

Image Digestion and Relevance Feedback in the ImageRover WWW Search Engine

Leonid Taycher, Marco La Cascia, and Stan Sclaroff
Computer Science Department
Boston University
Boston, MA 02215

Abstract

ImageRover is a search by image content navigation tool for the world wide web. The staggering size of the WWW dictates certain strategies and algorithms for image collection, digestion, indexing, and user interface. This paper describes two key components of the ImageRover strategy: image digestion and relevance feedback. Image digestion occurs during image collection; robots digest the images they find, computing image decompositions and indices, and storing this extracted information in vector form for searches based on image content. Relevance feedback occurs during index search; users can iteratively guide the search through the selection of relevant examples. ImageRover employs a novel relevance feedback algorithm to determine the weighted combination of image similarity metrics appropriate for a particular query. ImageRover is available and running on the web site.

1 Introduction

For some time now, there have been “spiders,” “worms,” and other search engines crawling the World Wide Web (WWW), collecting index information about the documents they find. Sadly, most WWW search engines are blind to image content. While the WWW is a huge distributed multimedia database – combining text with images, video, graphics, and sound – existing WWW search tools are limited to text-based queries.

Image search engines could be employed in many applications; *e.g.*, on-line catalogs of consumer goods and services, museums, libraries, and medical or other scientific data collections. Such engines might also be useful in erotica-on-demand, image copyright enforcement, forensics and intelligence gathering. Lastly, such a web image robot would be useful to machine vision researchers studying image databases, since it could provide a very large testbed for image database indexing methods.

Given the number of unsolved problems in image understanding building a image search engine seems overly ambitious. One possible solution to this indexing problem is to use a traditional text-based approach. The strategy is to ex-

tract keywords automatically from HTML documents containing the image, using them as indices for image search. Variations of this approach are employed in Yahoo's Image Surfer, Lycos WebSeek[1].

The WebSeer search engine [2] supplements keywords with extracted information about images: grayscale *vs.* color, graphic *vs.* photo, image dimensions, file type and size, file date. In addition, their system includes a face detector that counts the number of faces and computes the largest face size. This information is stored as additional fields in a standard text database.

Unfortunately, the text-based approach is fraught with problems. In most images there are literally hundreds of objects that could be referenced, and each object has a long list of attributes. In addition, there are many image properties that defy description in words: textures, composition, unknown objects, etc. The old saying “a picture is worth a thousand words” is an understatement.

Clearly the best strategy is to let the images speak for themselves. It is imperative that we provide methods for users to search directly on image content. This strategy is adopted in ImageRover, the system described in this paper. The goal is to provide an arsenal of decompositions and discriminants that can be precomputed for images: color histograms, edge orientation histograms, texture measures, shape invariants, eigendecompositions, *etc.* In ImageRover, this computation of image measures is called *image digestion*.

In general, it is difficult for users to directly specify the weighted combination of measures needed for a particular query. This might require understanding the underlying image analysis algorithms. An alternative is to allow users to specify these weightings implicitly via a visual interface known as *query by example*[3]. Users can iteratively refine the search through the selection of more relevant example images. Using this *relevance feedback*, ImageRover then employs a novel relevance feedback algorithm to determine the weighted combination of image similarity metrics appropriate for the query.

The next section gives an overview of the ImageRover system architecture. After that, image digestion and relevance feedback algorithms will be described in detail.

2 ImageRover System Overview

Gathering 30 million images has the potential to take many years with a single-threaded robot on a single computer. We address this problem by employing a fleet of robots distributed across many machines. Experiments indicate that our framework can allow a modest fleet of 32 robots to collect and process over one million images monthly. For a detailed description of the ImageRover robot system architecture, readers are referred to [4].

As images are gathered, the robot then needs to digest each image, extracting the needed image statistics and decompositions. Each image digestion module processes an input stream of image URLs. Processing begins with translating the image file format (*e.g.*, GIF, TIFF, JPEG) to the internal format, and performing color transformations. A reduced resolution image thumbnail is computed for use as an icon during search.

2.1 Image Digestion

With preprocessing completed, the digester then executes M image analysis submodules that calculate information about the distributions of color, texture, orientation, faces, or other properties of the image. The image analysis algorithms used in ImageRover are described in Sec. 3.

Each digestion submodule computes distributions over N subimages. In the current implementation, $N = 6$; distributions are calculated over the whole image and over five image subregions: center, upper right, upper left, lower right, lower left.

The resulting distribution information is then stored in vector form, with each of the modules contributing subvectors to an image index vector \mathbf{X} . Given M modules and N subimages, the image index vector will have $n = M \times N$ subvectors:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \quad (1)$$

The image indexing vectors stored by the robots have rather high dimension. As pointed out by White and Jain [5], the data has *intrinsic dimension* that is significantly less than this. Furthermore, the distribution of samples may not be distributed uniformly across all dimensions. As a preliminary step, it is therefore useful to perform a dimensionality reduction via a principal components analysis (PCA) for each of the subvector spaces [4].

Using the truncated basis, each original image index subvectors \mathbf{x}_i undergo the dimensionality reducing transform, producing a reduced vector $\bar{\mathbf{x}}_i$. This transform is performed once for all vectors in the database as a precomputation, yielding a dimensionality reduced image feature

vectors of the form:

$$\bar{\mathbf{X}} = \begin{pmatrix} \bar{\mathbf{x}}_1 \\ \vdots \\ \bar{\mathbf{x}}_n \end{pmatrix} \quad (2)$$

This transform has the dual effect of 1.) concentrating the variance in a relatively small number of dimensions, and 2.) normalizing the principal directions by their inverse principal standard deviations, thus spreading samples more evenly within the k -dimensional space.

2.2 Image Index

The image index subsystem is based on a client-server architecture. At startup, the server first performs a dimensionality reduction, and then builds an optimized k -d tree[6]. Once initialized, the index server runs as a process separate from the database query server, possibly on a different computer. For each query, a client connects to the server to send the query data and then waits for the resulting k nearest neighbors. The server performs the query and returns the results to the client.

To obtain better indexing performance, ImageRover employs an approximation algorithm for k -d search[7, 5, 4]. Experimental evaluation of ImageRover's indexing strategy has shown approximation cuts indexing time by nearly two orders of magnitude without significant loss in retrieval accuracy [4].

2.3 User Interface

ImageRover employs the query by example paradigm[3, 8, 9]. To get the search going, a set of randomly selected images are shown to the user. The user can ask the system for another set of random images, or he/she can mark example "relevant" images from those presented. The relevance feedback algorithm is detailed in Sec. 4

An example ImageRover search is shown in Fig. 1. The query image(s) are also shown in the top of the screen. Similar images (the number of returned images is a user chosen value) are then retrieved and shown to the user in decreasing similarity order. This gives users an opportunity to see the collection of example images used so far. The user can then select other relevant images to guide next search and/or unselect one or more of the query images and iterate the query. There is no limit to the number of iterations in providing relevance feedback, nor in the number of example images.

Once the user finds and marks images to guide the search, the user can initiate a query with a click on the search button. The system usually retrieves relevant images and false matches, the user checks the images that are more relevant with respect to what he/she is looking for and



Figure 1. Example search for pictures of sports teams, based on relevance feedback from the user. The user-selected relevant images appear in the upper row (two example images of soccer team photos). 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 five more sport team photographs as closest to the user-provided examples. The other returned images share similar color and texture distributions.

reiterates the query until desired images are found. During a search, the user can unselect one or more of the example images and check some other relevant images.

In this example, the user was searching for images of sports teams. The user first selected one picture of a soccer team. The search iterated twice, with the user providing relevance feedback by marking another sports team image. The user-selected relevant images appear in the upper row of the image in the figure. The next three rows contain the

retrieved 15 nearest neighbors. In this particular example, ImageRover ranked five more sports team photographs as closest to the user-provided examples. The remaining images share similar color and orientation distributions.

Due to space limitations, it is difficult to include more than one example of image search in the ImageRover system. Readers are therefore invited to visit the ImageRover WWW site to try the system: <http://www.cs.bu.edu/groups/ivc/ImageRover/>.

3 Image Digestion Modules

At this writing there are four image analysis submodules fully-implemented in our system: a color module, and three texture modules computing the Wold features of periodicity, directionality, and randomness [10]. Efforts are underway to expand the system to include face detection and description using eigenfaces [11, 12], and text cue vectors extracted via latent semantic indexing (LSI) [13, 14] on the text surrounding the image in an HTML document.

3.1 Color

Color distributions are calculated as follows. Image color histograms are computed in the CIE $L^*u^*v^*$ color space, which has been shown to correspond closely to the human perception of color [15].

To transform a point from RGB to $L^*u^*v^*$ color space, it is first transformed into CIE XYZ space:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 1.116 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (3)$$

where the conversion matrix is for CIE Illuminant C (overcast sky at noon).

The $L^*u^*v^*$ values are then calculated as:

$$L^* = \begin{cases} 25 * (100 * \frac{Y}{Y_0})^{\frac{1}{3}} - 16 & \text{if } \frac{Y}{Y_0} \geq 0.008856 \\ 903.3 \frac{Y}{Y_0} & \text{otherwise} \end{cases} \quad (4)$$

$$u^* = 13L^*(u' - u'_0) \quad (5)$$

$$v^* = 13L^*(v' - v'_0) \quad (6)$$

$$u' = \frac{4X}{X + 15Y + 3Z} \quad (7)$$

$$v' = \frac{9Y}{X + 15Y + 3Z} \quad (8)$$

where the reference values are $(X_0, Y_0, Z_0) = (0.981, 1.000, 1.820)$, and $(u'_0, v'_0) = (0.1830, 0.4198)$, for white under CIE Illuminant C (Overcast sky at noon).

For each of the subimages, the color distribution is then calculated using the histogram method [16]. 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 Texture Orientation

The texture orientation distribution is calculated using steerable pyramids [17, 18]. 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 [17]. The resulting basis is comprised 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 :

$$E_{G_2H_2} = C_1 + C_2 \cos(2\theta) + C_3 \sin(2\theta), \quad (9)$$

where the terms C_1, C_2, C_3 are as prescribed in [17], Appendix I.

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

$$\theta_d = \frac{1}{2} \arg [C_2, C_3] \quad (10)$$

$$S_{\theta_d} = \sqrt{C_2^2 + C_3^2}. \quad (11)$$

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 [19].

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

The orientation measure employed in ImageRover differs from that proposed by Gorkani and Picard [18]. 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 ImageRover, histograms are compared directly via histogram distance, thereby avoiding problems with direct peak extraction and matching.

3.3 Texture Harmonic Structure

The distribution of harmonic features is computed as a histogram of the harmonic peak magnitudes extracted from the DFT magnitude image [10]. Since it is based on the Fourier spectrum magnitude, this harmonic structure measure offers the useful property of spatial shift-invariance.

In practice, each subimage is first converted to grayscale, normalized to zero mean, and its discrete Fourier transform magnitude image is computed. Next, a list of peak locations and magnitudes is computed. Peaks are defined as the local maxima of the DFT magnitudes, and are found by searching a 5×5 neighborhood of each sample in the DFT magnitude image.

The resulting peak list is then cleaned up by deleting all peaks which do not have corresponding harmonic peaks (peaks of the same angular direction but different frequency). Following [10], a peak is considered harmonic if it is either: 1.) fundamental, *i.e.*, its frequency can be used to linearly express frequencies of other peaks, or 2.) harmonic, *i.e.*, its frequency can be linearly expressed as a combination of frequencies of fundamental peaks.

Lists of peaks are computed at multiple scales. This is accomplished by first computing a Gaussian pyramid for the input image, and then analyzing the texture harmonic structure in the DFT. For this application, a pyramid of four levels was found to be sufficient.

For each level in the pyramid, a direction histogram of the harmonic peak magnitudes is then computed. Direction 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 harmonic information stored per subimage is 64 (4 levels * 16 bins/ level). Finally, each histogram is normalized to have unit sum and then circularly smoothed.

In the method proposed by Liu and Picard[10], harmonic peaks are matched directly by using an inter-peak distance heuristic. The method proposed here is different, in that a histogram of the harmonic peak magnitudes is computed. Images can then be compared in terms of histogram distance between normalized histograms. This histogram-based approach offers the advantage that it accounts for both the strength and direction of harmonic structure. In addition, the method offers robustness to changes in scale.

3.4 Texture Randomness

The ‘‘randomness’’ analysis module captures information about the indeterministic component of texture. The image is modeled in terms of the coefficients for a second order symmetric multiresolution simultaneous autoregressive (MRSAR) process [20, 10]. In the current implementation, color images are converted to grayscale before MRSAR coefficients are computed.

The value of the pixel $p_{i,j}$, at image position (i, j) is modeled as linear function of neighboring pixels:

$$p_{i,j} = c_1(p_{i-l,j} + p_{i+l,j}) + c_2(p_{i,j-l} + p_{i,j+l}) + c_3(p_{i-l,j-l} + p_{i+l,j+l}) + c_4(p_{i-l,j+l} + p_{i+l,j-l}), \quad (12)$$

where c_i are the coefficients, and l determines the resolution of the pixel neighborhood employed. It is convenient to write Eq. 12 as the dot product between a pixel vector and a coefficient vector: $p_{i,j} = \mathbf{P}_{i,j}^T \mathbf{c}$.

The solution for MRSAR coefficients at the pixel $p_{i,j}$ is under-determined; therefore, coefficients for $p_{i,j}$ are estimated over an $n \times n$ pixel window centered at i, j . In ImageRover, the four coefficients and least squares error (LSE) at each pixel are estimated via least squares fitting, with an estimation window dimension $n = 21$. This estimation process is repeated for each pixel within an image. The pixel-wise estimates are utilized to determine the mean coefficient vector and mean LSE for the whole image.

As described in Sec. 3, each digestion module computes distributions separately over N subimages. This implies that a five-dimensional vector of MRSAR coefficients and LSE is computed for each subimage. Furthermore, these features are computed at three resolutions $l = \{2, 3, 4\}$, yielding 15 values per subimage.

4 Relevance Feedback

Relevance feedback enables the user to iteratively refine a query via the specification of relevant items[21]. 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. The ImageRover system employs a relevance feedback algorithm that selects appropriate L_m Minkowski distance metrics on the fly. The formulation of this algorithm is as follows.

Let $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ denote image index vectors in a database. Let $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{y}}_i$ denote subvectors corresponding to the output of a particular image analysis module for a particular region in the image (as described in Sec. 3).

We define the normalized L_m distance between two subvectors:

$$\tilde{L}_m(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i) = \frac{L_m(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)}{\mu_m^{(i)}} \quad (13)$$

where the normalization factor is computed based on the probability distribution of the images contained in the database,

$$\mu_m^{(i)} = E[L_m(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)]. \quad (14)$$

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 don't need to recompute this factor when new images are added to the archive.

It is difficult to determine in advance which \tilde{L}_m distance metric is best suited for a particular similarity detection task [22]. Therefore, our system selects the appropriate \tilde{L}_m metric each time a query is made, based on relevance feedback from the user. Furthermore, instead of using the same metric for each image region and image measure, we allow selection of appropriate metrics for each of the various image index subvectors.

Assume that the user has specified a set S of *relevant* images. The appropriate value of m for the i^{th} subvector should minimize the mean distance between the relevant images. The dimension of the distance metric is determined as follows:

$$m_i = \arg \min_m \eta_m^{(i)} \quad (15)$$

where

$$\eta_m^{(i)} = E[\tilde{L}_m(\bar{\mathbf{p}}_i, \bar{\mathbf{q}}_i)], \quad \bar{\mathbf{P}}, \bar{\mathbf{Q}} \in S. \quad (16)$$

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

$$\delta(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) = \sum_{i=1}^n w_i \tilde{L}_{m_i}(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i), \quad (17)$$

where the w_i are *relevance weights*

$$w_i = \frac{1}{\epsilon + \eta_m^{(i)}}. \quad (18)$$

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

5 Summary

ImageRover is a content-based image browser for the world wide web. Technical challenges associated with this project are due in part to the staggering scale of the world wide web, and to the problem of developing fast and effective image indexing methods that support a wide variety of possible users. In this paper, we describe two key components of the ImageRover strategy: image digestion modules, and relevance feedback.

Image digestion modules extract information about the distribution of various image properties present in an image. These properties are then stored in vector form for

use in indexing and search. We have described the four image digestion modules which are fully-implemented in our system: a color module, and three texture modules computing the Wold features of periodicity, directionality, and randomness.

The system employs a novel relevance feedback algorithm that selects a weighted combination of L_m distance metrics appropriate for a particular query. The user can implicitly specify what image properties are appropriate by selecting example images. Thus, naive users are shielded from the need to understand the particular image measures and statistics employed in the index.

The resulting search tool provides a powerful method for *data exploration* or browsing of WWW images.

Acknowledgments

This work is sponsored in part by through grants from the National Science Foundation (Faculty Early Career Award #IRI-9624168, and CISE Research Infrastructure Awards #CDA-9623865 and #CDA-9529403). Marco La Cascia is sponsored in major part by a doctorate scholarship from the Italian Ministry for University and Scientific Research.

References

- [1] J. R. Smith and S.-F. Chang. Visualeek: a fully automated content-based image query system. In *Proc. ACM Multimedia '96*, November 1996.
- [2] C. Frankel, M. Swain, and V. Athitsos. Webseer: An image search engine for the world wide web. Technical Report 96-14, University of Chicago, August 1996.
- [3] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, D. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The qbic system. *IEEE Computer*, pages 23–30, September 1995.
- [4] S. Sclaroff, L. Taycher, and M. La Cascia. Imagerover: A content-based image browser for the world wide web. In *Proc. of IEEE Workshop on Content-based Access of Image and Video Libraries*, June 1997.
- [5] D. White and R. Jain. Algorithms and strategies for similarity retrieval. Technical Report VCL-96-101, University of California, San Diego, July 1996.
- [6] J. H. Friedman, J. H. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

- [7] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.
- [8] A. Gupta. Visual information retrieval technology: A virage perspective. Technical Report Revision 3a, Virage Inc., 177 Bovet Road, Suite 540, San Mateo, CA 94403, 1996.
- [9] A. Pentland, R. Picard, and S. Sclaroff. Photo-book: Tools for content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254, June 1996.
- [10] F. Liu and R. W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(7):722–733, 1996.
- [11] A. Pentland, B. Moghaddam, T. Starner, O. Oliyide, and M. Turk. View-based and modular eigenspaces for face recognition. In *Proc. CVPR*, pages 84–91, 1994.
- [12] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [13] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Low-rank orthogonal decompositions for information retrieval applications. *SIAM Review*, 37(4):573–595, 1995.
- [14] M. W. Berry and R. D. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numerical Linear Algebra with Applications*, 3(4):301–328, April 1996.
- [15] U. Gargi and R. Kasturi. An evaluation of color histogram based methods in video indexing. In *Proc. of International Workshop on Image Databases and Multi-media search*, August 1996.
- [16] J. Hafner, Harpreet Sawney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(7):729–736, 1995.
- [17] W. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.
- [18] M. Gorkani and R. Picard. Texture orientation for sorting photos at a glance. In *Proc. IEEE Conf. on Vision and Pattern Recognition*, 1994.
- [19] R. Picard and T. Minka. Vision texture for annotation. *Multimedia Systems*, 3(3):3–14, 1995.
- [20] J. Mao and A. K. Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern Recognition*, 25(2):173–188, 1992.
- [21] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1989.
- [22] R. O. Duda and P. E. Hart. *Pattern Recognition and Scene Analysis*. John Wiley, New York, 1973.