Differential Compression and Optimal Caching Methods for Content-Based Image Search Systems

Di Zhong[a], Shih-Fu Chang[a], John R. Smith[b]

[a]Department of Electrical Engineering, Columbia University, NY, USA

[b]IBM T.J. Watson Research Center, NY, USA

ABSTRACT

Compression and caching are two important issues for a large online image server. In this paper, we propose a new approach to compression by exploring image similarity in large image archives. An adaptive vector quantization (VQ) approach using content categorizations, including both the semantic level and the feature level, is developed to provide a differential compression scheme. We show that this scheme is able to support flexible and optimal caching strategies. The experimental results demonstrate that the proposed technique can improve the compression rate by about 20% compared to JPEG compression, and can improve the retrieval response by 5% to 20% percent under different typical access scenarios.

Keywords: image compression, network caching, content-based image search, vector quantization, multimedia database, content categorization

1. INTRODUCTION

Large-scale media archives are increasingly popular as more digital media content is created and deployed on-line. Such archives may include different types of media such as images, video, audio, graphics, and text documents. An important issue in designing such archives is compression and storage of the data. Although the cost for storage is rapidly dropping, compression is still a significant issue when the archive size is large. In addition, choice of compression techniques has great impact on other aspects of the system. For example, by exploring the characteristics of the compression algorithms (e.g., intra-frame coding vs. inter-frame coding, scalable coding vs. non-scalable coding), optimal storage and delivery schemes can be achieved.

In this paper, we explore a new dimension of interaction between compression and content indexing. Today, most image servers include techniques for feature extraction, indexing, and categorization. One popular approach is to use similarity matching to support functions like “query by example” or “find me more images similar to this”. Image similarity has been explored to cluster images into hierarchical structures for browsing or subject categorization.

We propose a new approach to compression by exploring image similarity in large image archives. Existing systems encode each image in the database independently. Image similarity is only exploited between image frames in the same video sequence, but not among others. We argue that for large image archives, visual similarity among images is likely to exist. If we can effectively find such similarity, coding redundancy can be minimized. We called this new approach “*similarity-based image compression*”. Conceptually, it uses the same principle as video coding, where similar blocks in subsequent image frames are identified and predictive coding is used.

In a large image server (*e.g.*, Altavista media search[2], WebSEEk[1]), images are usually grouped to various categories based on some semantic-level or feature-level criteria. For example, at the semantic level, images may be categorized into classes of “nature/flowers/rose” or “art/painting/vangogh”. At the feature level, images can be classified to clusters based on visual features (*e.g.*, color, texture). These groupings usually are based on hierarchical structures and may be used to support interactive browsing by users. These hierarchical structures also provide a suitable framework for similarity-based compression. Images in the same class may have similar visual content - particularly for classes obtained by visual feature

clustering. Similarity-based compression aims at removing the "visual" redundancy among images in the same class and thus achieving high coding efficiency.

Note that similarity-based compression is different from scalable coding. Scalable coding is used to provide a multi-resolution representation for a single image. Similarity-based compression is used to encode a group of similar images. Actually, these two approaches can be combined to achieve both functionalities. One analogy is the combination of inter-frame predictive coding and spatial scalable coding in MPEG-2[3]. The inter-frame coding removes the redundancy between subsequent frames while the spatial scalable coding provides multi-resolution coding for each frame. In [7], an MPEG based method is proposed to find similar image blocks across the whole collection and encode them with indices. It is shown to be useful for large image collections where a large amount of similar image blocks exist.

Specifically, we develop efficient vector quantization (VQ) compression algorithms that take advantage of the categorization structure in the image archive. The optimal VQ codebook is developed for each image class at in the hierarchy. We also develop a hierarchical structure to organize multiple VQ codebooks associated with different levels of the hierarchy. When a VQ codebook is used to encode an image class, each image in that class is represented by a codeword in the codebook plus the residual difference between the codeword and the image. The residual difference is further compressed by lossy compression like JPEG. Note that an image may be encoded multiple times with respect to different VQ codebooks if needed. For example, an image may belong to several semantic classes (e.g., "architecture" and "building") and/or visual clusters.

Another focus of this paper is to develop efficient caching schemes for prefectching VQ codebooks by exploring the context of the user interaction. We propose to prefetch the VQ codebooks that will likely be used in subsequent user operations in order to reduce the retrieval delay when users actually request to retrieve images later. Prediction of user operations is based on the current state of user access (e.g., the current state in the subject hierarchy when the user is browsing) and likelihood distribution of different operations based on empirical data. Prefectching is feasible and can be done during frequent time windows between subsequent user operations. Note, however, as is the case for operation system, prefetching is most effective when the prefetched data is frequently referenced by the subsequent operations.

In this paper, we present innovative similarity-based image coding methods based on adaptive VQ and content categorization. We compare the new similarity coding with JPEG in terms of compression efficiency. We also propose optimal caching algorithms for such adaptive VQ techniques to improve the access speed during interactive image retrieval. The experimental results demonstrate that compared to the JPEG compression, with the simple MSE-based LBG training the proposed VQ technique can improve the compression rate by about 20%, and can improve the retrieval response by 5% to 20% percent under different typical access scenarios.

This paper is organized as follows. Section 2 includes description of the client-server system architecture for image retrieval. It also includes assumptions about the system functionalities and user access scenarios. In Section 3, we discuss the adaptive VQ compression scheme using content categories, including semantic categorization and feature clustering. Section 4 describes multi-level structure of VQ codebooks based on hierarchical categorization. Caching strategies and performance evaluation are presented in Section 5. Section 6 presents the conclusion and future work.

2. SYSTEM ARCHITECTURE AND PROBLEM FORMULATION

In this section, we discuss the system architecture and scenarios of user access to the image archive. We then formulate the technical problems and define specific performance metrics.

Figure 1 shows the client-server architecture for a typical image search system. Users access the image server through a bandwidth limited network link (bandwidth W). The cache is the memory space that's used to store previously delivered data. It could be located at the user site or the intermediate proxy site. On the server side, we assume the image collection is associated with keyword annotations for individual images, visual feature indexes, one semantic classification tree, and one or more hierarchical feature cluster structures. Each semantic class can also be indexed by keywords associated with images in that given class.

To clearly define the problem, we also make the following assumptions.

Network link: We assume the user accesses the server through a network link with limited bandwidth, W (bits per second).

Cache: We assume that previously delivered data is stored in the cache, with a size of B (bits).

User interface: We assume users have limited display space on their screens. The maximum number of images simultaneously displayed on the interface is K. We also assume that for each access request, thumbnail images are shown first and full-size images are displayed only upon user's request.

Image representation: Images in the database may be represented with multiple resolutions. This paper focuses on the compression of images at one single resolution (i.e., the thumbnail level). As discussed earlier, coding of multi-resolution images is similar to issues encountered in scalable coding in MPEG-2 high profiles[3]. But we will not address this issue in this paper.

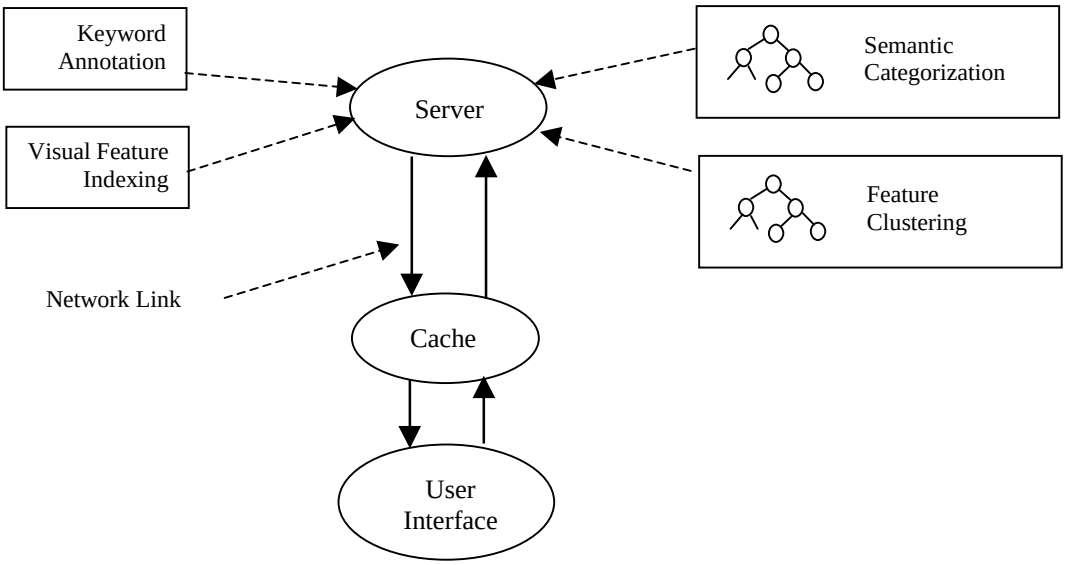


Figure 1. Client-server architecture for a typical image search system.

Access scenario: We assume the following three access scenarios.

1. *Browsing using the semantic tree*. Users may choose to navigate arbitrarily in the semantic subject tree. When the user selects a specific class at the terminal level, images of that class will be retrieved.
2. *Keyword search*. After the user issues a query with one or more keywords, the matched images will be retrieved. In some systems, subject classes with matched keywords are returned and then users may choose to browse content in specific classes. We do not include this option in the study. However, each semantic class is also indexed by representative keywords. Upon a keyword search, VQ codebooks of the matched classes can be prefetched to the cache. This process can be run in the background and is transparent to users.
3. *Find more similar images*. After seeing some retrieved images, the user may request to retrieve more images similar to one or more specific images displayed on the screen. Or users may select specific images at the very beginning of the session and request to find images similar to the specific ones. This is a popular search function called "query by example".

Based on the above assumptions, one technical objective is to develop efficient similarity-based compression algorithms to remove redundancy among similar images in the archive. Another objective is to develop efficient schemes for prefetching codebooks based on the user access scenarios. Here we make an important assumption that there are non-zero time intervals between user operations. For example, users usually spend some time in reviewing the subject categories before they choose the next operation. If users choose to do direct queries, there is always some query process delay, particularly when the query is based on visual features or image examples. It is during these non-zero time intervals we execute prefetching. In an ideal case when the cache size is very large, we may even be able to prefectch all codebooks at the beginning of the session.

We will evaluate the proposed approach in the following two aspects.

1. Compared to the existing compression techniques, how do the similarity-based coding techniques perform in terms of the compression ratio and image quality?

2. Compute the access delay for each stage of different access scenarios. We will show that the proposed compression and caching methods can greatly reduce the access delay when the access link is bandwidth limited.

3. ADAPTIVE VQ USING CONTENT CATEGORIES

The basic idea of the vector quantization[4] of images is shown in Figure 2. An input image is partitioned into non-overlapping blocks. Each block of pixels is encoded as the index of its nearest codeword in the VQ codebook, according to certain distortion measures. The decoder simply looks up the codebook using the indices, and outputs the corresponding codewords. The average distortion of the VQ coding scheme is

$$\overline{D} = E[d(F_i, F_i^{'})] \tag{1}$$

where F_i is the input block and $F_i^{'}$ is the decoded block. Thus the optimal codebook is the one that yields the least average distortion. The LBG clustering algorithm has been widely used to find an optimal codebook.

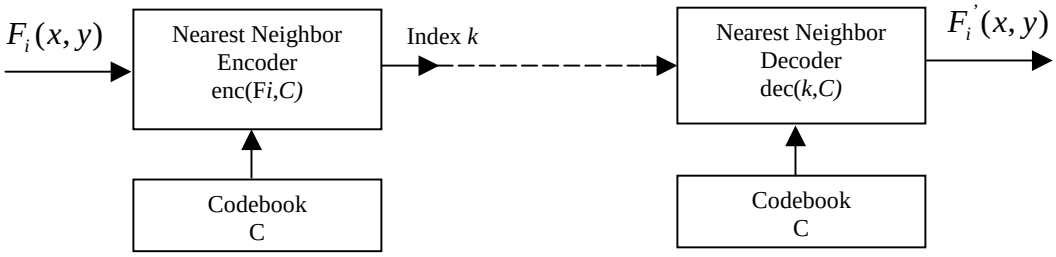


Figure 2. A typical vector quantization scheme

The reason why the VQ scheme can be used to compress images is that images are comprised of repeated patterns. Given a set of example patterns (i.e. a codebook), we can reduce the redundancy of such patterned structure by encoding it with pattern indices.

Usually, a codebook is derived from a set of training images, and thus is data-dependent. While the maximum encoding error of any image within the training set can be controlled, an outside image may be encoded with an error exceeding that maximum value. Such dependence on the training data has limited the usage of VQ in many areas. However, as we will discuss in the following sections, for a large online image database, where users only have remote access through a bandwidth-limited network link, an adaptive VQ based on content categories can provide efficient compression and caching performance.

In the following, we will discuss two different types of content categorization. They are used to support different browsing and retrieval functions in image databases.

3.1 Semantic Content Categorization

Semantic classification of images in a database can be done based on the text information (e.g., keywords) associated with image[1]. Such a classification structure allows for efficient browsing of content in the archive. Given this kind of structure, users often request to retrieve a subset of images from each class for preview. This is also true for another common access scenario - keyword query. For a keyword query, images that will be accessed are usually within one or a few categories associated with this keyword.

Based on the above category-based access pattern, a compression scheme can be designed at the server side to help clients browse and search the database more efficiently. Instead of looking at the compression of each image independently, we will explore the correlation among the images within each category, and compress them as a single entity.

As we mentioned before, VQ techniques are based on discovering and encoding of repetitive image patterns, and thus provide a good approach to exploring similarity among images in a category. In the traditional VQ compression, the goal is to minimize the average coding/decoding distortion (Eq. 1). Here our goal is to separate the common information (i.e., codebook) from unique features of each image (i.e., residual error) in the encoding process. During users' interaction with the system, codebooks are pre-fetched to clients during non-active periods. Non-active periods are defined as the time interval between subsequent user operations. It could be the time that users spend in reviewing categories between successive requests, or the query process time between a query is issued and the time processing of that query is complete. This inactive period basically is the time interval that the network link is idle and can be used to download the data such as codebooks.

The residual errors are not thrown away as in the traditional VQ compression. They are also encoded and will be transmitted to clients when users want to view images.

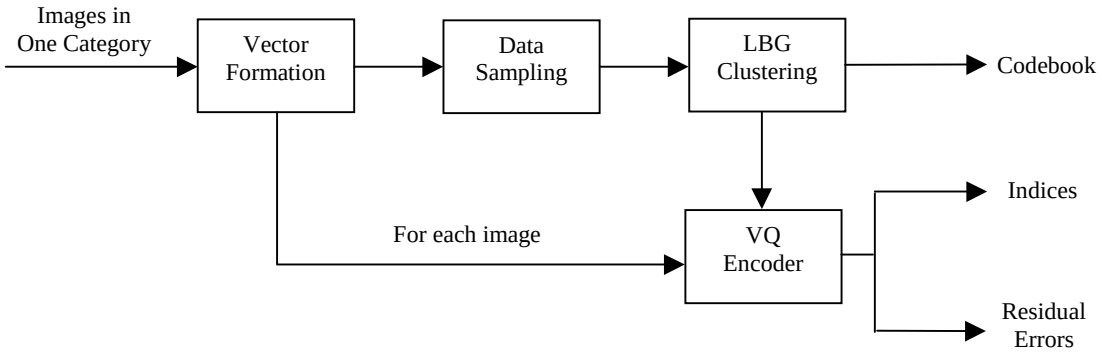


Figure 3. Content category based VQ compression

The VQ encoding process is shown in Figure 3. Among the three outputs, codebooks are expected to be transmitted before the indices and residual errors. The indices are usually smaller than the other two and can be compressed with run-length coding. The size of a codebook is critical. It depends on the access bandwidth of clients, *W*, and the average non-active period, dt . Assume B is the size of each codeword, the maximum number of codewords that can be downloaded is,

$$N = \frac{W * dt}{B} \tag{2}$$

Residual errors are expected to be coded smaller than conventional compression methods (e.g., JPEG). A natural approach is to encode the residual error using the JPEG compression with a flat quantization table (i.e., similar to the inter-frame compression in the MPEG compression). This is because an error image contains only high frequency signals. How to design a LBG that will minimize the size of compressed residual errors is an open issue to be studied. Here we demonstrate the idea with the common LBG algorithms based on mean square error (MSE) criterion.

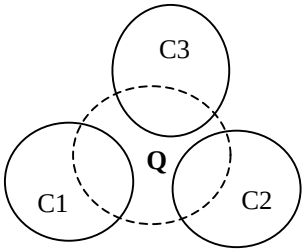
3.2 Visual Feature Based Clustering

In addition to the above text-based semantic access, feature based similarity query is another important method to find images in a large database. Query based on either example images or sketches allows users to retrieve images that are visually similar to the query. Color, texture and shape are common features that have been used to match the visual similarity. Similar to the semantic-level categorization, low-level features have been used to form clusters in image archives. Such clusters can be used to aid interactive browsing or automatic semantic classification.

To build the feature cluster, an extended k-means clustering algorithm is utilized. It ensures that the maximum intra-class distance (e.g. L2 distance) of generated clusters is smaller than a pre-defined threshold. This will create various number of classes based on the actual distribution of feature vectors, and most importantly, it increases the VQ compression efficiency by limiting the variance of images within each cluster.

Unlike the keyword searching, where we know exactly which categories a keyword matches, due to the nature of similarity matching, a visual query may return results from several neighboring categories (as shown in Figure 4). As we will discuss in Section 5, by building a hierarchical clustering structure as the index for visual feature search, we are able to know which

clusters contain search results before these results are retrieved. Thus it is feasible to pre-fetch codebooks before actual images are returned.



C1, C2 and C3 are image classes based on visual features.

Q is the visual query. The dash circle contains images that will be returned to users based on similarity matching.

Figure 4. A visual similarity search where return results belong to several neighboring classes

3.3 Observations and Possible Improvements

As the VQ codebook is computed under a mean square (or L1-distance) error criterion, it tends to capture smooth image blocks well, but represents edge blocks poorly. A typical example is shown in Figure 5. As we can see from the image of the residual errors, error values are larger around edge pixels. The reconstructed image (i.e., codewords) usually represents the smoothed values.

(a) Original image

(b) Reconstructed image from codewords

(c) Residual error (the background level is 128)

Figure 5. A typical example of the VQ based compression

Some advanced VQ approaches can be used to greatly reduce the residual errors. The classified VQ[5] is one of such approaches. It classifies image blocks into smooth and edge parts, and generates different optimal codebooks for them respectively.

Another important issue that needs to be studied in a practical system is the optimal size of each codebook and the optimal number of codebooks (i.e., the number of content categories). With larger size and number of codebooks, residual errors will be smaller. On the contrary, residual errors will be larger. The optimal decision may depend on the available bandwidth, the cache size as well as the nature of images in the database.

4. MULTI-LEVEL STRUCTURE FOR VQ CODEBOOKS

Considering the large amount of data in the database, multi-level semantic categorization and feature clustering provide an efficient abstraction hierarchy. Users can quickly get a sense of the images contained in the database with a look at a few top-level categories. In [1], a subject taxonomy for image and video is developed using a process of key-term mappings. A hierarchical feature clustering tree can be constructed by applying the aforementioned k-means algorithm recursively

following a top–down approach. That is to say, we can first create N clusters at the top level and then classify images within each cluster into sub-classes again.

In the previous section, we introduced the adaptive VQ for images within each content category. How to adapt it to the multi-level hierarchical categorization structure and to manage codebooks effectively for online image access will be addressed in this section. One straightforward approach is to create codebooks for all categories at the bottom of the categorization tree, and then use the tree only as the index to the codebooks. In other words, codebooks only exist in the terminal nodes of the tree. With such a design, codebooks are independent of each other. To show two images from two neighboring categories simultaneously requires two separate codebooks. This certainly does not provide a flexible method for efficient caching.

To further explore the content similarity at a higher level category, the LBG training process is applied to its lower level codebooks in the hierarchy to create a new codebook. This new higher level codebook thus contains more general codewords with respect to its lower level codebooks (Figure 6). The bottom-up process continues until the top-level codebook is generated.

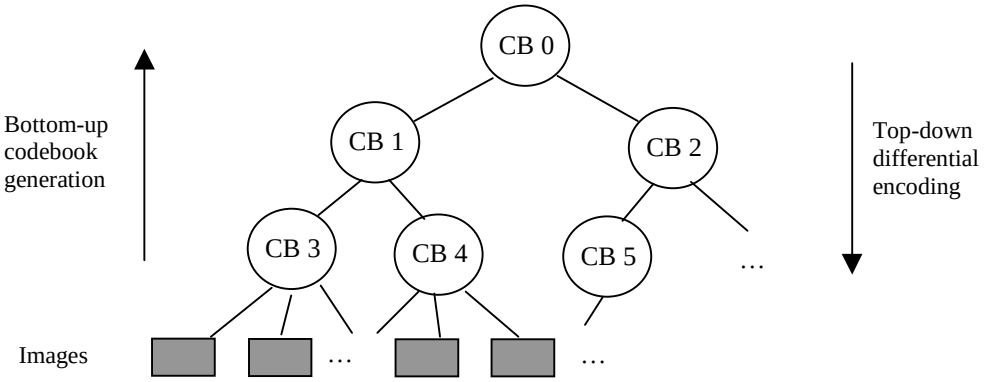


Figure 6. Muti-level structure of VQ codebooks

Because only the codebooks at the bottom level will be used to encode images, the above multi-level structure of codebooks can not improve the transmission efficiency of codebooks. It is desirable to have a coding scheme that links parent codebooks to their child codebooks so that codebooks at higher levels can be used to help download low level ones.

To fulfill this goal, we develop a top-down differential coding scheme for the codebook hierarchy. The top-level codebook is represented as a set of example blocks. Codebooks at the following levels are encoded using their parent codebooks, and thus each includes an index table and an image of residual errors. This process can be applied until the bottom level where images are VQ encoded. As we discussed before, the residual error of an image can be coded with JPEG compression. Residual errors of intermediate codebooks can be encoded with either lossless compression or lossy compression with limited distortion. For example, codewords (i.e., their associated residual errors) in a codebook can be arranged to a two dimensional array and treated as a normal image for compression. For this purpose, the Self-Organization Map (SOM) [6] provides a better solution than the LBG, as the output of the SOM is a 2-D map of codebooks where similar codewords are spatially close to each other.

Assume the top-level codebook is always cached at the client side, the above coding structure supports progressive transmission of codebooks when the database is accessed following a top-down pattern. This structure also provides some other benefits. In both semantic and feature classification, there are some classes containing only a few images. To create codebooks for these classes degrades the compression ratio. Within the tree structure, such codebooks can be easily replaced by their corresponding parent codebooks.

It is also possible to combine the semantic content categorization and the feature clustering. For example, in the semantic taxonomy, there are usually some bottom-level classes which contain a large number of images. This situation may cause large residual errors of VQ encoding. Feature-based clustering can be used here to classify the images into a few sub-classes with much more efficient VQ compression.

5. PERFORMANCE EVALUATION

For the performance evaluation, we built an image database with about 2700 images (in either GIF or JPEG format). They are a subset of images collected by the Internet spider of the WebSEEk system. The semantic taxonomy is also obtained from the WebSEEk system[1]. There are four categories at the first level, *animal, architecture, clothes and sports*. The depth of the semantic tree is about 3 to 5 levels. There are about 90 bottom-level categories, and thus in average there are 30 images in each bottom-level category.

Visual features used to match the image similarity is the typical 64-bin color histogram. The clustering process is done with a target number of 20 classes under each node. This creates a 3-level feature clustering tree with about 120 leaf nodes.

Under our assumption of the user access pattern of an image database, users may first browse or query the database and then examine a relatively large number of icons. The transmission of full-size images only occurs upon users' specific requests. Thus in this study, experiments are conducted on the icons of the images. The size of each icon is around 96x96.

In the following sections, we present our evaluation results based on several practical scenarios.

5.1 Compression Efficiency

In the adaptive VQ coding of icon images, we blocks 4x4 of pixels with RGB values (i.e., a 48 dimensional vector). The size of the codebook is 512. The LBG training algorithm with MSE criterion is used to generate codebooks. In the encoding process, residual error images are compressed using JPEG with a flat quantization table. The quantization step value is 24. Table 1 provides a comparison of the compression efficiency of the proposed VQ method with JPEG compression and the global VQ. It is conducted on the 1329 GIF images in the database. Semantic adaptive VQ uses techniques described in Section 3.1. Feature adaptive VQ uses techniques in Section 3.2. Global VQ uses a single VQ codebook to encode images in the whole collection.

Table 1. Comparison of compression efficiency

Compression Scheme	Average Compression Ratio	Average PSNR (luminance)
JPEG (default)	0.42	32.2 dB
Global VQ	0.42	31.9 dB
Semantic Content Adaptive VQ	0.34	32.4 dB
Feature Content Adaptive VQ	0.32	32.7 dB

Here JPEG compression uses the default quantization table. When computing the size of VQ compressed images, codebooks are not included based on the assumption that they will be transmitted separately. In 5.2, we will present a summary of the sizes of codebooks. Global VQ gives the similar compression and PSNR results as the JPEG, as the residual errors are compressed in JPEG. Semantic and feature content adaptive VQ achieve 18% and 23% more compression ratios respectively and with similar PSNR's. This validates our belief that in large image archives, a more efficient compression scheme can be developed by exploring the content similarity, especially the visual similarity. As we discussed before, more advanced VQ techniques, such as classified VQ, are expected to bring better results.

5.2 Download and Cache All Codebooks

When the client machine has enough cache and can pre-fetch all the codebooks from the database, users will be able to experience the maximum speed-up provided by our compression scheme. The size of the top-level codebook is 512*48=24576 bytes or 24 kbytes. The rest codebooks are encoded differentially. Here we use a simple JPEG (flat quantization table with the step size 8) to compression the residual errors. The average size of each subsequent codebook, *B*, is 5149 bytes or 5.0 kbytes. The total size of the 121 codebooks of the semantic categorization tree is 24+5.0*120=624 kbytes (which is about 9% percent of the total size of compressed icons).

Thus, to hold the whole codebook structure, the client machine needs to have at least 636 kbytes cache size. Downloading these data with reasonable speed probability is feasible for network links of high bandwidth such as ISDN or Ethernet.

5.3 Semantic Category Browsing

We assume there is a 5 second average interval between successive operations, when users are browsing the semantic categories. Without losing the generality, we also assume the top-level codebook is already cached at the client side, as it needs to be fetched only once at the beginning. With a 56kbps connection, about 7 codebooks can be downloaded during each operation interval.

Now consider a user who is at one semantic category with less than 7 sub-categories, and he is going to choose one sub-category with 15 icons. As the codebooks of all sub-categories are pre-fetched, only indices and residual errors of the 15 icons need to be downloaded. The total size is thus 15*2.4=36 Kbytes, where 2.4 kbytes is the average size of icons in the database. This will take 36/7=5.1 seconds to download. Compared to JPEG icons (with the average size of 2.9 kbytes), it saves about 1.1 seconds. More cases are given in Table 2.

Table 2. Some typical cases of the semantic category browsing

Case	Time Delay for Adaptive VQ	Time Delay for JPEG Compression	Difference
≤ 7 sub-categories 15 icons	5.1 s	6.2 s	1.1 s
≤ 7 sub-categories 30 icons	10.3 s	12.4 s	2.1 s
14 sub-categories 15 icons	5.5 s	6.2 s	0.7 s
14 sub-categories 30 icons	10.6 s	12.4 s	1.8 s

When there are more than 7 subcategories, there is a possibility, p, (e.g., 50% for 14 sub-categories) that the codebook of a selected sub-category is not pre-fetched. So $B*p$ is included in the final download size, and increases the download time delay.

5.4 Keyword Searching

Assume a keyword search returns a total of 150 icons from 15 categories. Also assume that 10 icons are displayed at one time due to the display size limit. In the worst case, the first 10 icons belong to 10 different categories. Thus with VQ compression, the client needs to download *10**(residual error+*codebook)=74 kbytes*. Compared to JPEG compression, the size of 10 icons is *29 kbytes.*

For the next 10 icons, the client should have pre-fetched the required codebooks. Thus there are only *10**(residual error)*=24 kbytes* to be downloaded. If users browse all the 150 return icons, the total download size is *410 kbytes* for the proposed method. The download size for JPEG compression is *435 kbytes.*

5.5 Query by Example

The Query by Example approach lets users to find visually similar images based on certain visual features. Different from keyword searching, *Query by Example* is usually a similarity matching process in high dimension feature spaces, and is much more computationally expensive. This query process delay gives the client a possible time slot to pre-fetch codebooks.

Generally, the feature clustering tree can be used as the index for the feature-based searching. The searching process is actually an automatic top-down "browsing" process. Thus the server is able to predict which clusters will contain the most search results. Before the final matches are found and sorted, related codebooks can be sent to clients.

To evaluate the performance of the feature cluster tree, we first compute the probability distribution of the number of clusters that contain top 10 results. This is fulfilled by performing example-based queries using all images in the database. As shown

in Figure 7, the top ten results usually come from only a few clusters. The average number of clusters is $\overline{N} = \sum_{i=1}^{10} i * p_i$, which is 3.

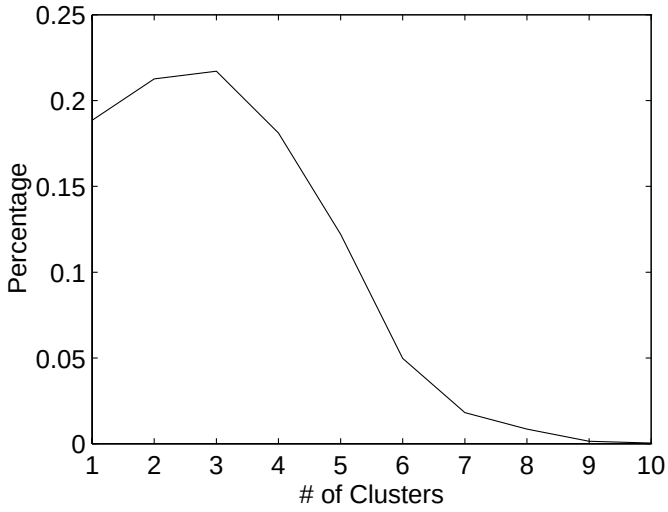


Figure 7. The histogram of the number of clusters that the top 10 return results belong to

Based on this analysis, on average, the client can pre-fetch required codebooks without causing much time delay. As the average size of icons under feature cluster based VQ is 2.2 kbytes (compared to 2.9 kbytes of JPEG), in downloading 10 icons through a 56kbps modem, users will save 1 second.

6. CONCLUSION

In this paper, we propose an adaptive VQ approach using content categorization, including both the semantic level and the feature level, to provide a differential compression scheme. It effectively explores image similarity in large image archives. We also show that this scheme is able to support flexible and optimal caching strategies under different typical access scenarios. The experimental results demonstrate that compared to JPEG compression, with the simple MSE-based LBG training the proposed technique can largely improve the compression rate and the retrieval response. Advanced training algorithms will be studied to create VQ codebooks that can capture the image similarity more effectively at different content levels.

REFERENCES

1. J. R. Smith and S.-F. Chang, *Visually Searching the Web for Content*, IEEE Multimedia Magazine, Vol. 4, No. 3, pp.12-20, 1997.
2. AltaVista Multi-Media Search Technology, http://www.altavista.com.
3. ISO/IEC 13818 – 2, Committee Draft, MPEG-2 Video.
4. A. Gersho and R. Gray, *Vector Quantization and Signal Compression*, Khiwer Academic Publishers, 1991.
5. B. Ramamurthi and A. Gersho, *Classified Vector Quantization of Images,* IEEE Trans. Commun. COM-34(11), pp1105-1115, Nov, 1986.
6. T. Kohonen, *Things You Haven't Heard about the Self-Organizing Map,* Proceedings of IEEE International Conference on Neural Networks, San Francisco, California, March 28 - April 1,1993
7. M. S. Lew, K. Lempinen, N. Huijsmans, *Webcrawling Using Sketches***,** Proceedings VISUAL97 conference, San Diego 15-17 Dec 1997, pp 77-84.