Web-WISE: Compressed Image Retrieval over the Web

Gang Wei, Dongge Li & I. K. Sethi
Vision and Neural Network Lab.
Dept. of Computer Science
Wayne State University
Detroit, MI 48202
{gaw, dil, sethi} @cs.wayne.edu

Abstract

Web-WISE is a system designed to address the need for efficient content-based seeking and retrieval of images on the web. It supports searching by multi-features, including color and texture. Web-WISE contains three automatic components, which are 1): Internet Agent responsible for searching the Web to fetch images; 2): Analysis Agent, which extracts the color and texture features of color JPEG images directly from compressed domain and 3): Query Agent, which explores the image database using certain metrics and returns a set of images (thumbnails) visually similar to the query image. Due to its generic design, adding other features to the system will be very natural, including the ability to index and retrieve other types of visual information and retrieval based on edge and shape. This paper presents the theoretic and practical background of the system, the schemes of these three components and the performance of the system.

Keywords: WWW, content-based retrieval, DCT, feature extraction, image database

1. Introduction

The explosive growth of images and videos on the World Wide Web is making the Web a huge resource of visual information. Today efficient content-based retrieval of images over WWW is becoming increasingly desirable due to the needs motivated by various applications. A straightforward solution is to store the images and their content descriptions (image visual features) in a local database and use a query scheme to implement the content-based retrieval. However this is not always feasible because of the large size of image files and the huge amount of images on the Web. The practical way should fully take advantage of the Web as storage rather than merely a resource of images. After extracting the features of the original images, we can just keep a reduced version of the images (thumbnails) and their corresponding locations (URLs) on the Web. Then the original images are deleted to save disk space. Then URLs can be given with thumbnails as the query results, enabling the user to access the original images from the Web.

Web-WISE is such a system. The name "WISE" stands for "Wayne Image Seeker". It employs a search engine to download images from the Web. Then it extracts features from the images in compressed domain and stores feature vectors in the database. Through a graphic interface, it enables users to do content-based image retrievals by clicking thumbmails. The result thumbnails of a query are displayed in descending order of the similarity values and are presented to the user as well as URL's.

2. An Overview of Web-WISE

2.1 Structure

Web-WISE is in nature a content-based WWW image search engine. There have been a lot of research work and practical applications about search engines over the Web. Yahoo, AltaVista and Lycos are typical applications of this technique. Most of these kind of search engines are keyword-oriented, which enable users to retrieve information by typing in the subjects they are interested in. The documents are indexed by keywords extracted from them. The system developed by J.R. Smith and S.F.Chang supports both keywords and content-based retrieval [2]. It categorizes image by keywords in a hierarchical structure and extracts color regions of images and get their color histograms. Keyword-based systems work well for tasks like "Give me images of this type". But if we need a query like "Give me images that look like this", content-based retrieval has to be employed. Research carried out by B. Agnew et al. [3] and O. Munkelt et al. [4] index images by their visual contents however using different metrics. In Web-WISE, we emphasize on content-based retrieval and feature extraction directly from compressed domain.

Web-WISE consists of three modules: Internet Agent, Analysis Agent and Query Agent. Those modules work independently. The overall structure of the system is illustrated by Fig. 1.

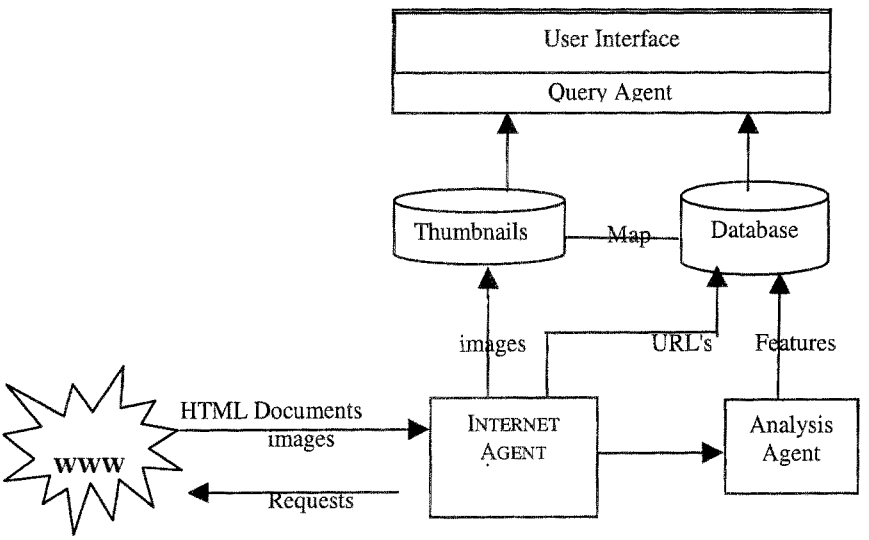


Fig. 1: The Overall structure of Web-WISE

- The Internet agent automatically traverses the Web by following hypertext links, fetching HTML documents and JPEG images to the local disk. The locations (URL's) of the images are also stored in a list in plain text format.

- Images are fed to the Analysis Agent which directly extracts color and texture features of the images in compressed domain. The feature values are imported to the database.
- Thumbnails are saved in a separate directory. There is a table which maps between the thumbnail filenames and the corresponding records in the database. URL list is also imported to the database.
- Query agent gets query image name from user interface, selects candidate images from the database, compares the query image with each candidate and assigns the candidate with a similarity value.
- The result thumbnails are displayed in descending order of their similarity values.

2.2 User Interface

Fig. 2 is the users interface of Web-WISE. To do an iteration of query, the user simply clicks the thumbnail. Randomly preview of the images in the database is allowed by clicking the "Random" button. As both global and local dominant histogramming retrievals are implemented, the user can try different schemes by choosing corresponding experiment numbers. Several parameters, which are to be discussed later, for similarity computations can also be set through the interface (however, each parameter has its default value).

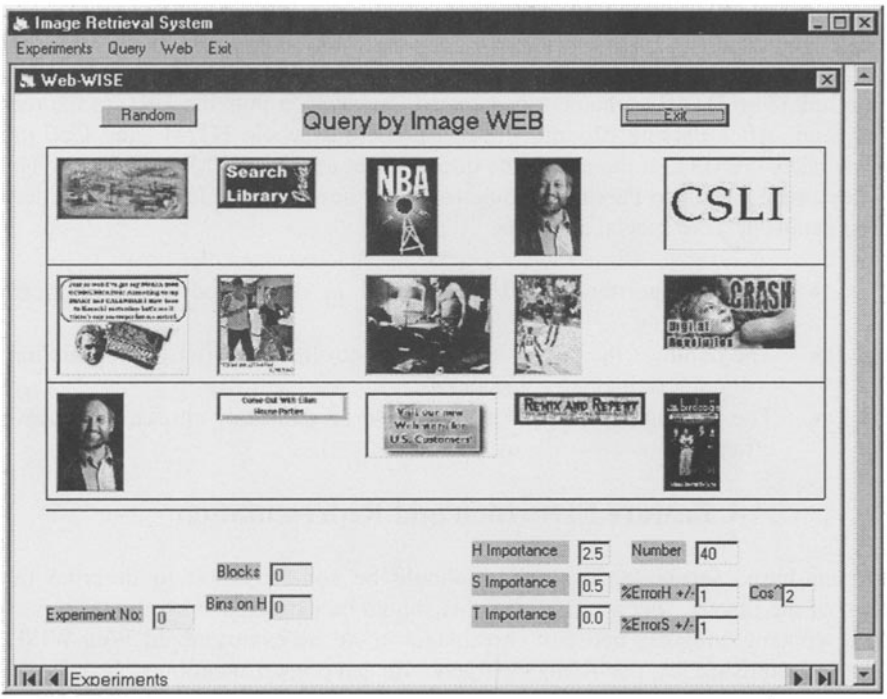


Fig. 2 Web-WISE interface

3. Searching Images on the Web

From a user's view, the Web is a large set of page documents, many of which contain reference to each other by hyperlinks. In this sense, the Web is a partially connected directed graph, with each page considered as a node and links as edges. Images on the Web are available through two forms: embedded and linked. To acquire images, a search engine should search the Web in some order from an arbitrary starting node and parse HTML documents to detect and download images.

The component in Web-WISE which explores the web for images is called Internet Agent. It is composed of three interactive sub-modules: GetLink, GetImage and Parsing Module.

The GetLink and GetImage modules are responsible for downloading HTML documents and images, respectively, from the URL's given by the Parsing Module. They hide the details of TCP/IP connections and HTTP requests from the Parsing Module. The Parsing Module analyzes the HTML documents fetched by GetLink module to extract URL's of JPEG images and links to more HTML documents, then send the URL's to the GetImage module and GetLink module.

The Internet agent traverses the Web in breadth-first search order. Starting from a designated URL, each time the GetLink Module fetches an HTML document and passes it to the Parsing Module. The Parsing Module is a push-down automaton model which analyzes HTML documents. When it detects the presence of a JPEG image, the URL is immediately sent to the GetImage Module to get the image. When a hyperlink to an HTML document is detected, however, it puts the URL to the rear of a queue. After Parsing Module finishes process a whole HTML, the GetLink Module takes the URL at the top of the queue to get another HTML document. The new document is sent to Parsing Module to start a new parsing. In implementation, several details deserve special attention.

- Certain algorithms need to be used to detect and avoid duplicate downloading.
- The depth of the search should be controlled (otherwise the searching will not terminate in some cases).
- The starting point of the agent should be delicately chosen to improve efficiency.

4. Feature Extraction and Representation

In content-based image retrieval, there should be some method to describe the content of the image. That is, some features should be extracted from the images so that the content similarity between two images could be evaluated. In Web-WISE, this is accomplished by the Analysis Agent. As have been mentioned before, the search engine fetches JPEG images, which are in compressed form. By extracting features directly from compressed data, we reduced the cost of computation significantly.

4.1 JPEG/DCT Model

Here we briefly review the JPEG/DCT compression algorithm [16] before describing how to extract feature information from compressed data. In baseline JPEG algorithm, the original image is transformed to frequency domain using forward discrete cosine transform (FDCT) on a block by block basis so that coefficients within the blocks are decorrelated and can thus be treated independently without loss of compression efficiency. The standard size of the block is 8*8. Then a weighted quantization is applied to each coefficient within the blocks to discard visually unimportant coefficients. This is a lossy process. Each quantized DCT block will become a sparse matrix. Then a lossless entropy coding is performed to obtain further compression. To decompress a JPEG image, entropy decoding is first applied to get quantized coefficients, then dequantization is performed, and the original image can be reconstructed by performing Inverse discrete consine transform (IDCT). FDCT and IDCT are the central parts of the JPEG compression/decompression schemes, which are defined as:

$$F_{uv} = \frac{c_u c_v}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} f(i,j) \qquad (1)$$

And

$$f_{ij} = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C_u C_v}{4} \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} F(u,v) \qquad (2)$$

Where

$$c_u, c_v = \frac{1}{\sqrt{2}} \quad for\ u, v = 0,\ and\ 1\ othewise$$

JPEG compression uses YCbCr color coordinate system, in which Y is the luminance component while Cb and Cr are chromatic components. Due to the aspects of human visual system, the sampling scale of Y is "finer" than that of Cb and Cr.

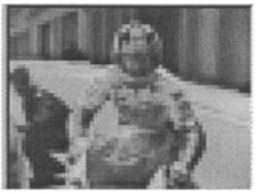
4.2 Properties of DCT coefficients

In JPEG decompression processes, IDCT is the most time-consuming part. Our approach extracts feature directly on the DCT coefficients without transforming them to spatial domain, thus considerable amount of time is saved.

By definition, a DCT coefficient F_{uv} is a linear combination of all pixel values in the block. Particularly, coefficient F_{00} (DC coefficient) is the average value of the original pixels in the block. If we extract the DC's of each block to construct a new image, it will be a downsampled version of the original one [9]. Fig. 3 compares the

original image and the reconstructed image using only DC coefficients of component Y. Here we enlarged the reconstructed image. Although looking rather blocky, it bears a lot of information of the original one.

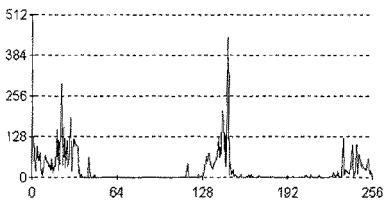
(a) Original Image



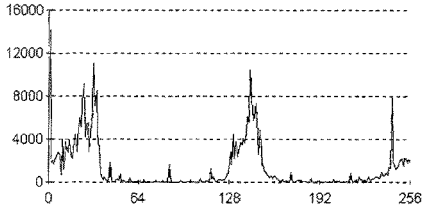
(b) Reconstructed Image

Fig. 3 Original Image and its Reconstructed Image

Color histogram reflects the color distribution in an image. We observed that the histogram constructed on DC coefficients resembles closely that built on the original image. Fig. 4 indicates this fact. Thus we can use the reconstructed histogram as an estimation of the color distribution for the original one.



(a) Histogram of the Reconstructed Image



(b) Histogram of the Original Image

Fig. 4 Histograms of the reconstructed and original images

The above two figures are the histograms of the hue component of the reconstructed image and original image, respectively. To get Fig. 4 (a), the DC coefficient in YCbCr should be transformed to HSI (hue, saturation and Intensity) color coordinate system first. The motivation of this transformation will be discussed in the next section.

The remaining coefficient (AC coefficients) reflects variations within a block [7]. For example, F_{10} essentially depends on the intensity difference in the vertical direction within a block. This relationship can be explained by a simple analysis of the definition of discrete cosine transform.

$$F_{10} = \frac{c_1 c_0}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} \cos \frac{(2i+1)\pi}{16} f(i,j) = \frac{c_1 c_0}{4} \sum_{i=0}^{7} \cos \frac{(2i+1)\pi}{16} \sum_{i=0}^{7} f(i,j) \qquad (3)$$

Considering cos (π-θ)=-cosθ, the above equation can be expanded as

$$F_{10} = \frac{c_1 c_0}{4}\left[\cos\frac{\pi}{16}\left(\sum_{i=0}^{7} f(0,j) - \sum_{i=0}^{7} f(7,i)\right) + \cos\frac{3\pi}{16}\left(\sum_{i=0}^{7} f(1,j) - \sum_{i=0}^{7} f(6,j)\right)\right. \\ \left. + \cos\frac{5\pi}{16}\left(\sum_{i=0}^{7} f(2,j) - \sum_{i=0}^{7} f(5,j)\right) + \cos\frac{7\pi}{16}\left(\sum_{i=0}^{7} f(3,j) - \sum_{i=0}^{7} f(4,j)\right)\right] \quad (4)$$

Similarly, F_{01} and F_{11} depend on the variation in the horizontal and diagonal direction, respectively. Fig. 5 shows the image constructed with the maximum value of F_{01}, F_{10} and F_{11} within blocks of Y component.

(a) Original Image

(b) Reconstructed Image

Fig. 5 Original Image and Reconstructed Image with AC coefficients

From above we can see that if a region in the original image has little variation, the corresponding region in reconstructed image will have low intensity near to zero. Otherwise they will have high intensities. The reconstructed image also provides minimal information about the edges.

4.3 Feature Extraction and Representation

All content-based image retrieval systems employ a set of features to describe images in the database. Color is maybe the most widely used feature. Reflecting the distribution of color within images, color histograms prove to be an efficient feature in content-based retrieval. I.K. Sethi et al. employed localized dominant hue and saturation values [1], which can achieve better performance than the use of global or local histogramming because it contains the information of the spatial distribution of color. J.R. Smith and J.K.Chang isolated color regions using back-projection and extracted properties for each region.

In Web-WISE, histograms are built on images reconstructed with only DC coefficient. Those images have already been reduced in size by 8*8 (Y component) or 16*16 (Cb and Cr components). If we further divide them using fixed image partition or color region extraction scheme, the number of pixels in each region may be too small to give a description of the feature with desired precision. Therefore, global histogram is the default criteria in determining the similarity although local histogramming retrieval also supported in our system.

Color features are represented in HSI (Hue, saturation and intensity) color coordinate system, which correlates human visual system better than YCbCr space. Hue and saturation are color components in which hue refers to the average spectral wavelength and saturation reflects the purity of the color. Intensity stands for brightness. After transforming the YCbCr components of the JPEG images to HSI color space, the Analysis agent calculates the global histograms of H and S channels.

Since each hue or saturation value is represented with eight bits, there are 256 bins in each histogram, which makes the number of dimensions of the feature vector very large for each image. To get compact representation of image feature for efficient storage and retrieval, quantization of the histograms is required. Noting that human visual system is more sensitive to of hue information, the quantized hue histogram has 16 bins while saturation histogram has 8 bins. Non-uniform quantization is applied to get better performance [17]. The distribution of hue and saturation values of all the images in database are calculated to decide the range of each bin so that all bins contain the same number of pixels as far as the all images in the database are concerned. This ensures that for any given pixel, its probability of falling into any bins is equal.

The texture feature is used as an auxiliary measure in Web-WISE. The maximum values of F_{01}, F_{10} and F_{11} of each DCT block are used to reconstruct an image as Fig.5 (b). Relying on a fixed image-partitioning scheme, we divide the reconstructed image into 4*4 blocks and compute the average intensity of each block. Each value reflecting the extent of variation in that block, the spatial distribution of texture can be represented by the 16-dimension vector.

Thus we represent an image with a 40-dimensional vector, in which 16 dimensions and for hue, 8 for saturation and 16 for texture.

5. Similarity Metrics and Image Query

The results of the retrieval are determined by the similarity between the query image and target images in the database. This section describes how we compute the similarity between two images are how the result of a query is generated.

5.1 Similarity Computation

As each image is represented by a 40-dimensional vector, the whole image database could be seen as a 40-D space where all images are distributed. In a query like "find k images that is most similar to this image", actually we are trying to decide the k nearest neighbor of the query image in this 40-D space, using some distance metrics. The smaller the distance is, the closer two objects are and thus the larger the similarity value.

The Query Agent gets the name of the query image from the user interface, then gets the feature vectors for the query image and the candidate images from the database. Distance and similarity values between query image and each candidate image are calculated. The interface displays the thumbnails of result images in descending

order of the similarity values. Since the sensitivity of human visual system may vary a lot to different features, 3 coefficients for the three sections (hue, saturation and texture) of the feature vector can be specified through the user interface to indicate the significance of each feature in determining the similarity between images. The similarity between two images Q and T is defined by the following formula:

$$s(Q,T) = \frac{1}{(1+(aD_h(h_Q,h_T)+bD_s(s_Q,s_T)+cD_t(t_Q,t_T)))} \qquad (5)$$

Here D_h, D_s and D_t are functions to calculate the vector section distances between Q ant T in hue, saturation and texture, respectively. Their parameters are feature vectors defined in section 4.3. The coefficient a, b and c are importance weights for the vector section distances, which are to be specified by the user. By default, a > b (we assume more importance to hue than to saturation), c is set to zero (texture is used as an auxiliary feature). Let M be the number of dimensions of the vectors, D_h, D_s and D_t are defined as:

$$D(u,v) = 1 - \frac{\sum_{i=0}^{M-1} \min(u(i),v(i))}{\min(\sum_{j=0}^{M-1} u(j), \sum_{k=0}^{M-1} v(k))} \qquad (6)$$

5.2 Image Filtering: Reduce the number of candidates

The similarity metrics described have some drawbacks. First, the computation complexity is high. If the similarity computation is applied to all the images in the database, there would be no surprise that the system is slow. Secondly, all bins within one section have the same impact on the distance measure. However, human visual system is usually more sensitive to the dominant color of images. In another word, to get better performance, the bins with larger values should assume greater significance than those with smaller values in determining the distance of vectors.

In query agent, a simple but effective method is used to solve the above two problems. For each image record in the database, an extra field, max_H_bin, is added, which represents the number of the bin with the maximum value (this is the dominant color bin) in the hue feature vector. In the query process, this field can be used as a filter to kick out images that do not seem promising so that the number of candidate images to be applied the similarity computation can be greatly reduced. For each image in the database, if the difference between its max_H_bin and that of the query image is greater than a threshold, then their dominant color is very likely to be so different that it is not worth further analysis as a potential result. In this case, this image can be discarded and the similarity value is simply set to 0. Otherwise it is chosen as a candidate to which the similarity calculation described above will be

applied. For further filtering, one more field, max_S_bin could be used with similar method.

After the filtering process, we have a set of candidate. Apparently, applying similarity computation only on these images can achieve great speedup. Each candidate gets a value through the similarity computation. Images with the largest similarity values are returned as the query results.

6. Performance Evaluation

Presently there are about 1000 images in the database of Web-WISE. As two retrieval schemes (global and local dominant histogramming) are supported, there are two tables in the database. An image is represented with one record in each table. For a default query, the global histogramming is used. The record contains a 40-D feature vector, the URL of the image, and two additional fields representing the numbers of bins with maximum values for hue and saturation, for the purpose described in Section 5.2. Fig. 6 is an example of the result of a default query. In this query only global hue and saturation are compared, with the importance coefficient of each being 2.5 and 0.5, respectively. The upper-left image is the query image and the numbers above each image are the similarity values. Note that the value for query image itself is 1.

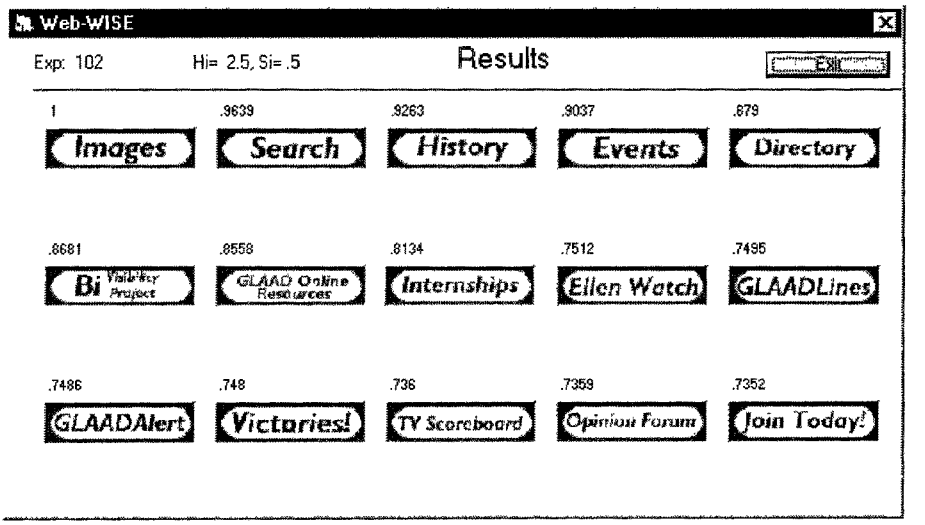


Fig. 6 The set of result images

Texture is an optional feature used in Web-WISE. Its effect on similarity value can be enabled or disabled by setting its weight to 1 or 0 from the user interface. Fig. 7 compares the performances of queries with and without using texture for the same query image.

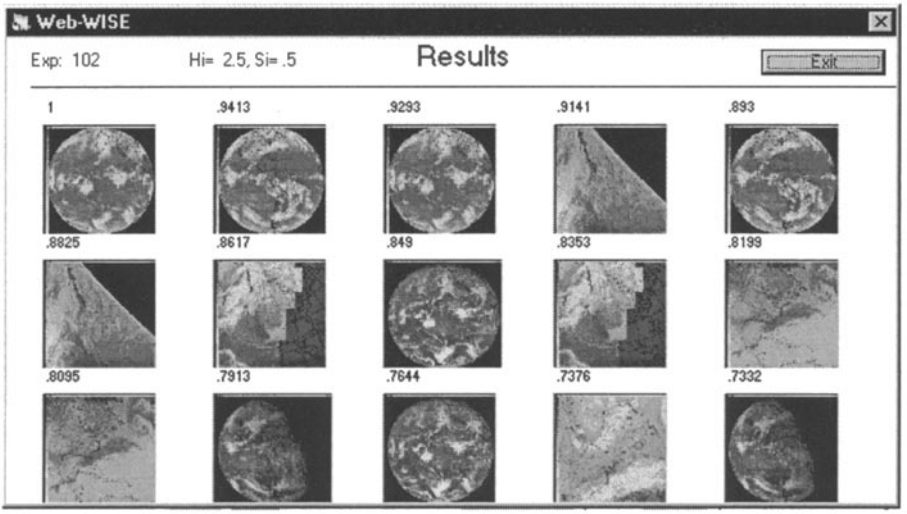


Fig. 7 (a) Query results without using texture feature

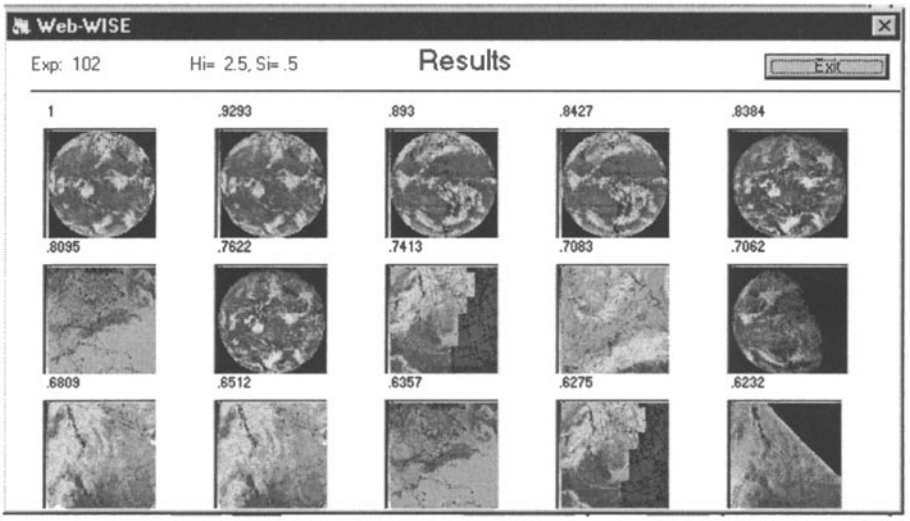


Fig. 7(b) Query results using texture feature

Local dominant color retrieval was also implemented as an optional function of Web-WISE. In this scheme, each image is partitioned into 4*4 blocks. For each block we extract the dominant hue and saturation value (peak area value, as described in [1]). By doing this we represent an image with a 32-D vector, 2 dimensions for a block. Fig. 8 compared this scheme with the global histogramming retrieval. Due to the reason explained in Section 4.3, this scheme outperforms global histogramming scheme only occasionally. Moreover, for image less than 32 pixels wide or high, the local scheme will fail in the feature extraction stage (an image constructed with DC

coefficients is 8*8 downscaled, thus an image smaller than 32*32 pixels can not be partitioned 4*4 properly). These images have to be discarded.

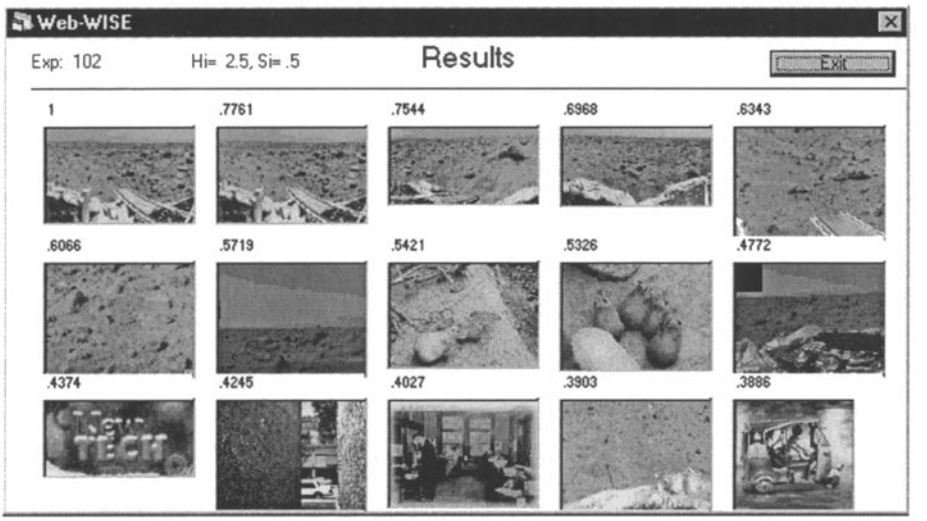


Fig. 8 (a) Query result using global histogram

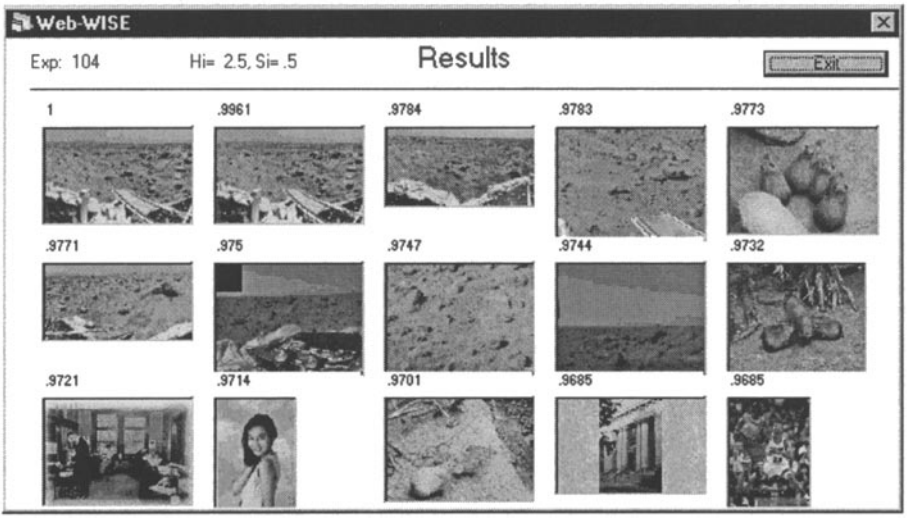


Fig. 8 (b) Query result using local dominant features

However, we observed that local scheme is efficient in retrieving images with salient color regions, provided the images are sufficiently large. This fact is illustrated in Fig. 9.

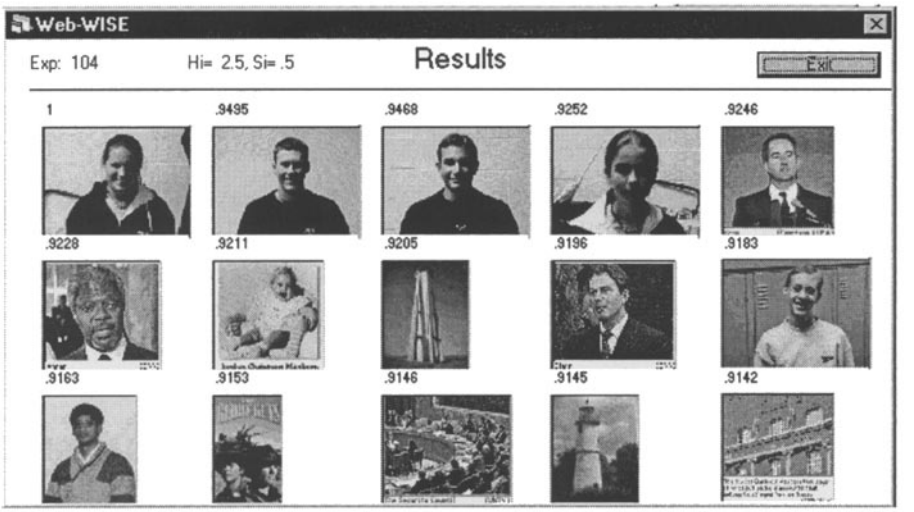


Fig. 9 Query results using local dominant color

7. Future Work

Web-WISE is a highly adaptive system. Two extensions will be explored in our future research. First, we will enable it to retrieve videos on the Web. This requires a minor change in the Internet Agent. For the Analysis agent, the ability to detect shot cut will be added to get key frames, and the features should be extracted from each key frame. In query process, the similarity should be based on the sequence of key frames instead a single still image. Second, edge information will be taken into account. This will be more efficient if we emphasize on the profile of the objects in an image rather than the distribution of color.

8. References

1. I.K. Sethi et al., "Color-WISE: A system for image similarity retrieval using color," Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 3312.
2. J.R. Smith and S.F. Chang, "An image and video search engine for the World-Wide-Web," Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 3022.
3. B. Agnew et al., "Multimedia indexing over the Web." Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 3022.
4. O. Munkelt et al., "Content-based image retrieval in the World Wide Web: a web agent for fetching portraits," Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 3022
5. M. Waston, "Programming Intelligent Agent for the Internet", Computing McGraw-Hill
6. B.L. Yeo, "Efficient Processing of Compressed images and video," Ph.D. dissertation, Dept. of Electrical Engineering, Princeton University

7. B. Shen, I.K. Sethi, "Direct feature extraction from compressed images," Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 2670
8. N.V.Patel, I.K. Sethi, "Compressed video processing for cut detection ", IEE Proceedings, Vis. Image Signal Process., Vol. 143
9. X. Wan, C.J.Kuo, "Image Retrieval based on JPEG Compress Data," SPIE Proceedings, Multimedia Storage and Archiving Systems, Vol. 2196
10. J.R.Smith, S.F.Chang, "Tools and Techniques for color image retrieval," Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 2670
11. A. Berman, L. Shapiro, "Efficient image retrieval with multiple distance measures, " Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 3022
12. V.N. Gudivada, V.V. Raghvan, "Content-based Image Retrieval System," IEEE Computer, Vol. 28, No. 9.
13. H. Lu, B. Ooi and K. Tan, "Efficient Image Retrieval by Color Contents," Proceedings International Conference on Applications of Databases, 1994
14. S. Santini, R Jain, "Similarity Queries in image databases," IEEE International Conference on Computer Vision and Pattern Recognition, pages 646-651, San Francisco, CA, USA, June 1996.
15. M.Stricker and M. Orengo, "Similarity of color images, " Proceedings of the SPIE: Storage and Retrieval for Image and Video Database, Vol. 2420
16. G.K.Wallace, "The JPEG Still Picture Compression Standard", Communications of the ACM, vol. 34, no. 4, Apr. 1991
17. R.C.Gonzalez, R.E.Woods, "Digital Image Processing", Addison-Wesley Publishing Company