the highest total potential.

3. Subtract the highest potential (as determined in step two) from all other points.

4. Repeat steps two and three until a terminating condition is realized.

The distance metric identified in step one is referred to as the 'Potential Function':

$$P_t(x(j)) = \sum_{i=1}^{K} \exp\left(-\alpha \left\| x(i) - x(j) \right\|^2\right)$$

where *x(j)* is the *jth* point in the DCT blocks of the collection of texture images; *i* is the iteration of the Potential Function; *K* is the total number of DCT blocks over the entire (texture) set and α is the cluster radius constant.

As such, DCT blocks *x(i)* with the most similarity to the current candidate block, *x(j)*, contribute most to the corresponding potential $P_t(x(j))$. Blocks with the most (or very near) neighbors will be assigned the greatest potential (as required in step two).

Step three removes the influence of the 'winning' point (the point with the highest potential) from all others within the same cluster. Thus, having found a winner, each cluster member point's potential is reduced by an amount proportional to its distance from the current winning potential $P_t(x^*(j))$, or for all $i \in \{1...K\}$ as follows:

$$P_{t+1}(x(i)) = P_t(x(i)) - P_t(x^*(j)) \exp\left(-\beta \left\| x(i) - x^*(j) \right\|^2\right)$$

where $P_{t+1}(x(i))$ is the updated potential at iteration *t+1* and β (< α) is the radius associated with the Potential decay process. This process continues until an end condition is reached, which is determined by the ratio of the potential at the current time step $P_t(x^*(j))$, to the initial potential, $P_0(x^*(j))$ as follows:

- IF $P_t(x^*(i)) > \gamma_{upper}\left(P_0(x^*(j))\right)$
  THEN create a new cluster;

- ELSE IF $P_t(x^*(i)) < \gamma_{lower}\left(P_0(x^*(j))\right)$
  THEN end;

- ELSE ignore the point (it does not represent a significant cluster).

where: $\gamma_{lower} = 0.15$ and $\gamma_{upper} = 0.5$ for this work.

In these experiments, the assignment of DCT blocks, *x(i)* to clusters is determined by the '*j*' which resulted in the greatest decrease of potential during the decay steps.

5. RESULTS

Initial results indicate the appropriateness of using clusters derived from JPEG DCT blocks for content-based indexing. In particular, test runs have shown excellent results when performing searches for certain categories of images, and promising results for other categories.

Since test results for such systems are difficult to quantify objectively, a series of sample queries and the top three nearest results returned by the system are provided in figures 1-4.

To summarize, initial experiments have focused on querying an image database of seven image categories (Table 1, column 1), each category containing roughly 100 related images. The database reported here was constructed with the following Potential Function method radii constants for texture clustering (section 4.1): $\alpha = 0.2$, $\beta = 0.00005$. The database assembly and query processes (sections 3.3 and 3.4) used the same β as during texture clustering. Keys for these results are composed of 64 texture features.

The query set was composed of the same seven categories, each containing a set of 10 new, randomly selected images (which do not appear in the database).

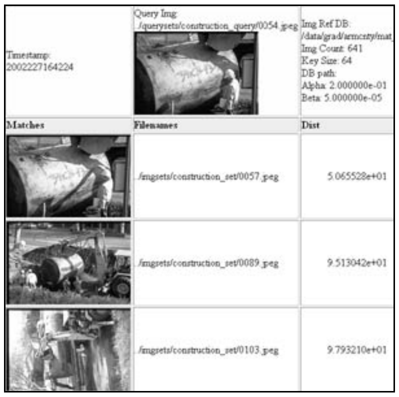
Table 1. Summary of query results

Query Category	Top 3 Cat Match	Top 3 Class Match
Construction	7/10	N/A
Flora	6/10	10/10
Fungi	8/10	10/10
Leaf	3/10	9/10
Mountain	8/10	9/10
Aerial	8/10	9/10
Plant	6/10	7/10
MEAN	~ 66%	90%

For reporting purposes, a total of three image *classes* were formed which represent tendencies of some of the seven categories to overlap. Class A consists of the union of the mountain and aerial image sets; class B consists of the union of the leaf and plant image sets and class C consists of fungi and plant image sets.

Results compiled for this experiment are summarized in Table 1. The 'Top 3 Cat Match' column represents the ratio of number of times a same category image appeared in the top three query results to the number of queries performed. The 'Top 3 Class Match' column indicates the ratio of the number of times a same class image appeared in the top three query results to the number of queries performed on each image category.

Additional experiments indicate that the query results obtained are moderately sensitive to the β value chosen for database assembly and the query process.



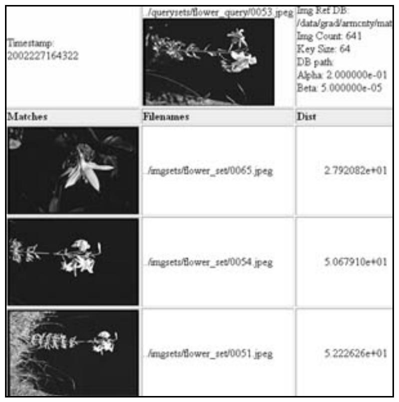
Timestamp:
2002227164224

Query Img:
../querysets/construction_query/0054.jpeg

Img Ref DB:
/data/grad/armcnty/mat
Img Count: 641
Key Size: 64
DB path:
Alpha: 2.000000e-01
Beta: 5.000000e-05

Matches	Filenames	Dist
	../imgsets/construction_set/0057.jpeg	5.065528e+01
	../imgsets/construction_set/0089.jpeg	9.513042e+01
	../imgsets/construction_set/0103.jpeg	9.793210e+01

Fig. 1. Sample construction query.



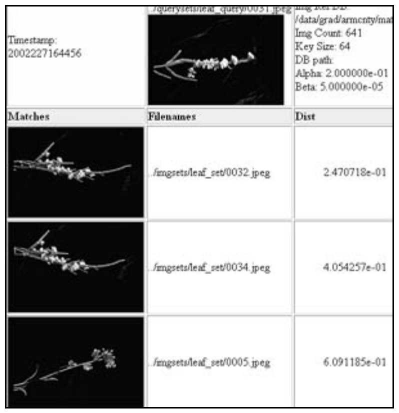
Timestamp:
2002227164322

../querysets/flower_query/0053.jpeg

Img Ref DB:
/data/grad/armcnty/mat
Img Count: 641
Key Size: 64
DB path:
Alpha: 2.000000e-01
Beta: 5.000000e-05

Matches	Filenames	Dist
	../imgsets/flower_set/0065.jpeg	2.792082e+01
	../imgsets/flower_set/0054.jpeg	5.067910e+01
	../imgsets/flower_set/0051.jpeg	5.222626e+01

Fig. 2. Sample flora query.



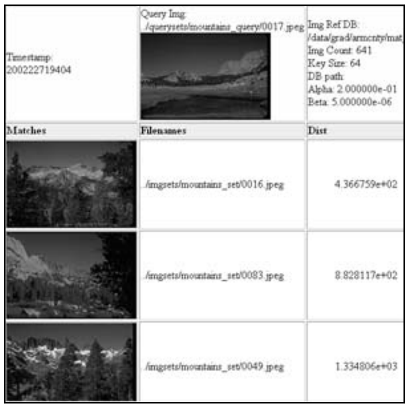
Timestamp:
2002227164456

../querysets/leaf_query/0031.jpeg

Img Ref DB:
/data/grad/armcnty/mat
Img Count: 641
Key Size: 64
DB path:
Alpha: 2.000000e-01
Beta: 5.000000e-05

Matches	Filenames	Dist
	../imgsets/leaf_set/0032.jpeg	2.470718e-01
	../imgsets/leaf_set/0034.jpeg	4.054257e-01
	../imgsets/leaf_set/0005.jpeg	6.091185e-01

Fig. 3. Sample leaf query.



Timestamp: 200222719404	Query Img: ../querysets/mountains_query/0017.jpeg	Img Ref DB: /data/grad/armcnty/mat Img Count: 641 Key Size: 64 DB path: Alpha: 2.000000e-01 Beta: 5.000000e-06
Matches	**Filenames**	**Dist**
	../imgsets/mountains_set/0016.jpeg	4.366759e+02
	../imgsets/mountains_set/0083.jpeg	8.828117e+02
	../imgsets/mountains_set/0049.jpeg	1.334806e+03

Fig. 4. Sample mountain query.

6. CONCLUSIONS

A method for offline indexing based on the clustering of DCT blocks in the compressed domain has been proposed. Though results are preliminary, they appear to indicate appropriateness of this approach. A test system is presented which is able to find a category match within the first three candidate results 66% of the time, on average.

Arguably, images in some categories are reasonable search results for queries on images in others. Though it is necessary to be more subjective, it is important to observe close relationships between certain categories of test images (we term related categories *classes*), which imply coarsely successful query results on 90% of all test queries, on average.

We anticipate that further extensions to this research will continue to improve the quality of the query results in many respects. Some issues that are to be addressed in future work include:

- Separation of DCT components into X and Y histogram contributions for spatial feature resolution;
- Removal the DC component from DCT blocks to remove reliance on pixel intensity in images;
- A methodology for enabling users to 'focus' the query process on specific objects. This will enable the system to narrow searches to more specific, interesting items in terms of the query's exact subject;
- Derivation of moment method features from blocks identified during the clustering process for expressing geometric properties of the object.

Acknowledgements

Dr. Heywood and Mr. McIntyre gratefully acknowledge the support of an NSERC Research Grant.

Mr. McIntyre would like to thank L. J. Breckenridge and A. J. L. Webb for helpful feedback.

References

[1] J. Jiang, A. Armstrong and G. C. Feng, "Direct Content Access and Extraction from JPEG Compressed Images," *Pattern Recognition*, pp. 1-9, November 2001.

[2] Ccharles Frankel, Michael J. Swain, and Vassilis Athitsos, "WebSeer: An Image Search Engine for the World Wide Web", *Technical Report 96-14*, Department of Computer Science, University of Chicago, 1996.

[3] D. G. Sin, H. K. Kin, and R. H. Park , "Fast texture description and retrieval of DCT-based compressed images," *Electronics Letters*, vol. 37, no. 1, pp. 18-19, 2001.

[4] M. Shneier, M. Abdel-Mottaleb, "Exploiting the JPEG Compression Scheme for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 849-853, 1996.

[5] S. Brin, and L. Page, "The Anatomy of a Large Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107-117, 1998.

[6] G. K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, vol. 34, pp. 31-44, April 1991.

[7] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent and Fuzzy Systems,* vol. 2, pp. 267-278, 1994.

[8] R. Wilensky, "CalPhotos," *The Digital Library Project, University of California, Berkley*. Accessed February 14, 2002. http://elib.cs.berkeley.edu/photos/index.shtml