Computer Vision and Image Understanding
Vol. 75, Nos. 1/2, July/August, pp. 86–98, 1999
Article ID cviu.1999.0765, available online at http://www.idealibrary.com on IDEAL®

Unifying Textual and Visual Cues for Content-Based Image Retrieval on the World Wide Web

Stan Sclaroff, Marco La Cascia, and Saratendu Sethi

Computer Science Department, Boston University, Boston, Massachusetts 02215
E-mail: sclaroff@bu.edu, marco@bu.edu, sethi@bu.edi

and

Leonid Taycher

Cognex Corporation, One Vision Drive, Natick, Massachusetts 01760
E-mail: ltaycher@cognex.com

A system is proposed that combines textual and visual statistics in a single index vector for content-based search of a WWW image database. Textual statistics are captured in vector form using latent semantic indexing based on text in the containing HTML document. Visual statistics are captured in vector form using color and orientation histograms. By using an integrated approach, it becomes possible to take advantage of possible statistical couplings between the content of the document (latent semantic content) and the contents of images (visual statistics). The combined approach allows improved performance in conducting content-based search. Search performance experiments are reported for a database containing 350,000 images collected from the WWW. © 1999 Academic Press

Key Words: **content-based image database indexing and retrieval; vector space representation; latent semantic indexing; world wide web search engines.**

1. INTRODUCTION

The growing importance of the World Wide Web has led to the birth of a number of image search engines [18, 23, 46, 46a, 48]. The Web's staggering scale puts severe limitations on the types of indexing algorithms that can be employed. Luckily, due to the scale and unstructured nature of the WWW, even the most basic indexing tools are welcome. Existing image engines allow users to search for images via an SQL keyword interface [18, 48] and/or via query by image example (QBE) [23, 46, 48].

In QBE, the system presents an initial page of representative (or randomly selected) image thumbnails to the user [15]. The user then marks one or more images as relevant to the search. The visual statistics for these images are then used in defining a query. The user's success in locating images in the database depends in great part on which images appear within this initial group of thumbnails. Given the numerous and diverse images available in a WWW index, it is difficult to guarantee that there will be even one relevant image shown in the initial page. We call this the *page zero problem*.

Other WWW image engines allow the user to form a query in terms of SQL keywords [18, 48]. This alleviates the page zero problem, since the user can give text information that narrows the scope of possible images displayed on page zero. To build the image index, keywords are extracted heuristically from HTML documents containing each image, and/or from the image URL. It is assumed that textual cues extracted from the text near an image in the HTML document will provide clues about that image's semantic content. Unfortunately, it is difficult to include visual cues within an SQL framework. This results in systems that inherently force visual information into textual form, or systems that treat textual and visual cues disjointly.

We present an approach that is a natural integration of textual cues and visual cues. The proposed technique allows for textual information and visual information to be used together in a vector space representation of the images without giving any different and arbitrary importance to the different sources of information. By truly unifying textual and visual statistics, one would in fact expect to get better results than either used separately.

In our system text statistics are captured in vector form using latent semantic indexing (LSI) [12, 29a]. The text associated with an image, as it appears in the HTML document, is represented by low-dimensional vectors that can be matched against user queries in the LSI "semantic" space. Visual statistics (e.g., color and orientedness) are also computed for each image. The LSI vector and visual statistics vector are then combined into a unified vector that can be used for content-based search of the resulting image database.

By using an integrated approach, we are able to take advantage of possible statistical couplings between the content of the

1077-3142/99 $30.00

document (latent semantic content) and the contents of images (image statistics). Furthermore, LSI implicitly addresses problems with synonyms, word sense, lexical matching, and term omission.

A quantitative evaluation of the retrieval performance of our system, based on the *target testing paradigm* [11], is also reported.

2. RELATED WORK

To date, there have been a number of query-by-image content (QBIC) demos available via the web, e.g., Virage [25], IBM QBIC [15], Cypress [16], Photobook [40], VisualSeek [48], Jacob [29], and Mars [38]. Nearly all systems include some form of color- and texture-based image similarity measures. In addition, some systems provide search on image composition [25], shape [15, 33, 40], faces [5, 40], and/or groupings of colored blobs [37]. None of these systems provides a Web search engine in that each only operates on a local demo database of a few thousand images stored at the host web site. However, the algorithms developed in these and other QBIC systems serve as an excellent starting point for building WWW image search engines.

ImageRover is joined by others in the first wave of image search engines: AltaVista's A/V Photo Finder [1], Yahoo's Image Surfer [52], Lycos media search tool [32], Corbis Picture Experience [2], WebSeer [18], WebSeek [48], ImageSearch [30], and PicToSeek [23]. Two of these systems [2, 32] use only standard textual information within a SQL framework and do not allow content-based search. The others allow some form of content-based search.

PicToSeek [23] indexes images collected from the WWW using purely visual information (no text cues). In particular, invariant color image features are extracted from the images. In addition, the collected images are automatically cataloged by image analysis methods into various image styles and types: JFIF–GIF, gray–color, photograph–synthetic, size, date of creation, and color depth.

A similar system, WebSeek [48], performs semi-automatic classification of images into a taxonomy of categories, using associated text and filename cues. Color histogram based similarity matching can then be used to find images with a similar color content within a category or over the entire catalog.

In Yahoo's Image Surfer [52], the images, collected from the Internet, are manually classified into categories. The user is allowed to search for visually similar images inside the category. The image similarity measure is based on color, texture, and structure.

ImageSearch [30] allows the user to specify the search example via an iconic sketch interface. The interface allows the user to position representative image icons to form the example image (e.g., human face, sky, sand stone, and grass). Search of the index is then conducted using trigrams derived from the binary contour map of each image. A weighting framework is then used in computing the similarity of image trigram features at multiple scales, based on Kullback relative information.

AltaVista's A/V Photo Finder [1] indexes images based on textual and visual cues. As a precomputation, index terms are extracted from the HTML document containing each image, and image features are extracted via Virage's Image Engine [25]. During a query, the user first specifies words that describe the desired image. Photo Finder then retrieves thumbnails of images in the index that best match the initial keyword query. For any particular retrieved image, the user can ask Photo Finder to display 24 other "visually similar" images. Unfortunately, these visually similar images are retrieved ignoring the key words specified earlier in the query. Visual and textual cues cannot be combined in guiding the query, nor is relevance feedback possible.

In WebSeer [18] the information for finding images on the WWW is obtained from two sources: the associated HTML text and the image itself. The text it uses to describe an image is easily found in the HTML document containing the image. In particular, the filename, the caption, the alternate text, the hyperlinks, and the HTML title are considered relevant. Information obtained directly from the image is some statistics (width, height, file size, type, etc.) and the number of faces present. These two sources of information are then integrated in a standard SQL framework.

A different approach in the exploitation of textual and visual information is taken in PICTION [49] (picture and caption), though not to images on the WWW. In this system the image caption is used as a cue to identify human faces in an accompanying newspaper photograph. Even though this technique could be successfully used to index magazine pictures, its application to HTML documents on the WWW is not straightforward.

If carefully extracted, keywords may help guide the search to basic categories. Unfortunately, keywords may not accurately or completely describe image content. Cues about image content must also come directly from the image, especially once the user wants to narrow the search within a basic category, e.g., cats or cars. The ImageRover approach is most similar in spirit to that of WebSeer; however, ImageRover differs in that it allows searches of Web images based directly on image content. ImageRover's overall system architecture and underlying algorithms will now be described in detail.

3. APPROACH

The system we implemented uses precomputed features. For each image collected from the Internet a vector characterization, based on the visual content and on the surrounding text, is computed off line. The vectors are then used in the query phase. In other words, the search for images on the WWW is conducted as a simple k-nearest-neighbor problem search with relevance feedback [46] in a local database.

In our current implementation the visual statistics computed are a color histogram and an orientation histogram. Other image

features that can be expressed in vector form can be easily added to the system. The textual statistics are based on LSI [12] and consist of a fixed size vector per image.

To start a query the user inserts a few words related to the images he or she is looking for. The system projects the query document (i.e., the words given by the user) into the LSI space to obtain a vector representation of the query and to compute the k-nearest neighbors in the LSI subspace of the complete feature space. This solves the *page zero* problem as, in general, most of the images returned by the system are related by context to what the user is looking for. The user can then steer the system toward the desired images by simply selecting the most relevant images and iterating the query. These refinements of the query are performed in the complete feature space and the different sources of information are mixed through our relevance feedback framework.

Basically, the way we combine the textual and the visual statistics is based on user feedback. Dissimilarity is expressed as a weighted combination of distances between subvectors. The main idea is to give more weight to the features that are consistent across the example set chosen by the user.

3.1. Latent Semantic Indexing (LSI)

The context in which an image appears can be abstracted from the containing HTML document using the LSI method [12]. LSI works by statistically associating related words to the semantic context of the given document. The idea is to project words in similar documents to an implicit underlying semantic structure. This structure is estimated by a truncated singular value decomposition (SVD). The latent semantic indexing procedure is as follows.

To begin with, each image's associated HTML document is parsed and a word frequency histogram is computed. The documents in the database are not similar in length and structure. Also, all words in the same HTML document may not be equally relevant to the document context. Hence, words appearing with specific HTML tags are given special importance by assigning them higher weight as compared to all other words in the document.

The system assigns different weights to the words appearing in the *title* and *headers* and in the *alt* fields of the *img* tags along with words emphasized with different fonts like *bold* and *italics* (see Table 1). These weight values were chosen heuristically according to their approximate likelihood of useful information that may be implied by the text. Selective weighting of words appearing between various HTML tags helps in emphasizing the underlying information of that document. Related weighting schemes are proposed in [18, 43].

The weights given are computed by just counting a single occurrence of the word as many times as the corresponding weight value for which it qualifies in the table. For example, if the word "*satellite*" appears between the *title* tags in the HTML document once, then it is counted as 5.00 instead of 1.00. All subsequent occurrences of the word are counted as weight values according to category in Table 1.

TABLE 1
Word Weights Based on HTML Tags

HTML tags	Weights
Alt field of IMG	6.00
Title	5.00
H1	4.00
H2	3.60
H3	3.35
H4	2.40
H5	2.30
H6	2.20
B	3.00
Em	2.70
I	2.70
Strong	2.50
⟨No tag⟩	1.00

In addition, words appearing before and after a particular image are also assigned a weight based upon their proximity to the image. The weighting value is computed as $\rho \cdot e^{-2.0 \cdot pos/dist}$, where *pos* is the position of the word with respect to the image and *dist* is the maximum number of words considered in applying such weighting. In the current implementation, the *dist* is 10 and 20 for words appearing before and after the image respectively. The constant $\rho = 5.0$, so that the words nearest to the images get weighted slightly less than and equal to the words appearing in the *alt* field of that image and the *title* of the Web page, respectively.

The choice of this weighting function was dictated by the assumption that words close to an image in the HTML document are related to the image itself. We also assumed that the degree of relatedness decreases exponentially with the distance from the image. This may not always be the case; however, we found these assumptions reasonable in most Web pages we visited in an informal survey of WWW sites.

As a result of this weighting scheme, images appearing at different locations in an HTML document will have different LSI indices. Each image in the Web page is now associated with its unique context within a document by selectively weighting words based on proximity to the image.

A term $\times$ image matrix A is created; the element a_{ij} represents the frequency of term i in the document containing image j with a weight based on its status and position with respect to the image. Retrieval may become biased if a term appears several times or never appears in a document. Hence further local and global weights may be applied to increase/decrease the importance of a term in and among documents. The element a_{ij} is expressed as the product of the local $(L(i, j))$ and the global weight $(G(i))$. Several weighting schemes have been suggested in the literature. Based on the performance reported in [14], the *log-entropy* scheme was chosen. According to this weighting

scheme, the local and the global weights are given as

$$a'_{ij} = L(i, j) \times G(i) \tag{1}$$

$$L(i, j) = \log(a_{ij} + 1) \tag{2}$$

$$G(i) = 1 - \sum_k \frac{p_{ik} \log(p_{ik})}{\log(ndocs)} \tag{3}$$

$$p_{ik} = tf_{ik} \Big/ \sum_k tf_{ik}, \tag{4}$$

where tf_{ik} is the pure term frequency for term i in HTML document k not weighted according to any scheme, and *ndocs* is the number of documents used in the training set.

The matrix $A' = [a'_{ij}]$ is then factored into U, Σ, V matrices using the singular value decomposition

$$A' = U \Sigma V^T, \tag{5}$$

where $U^T U = V^T V = I$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$, and $\sigma_i > 0$ for $1 \le i \le r$, $\sigma_j = 0$ for $j \ge r + 1$.

The columns of U and V are referred to as the left and right singular vectors, respectively, and the diagonal elements of Σ are the singular values of A. The first r columns of the orthogonal matrices U and V define the orthonormal eigenvectors associated with the r nonzero eigenvalues of AA^T and $A^T A$, respectively. For further details about the SVD and the information conveyed by the matrices, readers are directed to [7].

The SVD decomposes the original term-image relationships into a set of linearly independent vectors. The dimension of the problem is reduced by choosing the k most significant dimensions from the factor space which are then used for estimating the original index vectors. Thus SVD derives a set of uncorrelated indexing factors, whereby each image is represented as a vector in the k-space

$$\bar{\mathbf{x}}_{\text{LSI}} = q^T U_k \Sigma_k^{-1}, \tag{6}$$

where q is the word frequency histogram for the image with the appropriate weight applied. The resulting LSI vector $\bar{\mathbf{x}}_{\text{LSI}}$ provides the context associated with an image and is combined with its computed visual feature vectors and stored in the database index.

3.2. Visual Statistics

Several color spaces, histogram techniques, and similarity metrics have been proposed (see for example [15, 39, 47, 50]). Due to the lack of a common testbed and the high degree of subjectivity involved in image retrieval tasks, it is still unclear which technique most closely mimics human perception [22]. Similarly, the use of texture content [15, 17, 31, 42, 51] for image retrieval has still some unanswered questions. Finally, even though the Brodatz [10] data set has been used extensively in comparing different techniques for texture indexing, it is not clear that one can draw any definitive conclusion about their efficacy for indexing general imagery on the WWW.

The visual statistics we use to describe an image are the color histogram and dominant orientation histogram. The statistics are computed over the whole image and five overlapping regions [46]. The technique provides a reasonable compromise between computational complexity and descriptive power. In any case, the choice of the optimal image statistics is beyond the current scope of the ImageRover system. The ImageRover architecture is general enough to allow inclusion of any image descriptor that can be expressed in vector form. This allows easy extension of the system when future, improved representations become available.

3.2.1. Color. Color distributions are calculated as follows. Image color histograms are computed in the CIE $L^*u^*v^*$ color space [27], which has been shown by Gargu and Kasturi [21] to correspond closely to the human perception of color. To transform a point from RGB to $L^*u^*v^*$ color space, it is first transformed into CIE XYZ space. In our implementation we use the conversion matrix for CIE Illuminant C (overcast sky at noon). The $L^*u^*v^*$ values are then calculated as

$$L^* = \begin{cases} 25 \cdot \left(100 \cdot \frac{Y}{Y_0}\right)^{1/3} - 16 & \text{if } \frac{Y}{Y_0} \ge 0.008856 \\ 903.3 \frac{Y}{Y_0} & \text{otherwise} \end{cases} \tag{7}$$

$$u^* = 13L^*(u' - u'_0) \tag{8}$$

$$v^* = 13L^*(v' - v'_0) \tag{9}$$

$$u' = \frac{4X}{X + 15Y + 3Z} \tag{10}$$

$$v' = \frac{9Y}{X + 15Y + 3Z}, \tag{11}$$

where the reference values are $(X_0, Y_0, Z_0) = (0.981, 1.000, 1.182)$ and $(u'_0, v'_0) = (0.2010, 0.4609)$, for white under CIE Illuminant C.

For each of the subimages, the color distribution is then calculated using the histogram method [26]. Each histogram quantizes the color space into 64 (4 for each axis) bins. Each histogram is normalized to have unit sum and then blurred.

3.2.2. Texture orientation. The texture orientation distribution is calculated using steerable pyramids [19, 24]. For this application, a steerable pyramid of four levels was found to be sufficient. If the input image is color, then it is first converted to grayscale before pyramid computation. At each pyramid level, texture direction and strength at each pixel is calculated using the outputs of seven X–Y separable, steerable quadrature pair basis filters.

The separable basis set and interpolation functions for the second derivative of a Gaussian were implemented directly using the nine-tap formulation provided in Appendix H (Tables IV and VI) of [19]. The resulting basis is composed of three G_2 filters

to steer the second derivative of a Gaussian and four H_2 filters to steer the Hilbert transform of the second derivative of a Gaussian.

At each level in the pyramid, the output of these filters is combined to obtain a first-order approximation to the Fourier series for oriented energy $E_{G_2H_2}$ as a function of angle θ,

$$E_{G_2H_2} = C_1 + C_2\cos(2\theta) + C_3\sin(2\theta), \tag{12}$$

where the terms C_1, C_2, C_3 are as prescribed in [19], Appendix I.

Dominant orientation angle θ_d and the orientation strength m at a given pixel are calculated via the formulae

$$\theta_d = \frac{1}{2}\arg[C_2, C_3] \tag{13}$$

$$S_{\theta_d} = \sqrt{C_2^2 + C_3^2}. \tag{14}$$

Orientation histograms are then computed for each level in the pyramid. Each orientation histogram is quantized over $[-\frac{\pi}{2}, \frac{\pi}{2}]$. In the current implementation, there are 16 histogram bins, thus the number of bins allocated for direction information stored per subimage is 64 (4 levels · 16 bins/level). Each histogram is then normalized to have unit sum. Once computed, the histogram must be circularly blurred to obviate aliasing effects and to allow for "fuzzy" matching of histograms during image search [41].

In practice, there must be a lower bound placed on the accepted orientation strength allowed to contribute to the distribution. For the ImageRover implementation, all the points with the strength magnitude less than 0.005 are discarded and not counted in the overall direction histogram.

The orientation measure employed in ImageRover differs from that proposed by Gorkani and Picard [24]. While both systems utilize steerable pyramids to determine orientation strengths at multiple scales, there is a difference in how histograms are compared. In the system of Gorkani and Picard, histogram peaks are first extracted and then image similarity is computed in terms of peak-to-peak distances. In practice, histogram peaks can be difficult to extract and match reliably. In our system, histograms are compared directly via histogram distance, thereby avoiding problems with direct peak extraction and matching.

3.2.3. Dimensionality reduction. For each image digested, there are n visual statistics vectors computed. In the current implementation $n = 2 \times 6$ as we use two image analysis modules (color and texture) and compute the statistics over six image regions (the whole image plus five regions [46]). These visual statistics vectors can be combined to form an aggregate visual statistics vector $\mathbf{X}_{\text{vis}}$.

In ImageRover, it is assumed that the probability distribution of the visual statistics vectors is Gaussian and that the n subvectors are independent. This may only provide a fair approximation to the true distribution of image statistics; however, the assumption simplifies computation considerably and can alleviate the *empty space* problem [13]. Under such assumptions, the covariance matrix $\mathbf{\Sigma}$ is block diagonal and the diagonal elements $\mathbf{\Sigma}_i$ are the covariances of the subvectors $\mathbf{x}_i$. Similarly the mean μ_x is composed by the means $\mu_{\mathbf{x}_i}$ of the subvectors $\mathbf{x}_i$. Restated in equation form:

$$\mathbf{X}_{\text{vis}} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}; \quad \mu_x = \begin{bmatrix} \mu_{x_1} \\ \mu_{x_2} \\ \vdots \\ \mu_{x_n} \end{bmatrix}; \quad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & & & 0 \\ & \mathbf{\Sigma}_2 & & \\ & & \ddots & \\ 0 & & & \mathbf{\Sigma}_n \end{bmatrix}. \tag{15}$$

Note that the $\mathbf{\Sigma}_i$ and $\mu_{\mathbf{x}_i}$ can be precomputed on a large training set of images selected at random from the full database. In practice, we have found that a training set of 50,000 images taken at random from the full image database is sufficient to characterize the distribution. Assuming a diverse training set, there is no need to recompute them when new images are added to the system.

Given $\mathbf{\Sigma}_i$ and $\mu_{\mathbf{x}_i}$, it is then possible to reduce the dimensionality for each subspace via principal components analysis (PCA) [13, 20]. For the ith subspace, we compute the eigenvalues and eigenvectors of the subspace covariance matrix $\mathbf{\Sigma}_i$. The resulting eigenvalues and eigenvectors are sorted in decreasing order by eigenvalue. The eigenvectors ϕ_j describe the principal axes of the distribution and are stored as columns in the PCA transform matrix $\mathbf{\Phi}_i$. The associated eigenvalues λ_j describe the variances along each principal axes and are stored in the diagonal matrix $\mathbf{\Lambda}_i$.

The transform matrix $\mathbf{\Phi}_i$ decouples the degrees of freedom in the subspace. Once it is decoupled, it is possible to compute a truncated feature space that accounts for most of the covariance of the distribution. Although all eigenvectors are needed to represent the distribution exactly, only a small number of vectors are generally needed to encode samples in the distribution within a specified tolerance. In practice, the first k eigenvectors are used, such that k is chosen to represent the variance in the dataset within some error threshold τ. In our experiments setting $t = 0.1$ (10% error) resulted in a dimensionality reduction of over 85%.

Once eigenvalues and eigenvectors are computed, it is possible to compute a truncated basis that accounts for most of the covariance of the distribution and then project each vector $\mathbf{x}_i$ onto the new basis to obtain $\bar{\mathbf{x}}_i$,

$$\bar{\mathbf{x}}_i = \Lambda_i^{-1/2}\mathbf{\Phi}_i^T\left(\mathbf{x}_i - \mu_{\mathbf{x}_i}\right), \tag{16}$$

where the columns of $\mathbf{\Phi}_i$ are the principal eigenvectors of the distribution and the diagonal matrix Λ_i is composed of the corresponding eigenvalues.

Note that, once Λ_i, $\mathbf{\Phi}_i$, and $\mu_{\mathbf{x}_i}$ have been computed on a training set, there is no need to recompute them when new images are added to the system. The transformation (16) is computed for each image during the initialization of the system and the result is stored in the archive as a much lower dimensional vector $\bar{\mathbf{X}}_{\text{vis}} = [\bar{\mathbf{x}}_1\ \bar{\mathbf{x}}_2 \cdots \bar{\mathbf{x}}_n]^T$.

3.3. Integration of Textual and Visual Statistics

The global feature vector, representing the content of the image, is a composite of several subvectors: dimensionally reduced color histograms and orientation histograms for each of six overlapping image regions [46] and the LSI descriptor as described above. Different techniques have been proposed to combine different features in a global similarity measure [3, 11, 28, 34, 36, 38, 44]. In some of these techniques the user has to state explicitly the relevance of each feature, in others the system infers the feature weights based on relevance feedback information.

We used a linear combination of normalized Minkowsky distances. One reason for choosing this approach is its inherent compatibility with optimizations to spatial indexing techniques [4, 46]. Another reason is that a weighted combination of Minkowski distances satisfies the axioms for a metric space. The weights of the linear combination and the order of each Minkowsky metric are inferred by the system via the relevance feedback technique described in next section.

As the global distance is a linear combination of distances between the subvectors, normalization is needed to make these distances comparable. To analyze the statistical properties of these distances we randomly selected 50,000 pairs of images from our test database. For all pairs of images, the distances between their index vectors $(\mathbf{x}, \mathbf{y})$ were computed using each of the Minkowsky distance measures. We considered separately the distances between subvectors $(\mathbf{x}_i, \mathbf{y}_i)$. We then examined the distributions of these distances for each metric and for each subvector. As these distance distributions appeared to be approximately normally distributed we decided to normalize using a Gaussian model, along the lines of [28, 38, 44].

In practice, for each feature i and for each Minkowsky distance metric L_m we computed the mean $\mu_m^{(i)}$ and the variance $\sigma_m^{(i)}$ on the 50,000 random couples of vectors randomly selected from our database. Let $\mathbf{X}$ and $\mathbf{Y}$ denote image index vectors in a database and $\mathbf{x}_i$ and $\mathbf{y}_i$ denote subvectors corresponding to a particular feature. We define the normalized L_m distance between two subvectors,

$$\tilde{L}_m(\mathbf{x}_i, \mathbf{y}_i) = \max\left(0, \frac{L_m(\mathbf{x}_i, \mathbf{y}_i) - \mu_m^{(i)}}{\sigma_m^{(i)}} + \Delta\right), \qquad (17)$$

where as stated before the means and the variances are computed based on the probability distribution of the images contained in the database,

$$\mu_m^{(i)} = E[L_m(\mathbf{x}_i, \mathbf{y}_i)]; \quad \sigma_m^{(i)} = Var[L_m(\mathbf{x}_i, \mathbf{y}_i)], \qquad (18)$$

and Δ is the shift we give to the normalized Gaussian to mimic the original distribution. In our experiment we set $\Delta = 3$.

Note that the expected value $\mu_m^{(i)}$ can be computed off line over an entire database or a statistically significant subset of it. Moreover, if the database is reasonably large, we do not need to recompute this factor when new images are added to the archive.

3.4. Relevance Feedback

In query by example it is difficult to determine the appropriate combination of similarity measures for a particular search. Directly prompting users for weightings is problematic, since it may require that users grasp the technical details of the underlying representation. One way around this is to allow the users to provide example images; this keeps nettlesome image content parameters hidden from the user. ImageRover employs a novel approach to this relevance feedback problem.

Relevance feedback enables the user to iteratively refine a query via the specification of relevant items [45]. By including the user in the loop, better search performance can be achieved. Typically, the system returns a set of possible matches, and the user gives feedback by marking items as relevant or not relevant. Given user-specified relevant images, the system must then infer what combination of measures should be used.

Cox *et al.* [11] proposed a Bayesian relevance feedback framework for image retrieval. For every image in the database they compute the probability of it being a relevant image given the complete history of the query (i.e., the images displayed and the corresponding user actions). Experimental results for this technique are given on a small database and the high number of iterations required to find an image makes this technique inadequate for use on a very large database.

Rui *et al.* [44] proposed a method to formulate image queries in the same framework used for text retrieval. In practice each component of the image feature is considered like the term weight in the classical *information retrieval* formulation. They report quantitative evaluation on a very small texture database and show that on that data set relevance feedback improves retrieval performance.

In [36] the authors formulate the relevance feedback problem as a probability density estimation. Based on the user feedback (positive and negative) they present an algorithm to estimate feature densities. In this way they infer which features are of interest to the user. Their technique has been evaluated on Columbia [35] and VisTex databases and has been shown to perform slightly better than standard relevance feedback approaches adapted from *information retrieval*.

Other techniques [8] have also been proposed recently to determine the optimal similarity metrics to use.

Our system employs a relevance feedback algorithm that selects appropriate L_m Minkowski distance metrics on the fly. The algorithm determines the relative weightings of the individual features based on feedback images. This weighting thus varies depending upon the particular selections of the user.

Assume that the user has specified a set S of *relevant* images. The appropriate value of m for the ith subvector should minimize the mean distance between the relevant images. The order of the distance metric is determined as

$$m_i = \arg\min_m \eta_m^{(i)}, \qquad (19)$$

where

$$\eta_m^{(i)} = E[\tilde{L}_m(\mathbf{p}_i, \mathbf{q}_i)], \quad \mathbf{P}, \mathbf{Q} \in S, \tag{20}$$

and $\tilde{L}_m(\mathbf{x}, \mathbf{y})$ is the normalized Minkowski metric as defined in previous section.

Queries by multiple examples are implemented in the following way. First, the average query vector is computed for S. A k-nearest neighbor search of the image index then utilizes the weighted distance metric

$$\delta(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} w_i \tilde{L}_{m_i}(\mathbf{x}_i, \mathbf{y}_i), \tag{21}$$

where the w_i are *relevance weights*,

$$w_i = \frac{1}{\epsilon + \eta_m^{(i)}}. \tag{22}$$

The constant ϵ is included to prevent a particular characteristic or a particular region from giving too strong a bias to the query.

The resulting relevance feedback mechanism allows the user to perform queries by example based on more than one sample image. The user can collect the images he or she finds during the search, refining the result at each iteration. The main idea consists of giving more importance to the elements of the feature vectors with the lowest variances. These elements very likely represent the main features the user is interested in. Experimental results have confirmed this behavior.

Another advantage of our formulation is that it is not dependent on the particular features employed in representing the visual content of images. We can reasonably expect that the system will remain efficient as more image analysis modules are included, because the computational complexity will scale linearly with the number of features employed.

4. SYSTEM IMPLEMENTATION

We implemented a fully functional system to test our approach. For the experiments described in this paper, our database contained approximately 350,000 unique and valid images. Two images are considered unique if they have different URLs. An image is valid if both its width and height are greater than 64 pixels; images not satisfying this heuristic were discarded. In practice some of the documents and the images are duplicated as sometimes the same document or the same image appears with a different URL due to name aliasing.

To have a significant sampling of the images present on the WWW a list of links related to diverse topics was needed. For the experiments reported in this paper, we selected the pages reported in Table 2 as the starting point for our Web robots. A demo version of the resulting ImageRover index is available on line at http://www.cs.bu.edu/groups/ivc/ImageRover.

TABLE 2
Starting Points for Web Robot Image Collection

Looksmart.com category
Automotive
Automotive : Motorcycles
Hobbies : Modelmaking
Shopping
Reference & Education : Nature
Reference & Education : Magazines
Sport : Soccer
Sport : Fishing
Sport : Baseball
World : Travel : Guides
World : Entertainment : Celebrities
Yahoo category
Science : Astronomy : Pictures
Science : Biology : Zoology : Animals : Pictures
Recreation : Aviation : Pictures
Arts : Visual Arts : Photography : Underwater
Arts : Visual Arts : Photography : Photographers
Arts : Visual Arts : Photography : Nature and Wildlife
Recreation : Travel : Pictures
Computers and Internet : Multimedia : Pictures
Recreation : Sports : News and Media : Magazines
Companies : Arts and Crafts : Galleries
News and Media : Television : Cable : Networks : US

The current implementation of the system is not optimized for speed. The query server and the Web server run on an SGI Origin 200 with 4 R10000 180 MHz processors and 1 GB RAM. After dimensionality reduction the visual features are reduced to a vector of dimension around 200, so fewer than 500 floating point numbers (about 2 Kbyte) per image are required and keeping in memory 100,000 images requires approximately 200 Mbyte. As all the data can be kept in memory, a brute force search of the k-nearest neighbors takes around 3 s. In the case of the page zero we have to compute the LSI vector corresponding to the key words provided by the user. This is a simple vector by matrix product, and, even though the dimension is high, it takes less than 1 s.

4.1. User Interface

The user interacts with the system through a web browser. The user specifies a set of keywords; this set of keywords can be considered as a text document. An LSI index is computed for the keywords and used to match nearest neighbors in the subspace of all LSI vectors in our image database. This is *page zero*.

Once the user finds and marks one or more images to guide the search, the user can initiate a query with a click on the search button. Similar images (the number of returned images is a user chosen value) are then retrieved and shown to the user in decreasing similarity order. The user can then select other relevant images to guide the next search and/or deselect one or more of the query images and iterate the query. There is no limit either on

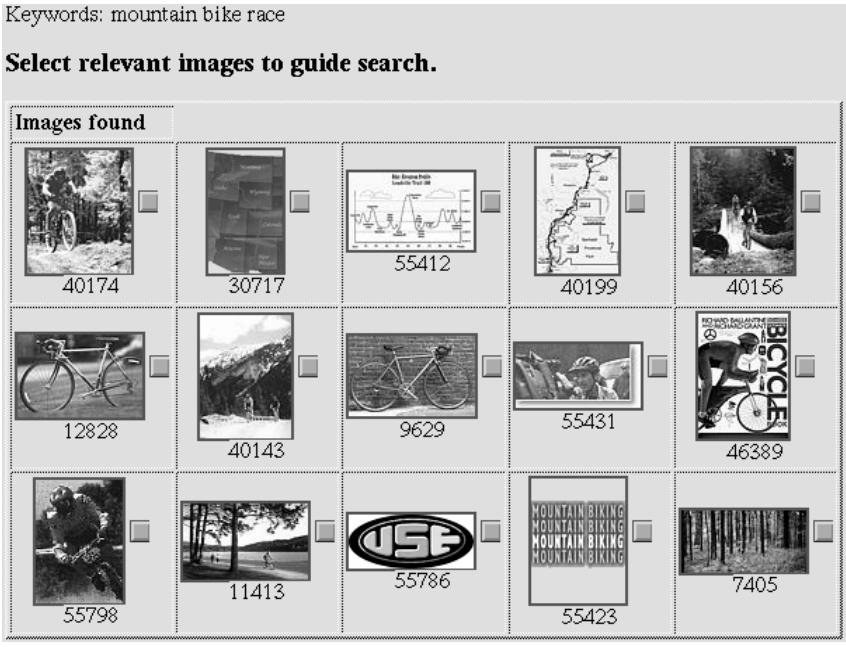

FIG. 1. Example *page zero* of ImageRover query given key words "mountain bike race."

the number of feedback iterations or on the number of example images employed.

4.2. Example Search

Figures 1–3 show an example search in our system. Figure 1 shows the first 15 matches in response to the query "mountain bike race." This is page zero. It is evident that several kinds of images are related to the same key words. By providing relevance feedback, the user can now narrow the search to images that share not only the same key words but also similar visual properties.

Figure 2 is a set of images found by the system using the relevance feedback. The top two images are the images selected by the user: images of cyclists racing together. The next three rows contain the retrieved 15 nearest neighbors. Images are displayed in similarity rank order, right to left, top to bottom. In this particular example, ImageRover ranked other racing photographs as closest to the user-provided examples. The other returned images not only share similar text cues, but also share similar color and texture distributions.

Similarly, Fig. 3 shows the output of the system given the same page zero, but different relevance feedback from the user. In this example, the user selected images of mountain bikes. The system then retrieved images that were relevant to the user's query. As can be seen, the use of visual features and relevance feedback is very useful in removing the ambiguity that is almost always present when using key words to retrieve images.

5. EXPERIMENTAL EVALUATION

The system has been tested with human subjects. The experiments were intended to evaluate the effectiveness of performance achieved through the combined use of visual and textual features. To evaluate quantitatively our system, the *target test paradigm* [11] was used. This technique evaluates how good a system is at finding a specific image in the database. Cox *et al.* [11] have pointed out that if a system performs well in a *target search* it is reasonable to think that it performs well also for the more useful *category search*, i.e., looking not for a particular image but for some class of images, though this has not been proven experimentally.

5.1. Experimental Setup

The system we used for the experiments was initially trained with a randomly selected subset of the collected data containing 58,908 images. The eigendecompositions for the LSI vectors and the visual features were performed only once and new images were inserted into the system only as a projection into the reduced feature space. Since the training set was selected so that it was representative of the documents available on the Web, and since the size of the training set was considerable, it seems reasonable to assume that retraining the system is not required. For instances where training is required, standard techniques for updating SVD-based indexing schemes have been reported in [9].

After the training step, we indexed a set of 10,000 images disjoint with the training set and stored them in the database. A subset of 100 images in the database were selected (randomly) to be retrieved. Two subjects were asked to find each of the images using our system, one at a time. The subjects used for the test are two of the authors. This implies that the performance reported is probably representative of what can be achieved by an expert user.

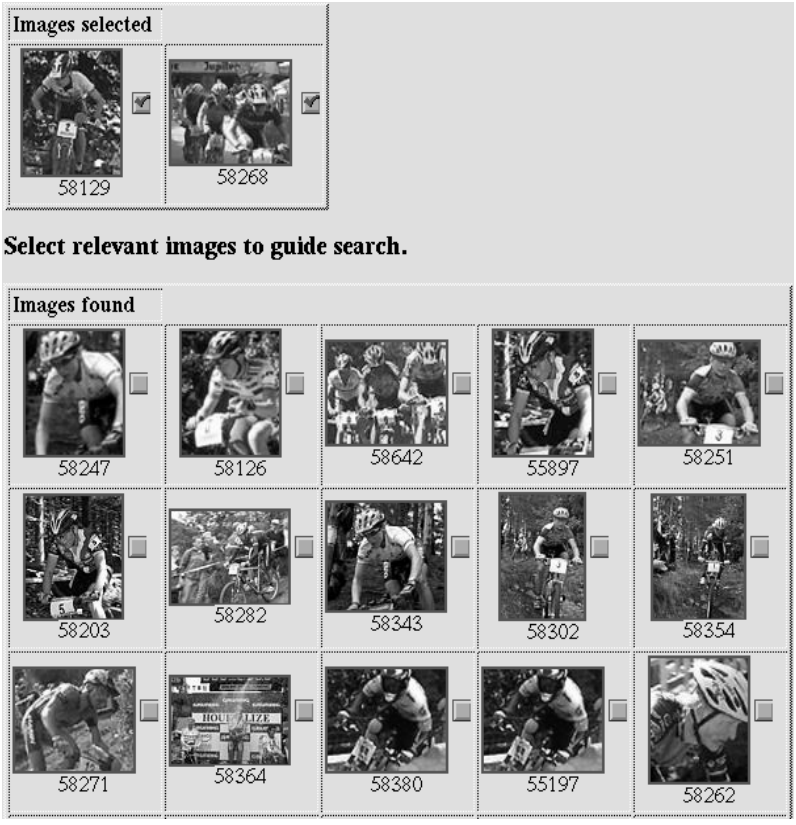

FIG. 2. Example of search using relevance feedback. The top two images are the *relevant* images selected by the user. The others are the response of the system.

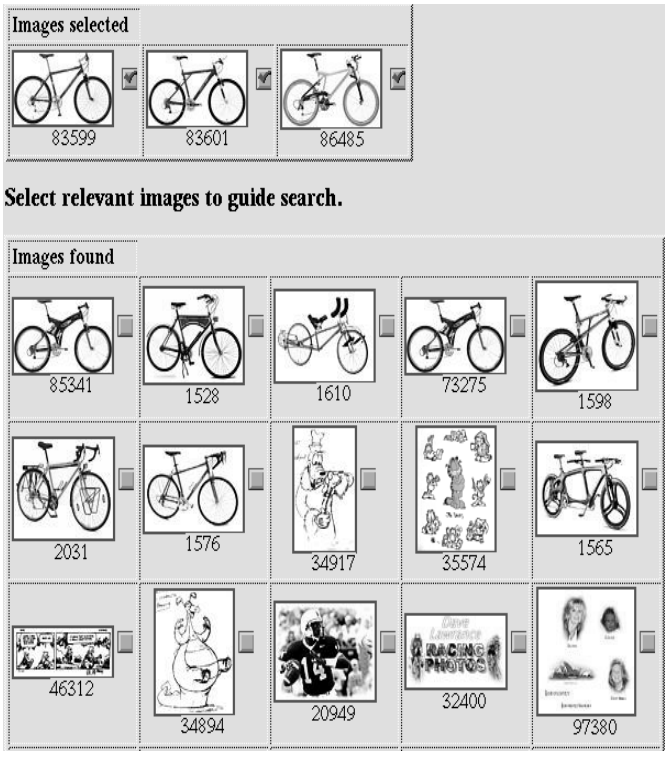

FIG. 3. Example of search using relevance feedback. The top three images are the *relevant* images selected by the user. The others are the response of the system.

The subjects were first presented with one of the 100 images displayed in a window on the computer screen. While the target image remained visible, the subjects could then formulate a query of the WWW image database via the user interface as described above. This process was repeated for all 100 images in the test set.

In practice, for each image, the subjects independently typed in a few keywords relevant to the target image to obtain a starting set of 100 images (page zero). In the current experiments the subjects have used four keywords on average per search. The subjects could then further refine the search through iterations of the relevance feedback mechanism. At each iteration, the system displayed the 100 top matches for the query. The resulting images were shown to the subject downsampled (the biggest dimension of the downsampled image was 128 and the other was such that the aspect ratio was unchanged) and ordered on the base of the matching score (decreasing left to right and then top to bottom).

A search was considered successful if the subjects could get the target image displayed in the top 100. A search was considered unsuccessful if the subject could not get the target image displayed in the top 100 within four iterations of relevance feedback. This limit on the number of feedback iterations was chosen to reflect the amount of time a typical user would be willing to devote in finding an image. The time the subject spent to consider the top 100 results was on average less than 30 s.

5.2. *Sensitivity to Database Size*

To measure how the system scales with the number of images archived, the experiments were repeated at various database sizes: 10,000, 30,000, 50,000, and 100,000 images, respectively. To evaluate search performance with respect to the types of feature vectors employed, multiple trials were conducted in which textual only, visual only, and combined textual and visual features were included into the indexing vector. In each set of trials, the subjects were asked to find the same randomly selected subset of 100 images. To avoid biasing of the subjects, due to an increased familiarity with the data set, we asked them to use the same keywords for generating page zero during each trial.

The average percentage of target images that subjects were able to successfully retrieve, is shown in Fig. 4. The graph depicts the percentage of images found as a function of the number of images contained in the test database. Note that the results included in the graphs also include the images which are not related to the surrounding text and hence carry no valuable LSI information. Out of the 100 images used in the test set, there were 44 such images. Assuming that such images are common on the Web, the results are expressed as the total number of images selected for search. The percentage of images that the subjects were able to retrieve decreases as the number of images increases but is still reasonable considering the database size. With a database size of 10,000, the retrieval success rate with combined cues was around 35%, degrading to about 18% as the database size increased to 100,000.

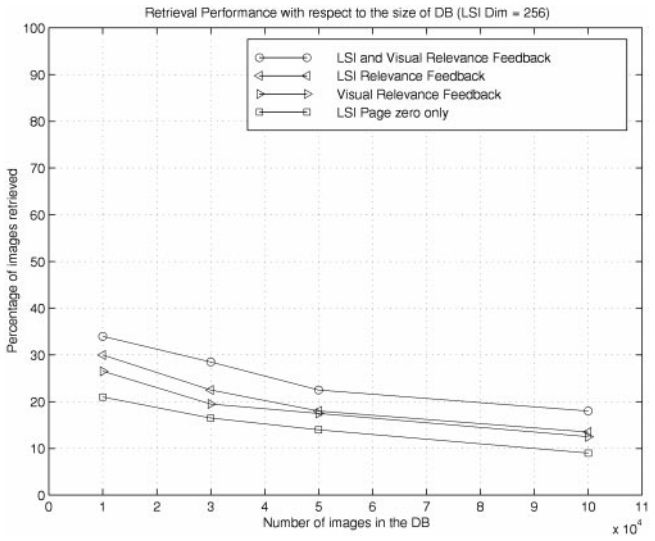

FIG. 4. Percentage of test images retrieved vs number of images in the database.

The lowest curve on the graph shows the percentage of images that subjects retrieved in page zero. The other curves show the performance when users were allowed to use relevance feedback. To determine the major contributor in performance improvement, experimental trials were conducted in which the system employed visual statistics only, LSI only, and LSI and visual statistics combined in relevance feedback. As can be seen, relevance feedback using combined features offers a significant performance improvement over using either of the features separately.

5.3. *Sensitivity to LSI Dimension*

In a separate set of trials with a different test set, the sensitivity to LSI dimension was tested for a database of 10,000 images at six levels: $dim(LSI) = 64$, 128, 256, 384, 512, 768. The tests were done by searching for 50 random images. Again as in the previous case, 18 images could not be described by key words. Results of this experiment are shown in Fig. 5. The graph shows how subjects' success rates improved when a higher LSI dimension was employed. The steepest improvement in performance was achieved with LSI dimension of 256. After that, performance increased more slowly with increasing LSI dimension. This is consistent with results reported in [14], where it was observed that as LSI dimension increases the performance curve flattens out and then actually drops off slightly (due to noise).

6. DISCUSSION

It is evident from our experiments that based on text features only, it is sometimes possible for subjects to find the target image in page zero. If the target image did not appear in page zero, then the subjects were almost twice as likely to find that image when

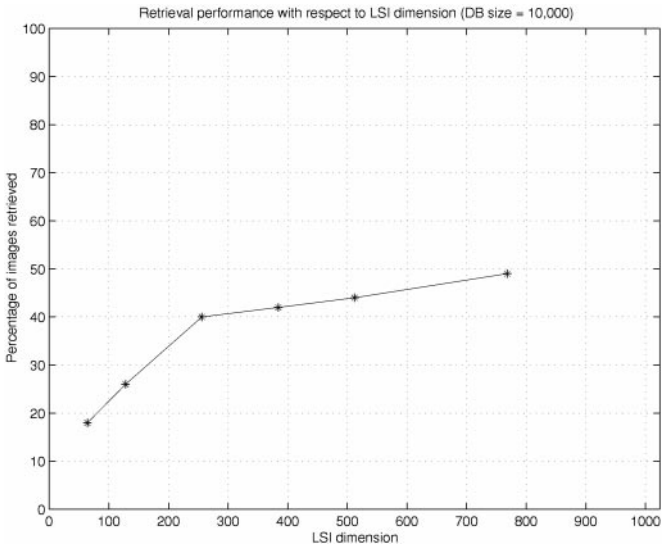

FIG. 5. Search performance with respect to LSI dimension. Relevance feedback was determined using combined visual and textual cues.

textual and visual features were combined in relevance feedback rather than used individually.

In most of the queries, the combined textual and visual refinements gave improved performance over visual only or textual only. However, in some cases, the visual features can confuse the system and negatively affect the performance of the system.

For example, if the user is looking for an image of "a man drinking" the visual features we use (color and orientation histograms) are of no help in the query refinement process as different images of men drinking can have totally different color and orientation histograms. In these cases, the textual information is more likely to encode the relevant information and the visual features can accidentally confuse the system; e.g., if the example images happen to have very similar orientation histograms in one of the subregions, this feature will be erroneously considered by the system as the feature that the user is interested in. Perhaps this problem can be addressed through the use of blob-based search methods [6] coupled with face detection and recognition.

In contrast, let us assume that the user is looking for "a man playing soccer." In this case the visual features are very likely to encode relevant information (we expect a green smooth background at least in the corner subregions) and they are very useful in the query refinement step. Note that in both cases we were interested in "actions," but only in one of the cases can visual features help during the refinement step.

Similarly when looking for "things" often the visual features can improve the retrieval performance, but sometimes they are of no help and confuse the system. For example, consider a user searching for a picture of "the surface of Mars." Using textual features, the user easily gets images of the surface and of the whole planet. Using visual features, it is very simple to steer the system toward the images the user is interested in. In contrast, if the user is looking for "a man wearing a space suit," the visual features are not very likely to help as different space suits and different environments can lead to totally different visual characteristics. The use of additional and more sophisticated visual features could definitely increase the number of cases where the visual cues can help in the query refinement. We leave this for future work.

In the experiments, it was also observed that the average number of relevance feedback steps required to steer the system toward the wanted image was independent of the database size and LSI dimensions used. Whenever an image was successfully retrieved via relevance feedback, it was found in 1.9 feedback iterations on average. This means that if the user cannot find the wanted image in a few steps he or she may not be able to find the image at all.

In both experiments, the retrieval for database size of 10,000 images and 256 LSI dimension was consistently around 34–40%. This further confirms the consistency of retrieval performance of our system independent of the set of images chosen for search.

We performed an analysis to characterize why subjects were unable to retrieve certain images from the index. We observed that often, the inability to find a particular image is due to a lack of correlation between the image content and the surrounding text, e.g., banners or specific logos. It was also found that in some cases, there are Web pages like photo galleries which contain several images and/or very little text. Finally, in some cases, subjects were simply unable to form a page zero query, because it was difficult to describe the content of a particular test image with words. Examples of such images are shown in Fig. 6.

Some failed searches using the system may also be attributed to the presence of ambiguous words in the documents. It was observed that if words like "*image*," "*photo*," etc. were given for *page zero*, then the system retrieved several images from photo galleries and did not provide any good starting images if the user was going to search for satellite images. Similar problems are also encountered if the document containing images does not contain any text.

We experimentally found that a 256-dimensional LSI vector leads to good results for our data set, despite the breadth of the subject matter of documents included in the LSI training. The optimal LSI dimension may also be obtained using an MDL framework [53].

Figure 5 shows that the net increase in the retrieval performance drops significantly after LSI dimension 256. The graphs

FIG. 6. Examples of images that subjects could not describe with words.

are still increasing but this increase may be compensated for by using more key words. In experiments reported in [14], the average number of key words for a search on a database of around 5000 documents was 10. Increase in the number of key words is expected to overcome this problem by a stronger match. Also, the dictionary used to create the word histograms is *ad hoc* and was chosen according to the frequencies of various words in a set of representative documents.

We expect that even in a very large database (with millions of images) it will be possible to retrieve specific images in many cases using our technique. In other cases, the user will be able to find several relevant images. Using all the features it is possible to converge the search faster (1.9 steps as reported by our experiments) as opposed to using individual features which may require several steps before finding the actual image.

7. SUMMARY

We proposed a general framework for the indexing of images with associated text and implemented a prototype WWW image search engine to evaluate the performance of our approach. We found that the maximum performance was achieved when both visual and textual information were used in the relevance feedback framework. The experiments showed that when both textual and visual information are used in the refinement, results are significantly better than those achievable using visual only or textual only information.

In our experience, the use of LSI in generating page zero has certain advantages over the classical keyword vector method [45] employed in most WWW text search engines and some image search engines [1, 18, 48]. In particular, LSI implicitly addresses problems with synonyms, word sense, lexical matching, and term omission. This is a distinct advantage over the keyword vector approach, in that user-specified terms do not need to be an "exact match." On the other hand, due to dimensionality reduction, LSI can sometimes lack the specificity provided by the keyword vector method.

For the sake of demonstrating our approach, we chose to utilize only two existing methods for encoding visual statistics. The choice of the optimal image statistics is considered beyond the scope of this paper. Our purpose was instead to formulate and evaluate a technique to integrate textual and visual information for image retrieval. We suspect that the demonstrated power of our unified approach can be easily extended to include more powerful visual features in the future.

ACKNOWLEDGMENTS

This work was supported in part through National Science Foundation Grants IIS–9624168 and EIA–9623865.

REFERENCES

1. AltaVista, A/V Photo Finder, *available at* http://jump.altavista.com/image.
2. AltaVista, Corbis Picture Experience, *available at* http://safari.altavista.digital.com.
3. E. Ardizzone and M. La Cascia, Automatic video database indexing and retrieval, *Multimedia Tools Appl.* **4**(1), 1997, 29–56.
4. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, An optimal algorithm for approximate nearest neighbour searching in fixed dimensions, in *Proc. of 5th ACM SIAM Symposium on Discrete Algorithms, 1994*, pp. 573–582.
5. J. R. Bach, S. Paul, and R. Jain, A visual information managment system for the interactive retrieval of faces, *IEEE Trans. Knowledge Data Eng.* **5**(4), 1993, 619–628.
6. S. Belongie, C. Carson, H. Greenspan, and J. Malik, Color- and texture-based image segmentation using the expectation-maximization algorithm and its application to content-based image retrieval, in *IEEE International Conference on Computer Vision, 1998*, pp. 675–682.
7. M. W. Berry and S. T. Dumais, *Using Linear Algebra for Intelligent Information Retrieval*, Technical Report UT–CS–94–270, Computer Science Department, University of Tennessee, December 1994.
8. B. Bhanu, J. Peng, and S. Qing, Learning feature relevance and similarity metrics in image databases, in *Proc. of IEEE International Workshop on Content-Based Access of Image and Video Libraries, June 1998*, pp. 14–18.
9. G. Brien, *Information Management Tools for Updating an SVD-Encoded Indexing Scheme*, Technical Report TR UT–CS–94–259, University of Tennessee, 1994.
10. P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.
11. I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yanilos, PicHunter: Bayesian relevance feedback for image retrieval, in *Proc. of International Conference on Pattern Recognition, 1996*, pp. C:361–C:369.
12. S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, Indexing by latent semantic analysis, *J. Soc. Inform. Sci.* **41**(6), 1990, 391–407.
13. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
14. S. T. Dumais, Improving the retrieval of information from external sources, *Behav. Res. Meth. Inst. Comput.* **23**(2), 1991, 229–236.
15. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: The QBIC system, *IEEE Comput.* **28**(9), 1995, 23–30.
16. D. A. Forsyth, J. Malik, M. M. Fleck, H. Greenspan, T. Leung, S. Belongie, C. Carson, and C. Bregler, *Finding Pictures of Objects in Large Collections of Images*, Technical Report CSD–96–905, University of California, Berkeley, 1996.
17. J. M. Francos, A. Zvi Meiri, and B. Porat, A unified texture model based on a 2-D wold like decomposition, *IEEE Trans. Signal Procces.* **41**(8), 1993, 2665–2678.
18. C. Frankel, M. Swain, and V. Athitsos, *Webseer: An Image Search Engine for the World Wide Web*, Technical Report 96–14, Computer Science Dept., University of Chicago, 1996.
19. W. Freeman and E. H. Adelson, The design and use of steerable filters, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(9), 1991, 891–906.
20. K. Fukunaga and W. Koontz, Application of the Karhunen-Loeve expansion to feature selection and ordering, *IEEE Trans. Comput.* **19**(4), 1970, 311–318.
21. U. Gargi and R. Kasturi, An evaluation of color histogram based methods in video indexing, in *Proc. of International Workshop on Image Databases and Multimedia Search, 1996*, pp. 75–82.
22. T. Gevers and A. W. M. Smeulders, A comparative study of several color models for color image invariant retrieval, in *Proc. of International Workshop on Image Databases and Multimedia Search, 1996*, pp. 17–26.

23. T. Gevers and A. W. M. Smeulders, Pictoseek: A content-based image search engine for the WWW, in *Proc. of International Conference on Visual Information Systems, 1997*, pp. 93–100.
24. M. M. Gorkani and R. W. Picard, Texture orientation for sorting photos at a glance, in *Proc. of IEEE International Conference on Pattern Recognition, 1994*, Vol. 1, pp. 459–464.
25. A. Gupta, *Visual Information Retrieval Technology: A Virage Perspective*, Technical Report, Virage, Inc., 177 Bovet Rd, Suite 540, San Mateo CA, 1996.
26. J. Hafner, H. Sawney, W. Equitz, M. Flickner, and W. Niblack, Efficient color histogram indexing for quadratic form distance functions, *IEEE Trans. Pattern Anal. Mach. Intell.* **1**(7), 1995, 729–736.
27. R. W. G. Hunt, *Measuring Color*, Wiley, New York, 1989.
28. A. K. Jain and A. Vailaya, *Shape-Based Retrieval: A Case Study with Trademark Image Databases*, Technical Report MSU–CPS–97–31, Michigan State University, 1997.
29. M. La Cascia and E. Ardizzone, JACOB: Just a content-based query system for video databases, in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, 1996*, pp. 1216–1219.

29a. M. La Cascia, S. Sethi, and S. Sclaroff, Combining textual and visual cues for content-based image retrieval on the world wide web, in *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries, 1998*, pp. 24–28.

30. M. Lew, K. Lempinen, and N. Huijsmans, Webcrawling using sketches, in *Proc. of International Conference on Visual Information Systems, 1997*, pp. 77–84.
31. F. Liu and R. W. Picard, Periodocity, directionality and randomness: Wold features for image modeling and retrieval, *IEEE Trans. Pattern Anal. Mach. Intell.* **18**(7), 1996, 722–733.
32. Lycos, Pictures and sounds, *available at* http://www.lycos.com/picturethis.
33. R. Mehrotra and J. E. Gary, Similar-shape retrieval in shape data management, *IEEE Comput.* **28**(9), 1995, 57–62.
34. T. P. Minka and R. W. Picard, Interactive learning using a society of models, *Pattern Recog.* **30**(4), 1997, 565–581.
35. H. Murase and S. Nayar, Visual learning and recognition of 3-D objects from appearance, *Int. J. Comput. Vision* **14**(1), 1995, 5–24.
36. C. Nastar, M. Mitschke, and C. Meilhac, Efficient query refinement for image retrieval, in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1998*, pp. 547–552.
37. V. E. Ogle and M. Stonebraker, Chabot: Retrieval from a relational database of images, *IEEE Comput.* **28**(9), 1995, 40–48.
38. M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and T. S. Huang, Supporting similarity queries in MARS, in *Proc. ACM Multimedia, 1997*, pp. 403–414.
39. G. Pass, R. Zabih, and J. Miller, Comparing images using color coherence vectors, in *Proc. ACM Multimedia, 1996*, pp. 65–74.
40. A. Pentland, R. W. Picard, and S. Sclaroff, Photobook: Tools for content-based manipulation of image databases, *Int. J. Comput. Vision* **18**(3), 1996, 233–254.
41. R. W. Picard and T. P. Minka, Vision texture for annotation, *Multimedia Sys.* **3**(3), 1995, 3–14.
42. A. R. Rao and G. L. Lohse, Towards a texture naming system: Identifying relevant dimensions of texture, in *Proc. of IEEE Conference on Visualization, 1993*, pp. 220–227.
43. N. C. Rowe and B. Frew, Distinguishing the picture captions on the World Wide Web, in *Proc. of the 2nd ACM International Conference on Digital Libraries, 1997*, pp. 263–263.
44. Y. Rui, T. S. Huang, S. Mehrotra, and M. Ortega, A relevance feedback architecture for content-based multimedia information retrieval system, in *Proc. of IEEE International Workshop on Content-Based Access of Image and Video Libraries, 1997*, pp. 109–116.
45. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw–Hill, New York, 1989.
46. S. Sclaroff, L. Taycher, and M. La Cascia, ImageRover: A content-based image browser for the world wide web, in *Proc. of IEEE International Workshop on Content-Based Access of Image and Video Libraries, 1997*, pp. 2–9.

46a. S. Sclaroff, *World Wide Web Image Search Engines*, Technical report B.U. TR95-016, Boston University, 1995. [Available at http://www.cs.bu.edu/techreports]

47. J. R. Smith and S.-F. Chang, Tools and techniques for color image retrieval, in *Proc. of IS&T SPIE: Storage and Retrieval Image and Video Database IV, 1996*, pp. 426–437.
48. J. R. Smith and S.-F. Chang, Visually searching the web for content, *IEEE Multimedia* **4**(3), 1997, 12–20.
49. R. K. Srihari, Automatic indexing and content-based retrieval of captioned images, *IEEE Comput.* **28**(9), 1995, 49–56.
50. M. Swain and D. Ballard, Color indexing, *Int. J. Comput. Vision* **7**(1), 1991, 11–32.
51. H. Tamura, S. Mori, and T. Yamawaki, Textural features corresponding to visual perception, *IEEE Trans. Syst. Man Cyber.* **8**(6), 1978, 460– 473.
52. Yahoo, Image surfer, *available at* http://isurf.yahoo.com.
53. H. Zha, *A Subspace-Based Model for Information Retrieval with Applications in Latent Semantic Indexing*, Technical Report CSE–98–002, Pennsylvania State University, 1998.