Progressive search and retrieval in large image archives

by V. Castelli
L. D. Bergman
I. Kontoyiannis
C.-S. Li
J. T. Robinson
J. J. Turek

In this paper, we describe the architecture and implementation of a framework to perform content-based search of an image database, where content is specified by the user at one or more of the following three abstraction levels: pixel, feature, and semantic. This framework incorporates a methodology that yields a computationally efficient implementation of image-processing algorithms, thus allowing the efficient extraction and manipulation of user-specified features and content during the execution of queries. The framework is well suited for searching scientific databases, such as satellite-image-, medical-, and seismic-data repositories, where the volume and diversity of the information do not allow the *a priori* generation of exhaustive indexes, but we have successfully demonstrated its usefulness on still-image archives.

1. Introduction

The last several years have seen the advent of numerous digital image and video libraries that, today, comprise tens of terabytes of on-line data. As a result of the continued proliferation of this kind of nontraditional data, these libraries will continue to grow significantly. For example, the instruments on the first two Earth Observing System platforms, to be launched in 1998 and 2000, will generate data at a rate of 281 gigabytes a day [1]. Other examples are in the seismic- and medical-imaging areas, in which terabytes of data are continuously acquired and stored. Consequently, new infrastructure that can support efficient storage, retrieval, and transmission of such data is needed. The search of image and video libraries, unlike the search of conventional text-based digital libraries and databases, cannot be realized simply through the search of text annotations. Because of the richness of detail in image and video data, it is difficult to provide automatic annotation of each image or video scene without human intervention. Therefore, we face the challenge of developing completely automatic mechanisms that extract meaning from this data and characterize the information contents in a compact and meaningful way—in other words, provide means for *content-based search*—and to ensure that these mechanisms scale well with the number of users, size of the library, and size of the objects stored within the library.

In general, there exist three different levels of abstraction at which image or video objects can be defined and searched—semantic, feature, and pixel. For instance, one can search for images containing houses (semantic level), regions with a specified texture (feature level), or similar, pixel-by-pixel, to a template (pixel level). Objects at each level can be either pre-extracted and indexed (an

0018-8646/98/$5.00 © 1998 IBM

index is a data structure designed to allow efficient retrieval) when new images are placed in the data repository, or evaluated at query time by analyzing pre-extracted information or even the images themselves. Current systems tend to pre-extract as much information as is practical, in order to allow efficient indexing and access to the desired information.

A *pixel-level object* is a connected subimage. Two subimages are similar at the pixel level if they have the same size and shape (e.g., they are both square and have the same number of pixels), and if pixels in the same position have similar values. Similarity matching thus relies on low-level operations, such as correlation (template matching). Template matching for satellite images can be used for coregistering images of the same region acquired by a satellite in different orbits. Since it is not practical to pre-extract all possible subimages, such operations must be performed at query time. Because this has been prohibitively expensive, most systems avoid the use of pixel-level techniques on large archives.

A *feature-level object* is a connected region of an image having uniform feature values: for instance, a region with homogeneous texture constitutes a texture object. The user specifies feature-level search parameters either by providing sample images, from which "feature vectors" for search are extracted, or by explicit specification of feature values or ranges. Our system allows the user to specify queries at the feature level with forms such as

Find all areas that have texture similar to this sample image.

and

Find all images that have color histograms similar to this set of color components: Red = X%, Purple = Y%, Green = Z%.

Feature-level objects are often pre-extracted by either blocking the image (i.e., by subsetting it into rectangular regions, possibly overlapping) or employing clustering and segmentation [2] techniques. Efficient indexing techniques, such as R-trees [3], can then be used to optimize the search for particular feature values.

A *semantic-level object* is a connected region of an image to which we can assign a unique semantic content. For instance, a photographic image can contain mountains, houses, flowers, people, animals, etc. Our system supports searches at the semantic level in the form:

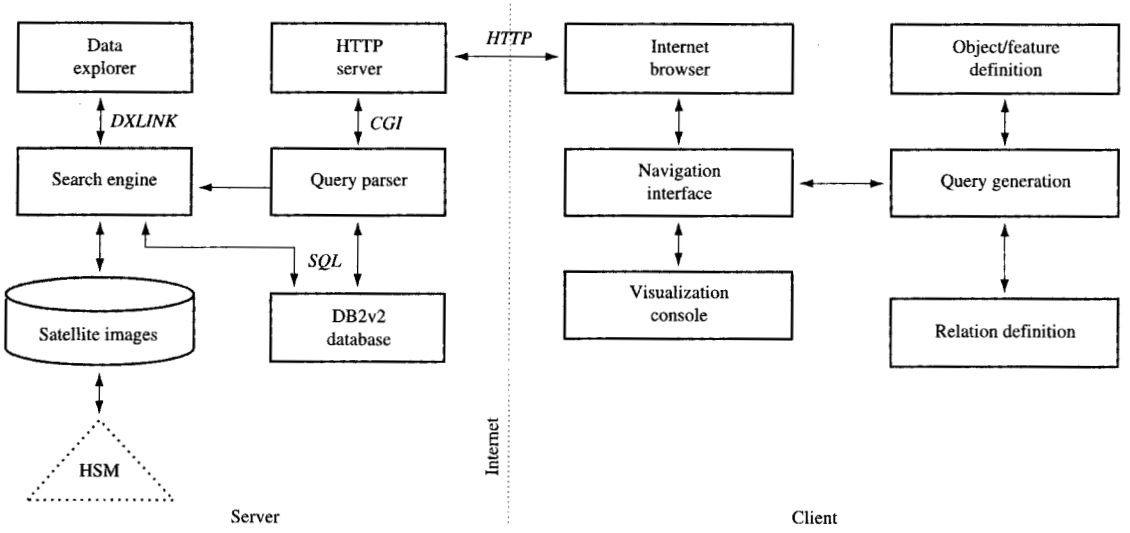
Find all bodies of water in the selected geographic region that are within 50 miles of any fire that is greater than 10 miles in diameter and has been burning for less than 3 days.

Semantic objects are typically predefined by applying classification algorithms to image features (such as texture or spectral histograms), or to the original data. Note that the primary distinction between semantic objects and feature objects is that semantic objects have a semantic label (such as "water" or "pine forest"), while feature objects are simply characterized by their feature values.

Extraction of features from pixels and of semantic objects from features are lossy operations. Although each higher level of abstraction improves search efficiency by reducing information volume, there are corresponding losses of accuracy. For this reason, it is necessary to provide mechanisms that support searches specified at all of these abstraction levels.

At the present time, there are many systems that perform indexing of image or video through the use of low-level image features such as shape, color histogram, and texture. Prominent examples for photographic images include IBM QBIC [4], the MIT PhotoBook [5], VisualSeek from Columbia University [6], and the Multimedia Datablade from Infomix/Virage [7]. These techniques have also been applied to specific application domains, such as medical imaging [8, 9], art [10], and video clips [11–15]. All of these examples rely on a preprocessing stage in which appropriate features are extracted and indexed. Although pre-extracted features and/or semantic objects permit efficient indexing schemes, they do not allow searching with all possible query semantics. To support a wide range of content-based queries, we must allow the user to form new semantic categories and/or new feature definitions.

We are currently studying these issues in a project under joint sponsorship by IBM and the NASA Goddard Space Flight Center. Our goal is to explore technologies that will facilitate the storage, query, and retrieval of images from large digital libraries by a diverse community of users, and to demonstrate the use of these technologies in a functional test bed. It is our thesis that though employing predefined schema, similar to those used in conventional databases, is useful and necessary, such schema will be insufficient to adequately support content-based search. It is necessary to provide users the capability to dynamically define new features and objects, and interactively specify and refine the target for content-based search of the image data. Thus, to extract the content dynamically defined by the user, we need the ability to analyze the raw images in the repository at query time. But the large size of the images stored in typical scientific repositories and the complexity of the operations involved make it impossible to use traditional image-processing operations in an interactive system. To overcome this difficulty, we have developed an extendible framework (called a progressive framework) in which the search targets are specified at one or more different abstraction levels. In this paper, we describe elements of our system that maximize retrieval efficiency, concentrating on the combination of image representation with image analysis at the heart of the progressive



Architecture of the content-based retrieval system.

framework. In particular, we show how to rely on properties of image-compression schemes that reorganize the information contained in the data, how to appropriately stage the analysis operators to produce successive approximations of their final results, by selectively operating on very small portions of the data, and how to identify and discard early in the computation those images that do not match the query.

To evaluate the effectiveness of the approach, we have developed a set of benchmark queries. We observed that conventional implementations of image-processing operators result in unacceptable execution times, while the progressive approach yields a significant increase in speed, which makes it appealing, even in an interactive system. The net effect is to make a wider range of image-processing and image-understanding operators available at run time, thereby extending the range of query operators that can be provided to the user.

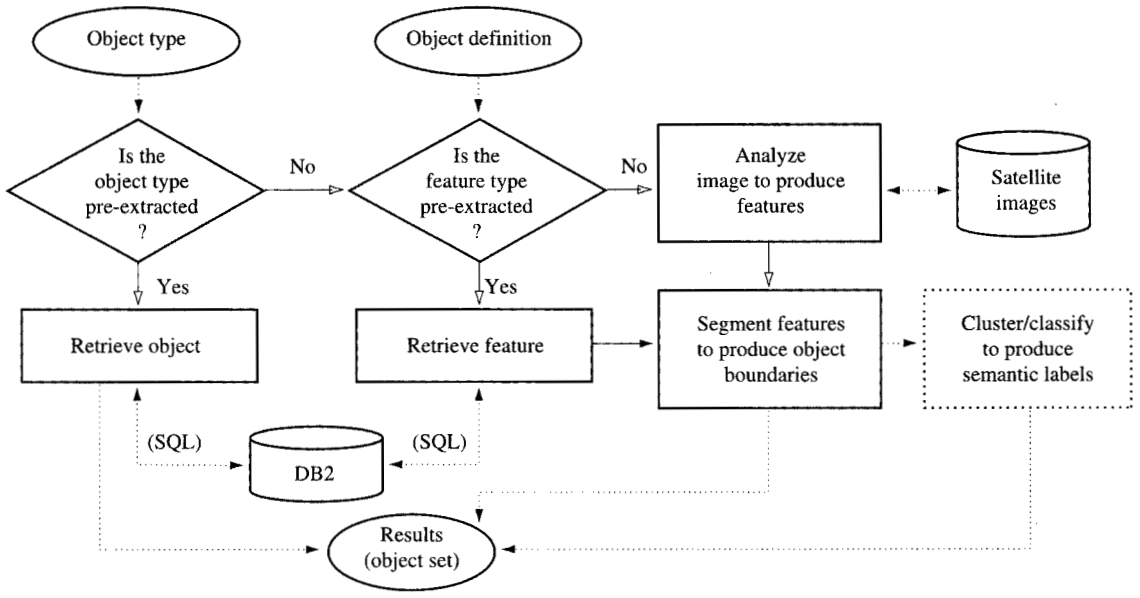
The remainder of this paper is organized as follows. The current system architecture is discussed in Section 2. Section 3 describes data representation and outlines the ways in which we use data representation to make the processing more efficient. Section 4 is devoted to the description of some fundamental content-search operators. In Section 5, we describe a set of queries that measures the benefits of our approach. Discussion and conclusions are in Section 6. Although the analysis in this paper is focused on satellite-image databases, the concepts presented herein are also applicable to other multimedia databases.

2. System architecture

In this section, we describe the architecture and the basic functionality of our content-based search system. As shown in **Figure 1**, our system is composed of a client and a server using the http protocol to communicate across the Internet.

The client is a modular Java** applet, which can be launched by any Java-enabled Internet browser. A navigation interface allows the user to visually navigate through the image database by graphically selecting geographical regions of interest, relevant time periods, and datasets (specified as instruments on satellite platforms, such as the LANDSAT MSS, or as derived product sets, such as the NALC triplicates). The navigation interface also allows the user to zoom in and out on selected portions of available images. The user can specify content-based queries in a quasi-natural language with the query-generation module, which is based on a drag-and-drop interface. Queries consist of relations between objects and constraints on the objects. Relations and objects are either predefined or specified by the user, through a relation-definition module and an object/feature-definition module, respectively. Finally, the user, by employing a visualization console, can specify the format for visualizing the results.

The following scenario provides an example of query specification. A fire manager from the forest service wants to assess the status of water sources near a forest fire. Helicopters, which can retrieve water (even from beaver ponds, if they are deep enough), are effective if the water is not too far from the fire, and they are much cheaper to



Object retrieval for a content-based query.

deploy than tanker planes. The fire manager thus restricts the region of interest to a large area around the fire, selects as time frame the last five months, and limits the datasets to those having at least 30 meters of resolution. Then the manager constructs the query: *Find all of the bodies of water at least 50 meters wide and 3 feet deep that are within 10 miles of the portion of the forest to which the fire might quickly extend*. The last part of the query is specified by providing an example of texture that corresponds to a "fuel type" that is particularly prone to burning quickly and by requiring the system to *find all examples of this texture that are near the geographical coordinates of the fire*. The depth of the lake cannot be directly derived from a satellite image, but it can be inferred, or guessed, from the reflectance in appropriate spectral bands. Finally, the fire manager requests that the results be plotted as bounding boxes on a map of the area. The query contains restrictions on the metadata, two types of objects (bodies of water and the region characterized in terms of texture), constraints on each of the objects (the size and depth of the lake and the location of the texture object), and a relation between the objects (they must lie within ten miles of each other). The client prepares the query in the form of a properly formatted message and sends it to the server. The server architecture is centered around a query parser, which communicates with an HTTP server via the CGI (Common Gateway Interface). The query parser receives the formatted message containing all of the query specifications from the client and translates it into a program [written in an internal representation language that combines Structured Query Language (SQL) statements with powerful search primitives and operators]. The query parser can communicate with a database, for example DB2* Version 2 (DB2v2 in the figure), using SQL. The program generated by the query parser is executed by the search engine. The search engine interfaces with the database and with a data repository, in our case containing satellite images, that, for scalability, can rely on a Hierarchical Storage Manager (HSM). Finally, to produce complex visualizations, the search engine can use a visualization engine. In particular, we currently use the IBM Data Explorer*, with the DXLINK application library as the interface.

During the execution of a query, the search engine first reduces the search space to the images that satisfy the metadata restrictions (region of interest, time frame, and data sets) by issuing an SQL query to the database and then executes the program only on the reduced search space. The latter step involves the retrieval of objects and features and the evaluation of constraints and relations.

The control flow of object retrieval is diagrammed in **Figure 2**. First, the search engine determines whether the type of the desired object has been pre-extracted and

stored in an appropriate file. In the example, the object type "body of water" would be pre-extracted, and all of the object instances of the type "body of water" that fall within the region of interest would be retrieved. Of course, not all possible types of objects are pre-extracted; rather, their definition in terms of elements at lower abstraction levels (such as pixels or features) can be stored or provided by the user. In the example, the user specifies a new object type in terms of a texture example—the fuel type. The search engine then determines whether texture data has been pre-extracted and stored in a file; if this is the case, it retrieves all of the feature elements for the region of interest, segments them into homogeneous regions (the "texture" objects), and retrieves those similar to the user-provided example. A further step involving automatic classification can be used to produce semantic objects. Finally, if the features are not pre-extracted, the search engine computes them by directly analyzing the stored images using image-processing operators.

Operators (such as retrieval from existing files, classification, feature extraction, and segmentation) produce candidate objects that are either "sharp" (e.g., a body of water) or "fuzzy" (e.g., a texture object that is similar to a user-provided sample image). Analogously, constraints on objects can be either sharp (in the example, the minimum width of the lake) or fuzzy (e.g., we could be looking for "deep" bodies of water). Finally, the relations between objects can themselves be sharp (as the "within 10 miles" relation in the example) or fuzzy (the texture objects must be "near" the fire). Fuzzy logic [16] is then used to compute a membership function (used as score) for each of the retrieved results, and the system returns them ranked in order of descending score.

The order of extraction of the objects and of the evaluation of constraints and relations is dynamically selected to maximize the pruning of the search space; each new step further reduces the amount of data and progressively refines the search (hence the name progressive framework).

3. Progressive image representation

Although the price of storage devices continues to drop at a dramatic rate, there is no doubt that the major cost of providing a digital library will continue to be in the storage devices. Thus, a reduction of even 30%, by the use of compression, in the storage required by the system would result in a significant reduction in overall cost. Although it would seem that processing images stored using compression techniques would result in reduced efficiency, this need not be the case; in fact, if we organize the data appropriately, the search process becomes more efficient and requires accessing only a small fraction of the data. One of the primary ideas of our project is the possibility of increasing the speed of searching through images stored in a digital library while simultaneously reducing the storage requirements. The remainder of this section explores this subject in more detail.

- *Compression and progression*

Image compression, or, more generally, source coding, can be categorized as lossless or lossy, depending on whether all of the original data can be recovered.

The first step of a large class of compression algorithms is a transform, such as the discrete cosine transform (DCT) used in both JPEG [17] and MPEG [18], or the discrete wavelet transform (DWT) [19] that improves the compression performance by concentrating most of the information (i.e., most of the energy) in a few transform coefficients. The transformed data is then "thresholded" (i.e., values less than a specified threshold are changed to zero) in order to eliminate the coefficients that are close to zero, quantized, and finally coded using lossless compression techniques. The thresholding and quantization steps in existing lossy compression standards such as JPEG are usually designed to minimize the perceptual difference between the original and the stored data. However, quantization schemes can be selected to best suit the requirements of the intended applications.

Applying query and retrieval operations directly on lossily compressed data generally leads to improved computational efficiencies along two fronts:

- The required I/O bandwidth between storage units and CPU is significantly reduced.
- The features and properties of the data can be emphasized by the transform step of the compression scheme.

In particular, query operations (e.g., analysis, retrieval, evaluation, transmission, and visualization) on image or video data can be staged progressively to minimize the total execution time as follows: Instead of operating on an entire image, the algorithm first analyzes small, selected portions of the transform and, on the basis of this information, decides whether the image can satisfy the query. Images that do not satisfy the query are quickly discarded; the others are further analyzed by accessing additional portions of the transformed data, and the process is repeated. At each step, the search space is progressively refined.

- *Wavelet-based compression*

Our system uses a transform-based image-coding scheme, which we call multiresolution compression for image analysis (MCIA), that simultaneously yields lossless and lossy compression. The structure of our algorithm is shown schematically in **Figure 3**. It consists of a lossy component and a lossless component; the lossy component uses DWT

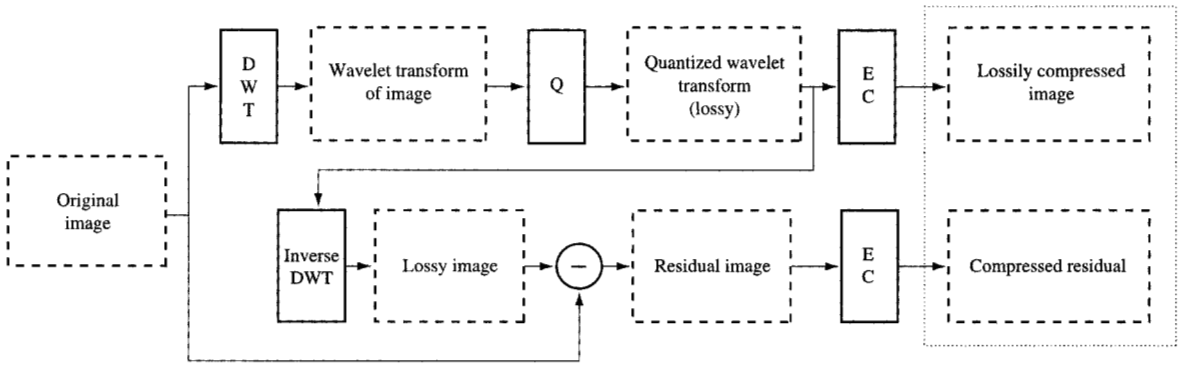


Figure 3

Overall structure of the compression algorithm. The image is transformed, quantized (Q), entropy-coded (EC), and stored. The difference between the original and the lossy data (residual) is entropy-coded and stored. The combination of compressed lossy data and residual yields lossless compression.

coding (for a simple introduction to wavelets and DWT, see [19] and the references therein) followed by quantization of the transform coefficients and lossless coding of the quantized values. To achieve lossless compression, we compute the difference between the lossy image and the original image, to produce a *residual* image, which is then losslessly coded ([20], Chapters 5 and 6).

We use a wavelet transform based on short orthogonal or biorthogonal filters, such as the Daubechies biorthogonal symmetric wavelets [21] of order 1 to 5 (although the system supports a large variety of wavelets). If the original data is stored in integer format, these filters allow perfect reconstruction; i.e., the transform step is perfectly invertible (of course, if the original data is stored as real numbers and the computation is performed with finite precision, rounding errors might prevent perfect reconstruction). The wavelet transform takes as input an image and produces a matrix of coefficients with the same number of rows and columns. This matrix is divided into portions called subbands, which, from the signal-processing viewpoint, are obtained by separating different spectral components using linear filtering operations, and sampling the results.

We quantize ([20], Chapter 13) the coefficient matrix using a uniform scalar quantization scheme. A different number of bits for each coefficient is allocated for each subband. Since quantization results in a loss of information, this step is the (only) lossy portion of our coding scheme. The quantized subbands are then losslessly coded independently by means of predictive coding (DPCM [17], Chapter 5.2.1) followed by a fixed-model two-pass arithmetic coding [17, 22]. These two steps are denoted by EC (entropy coding) in the figure. The resulting lossily compressed image requires ten to twenty times less space than the original, without displaying appreciable artifacts.

The residual is computed by inverting the DWT of the quantized wavelet coefficients and calculating the difference between that and the original image; this difference is then losslessly encoded, using DPCM followed by arithmetic coding. The residual contains information that is difficult to predict (hence to compress), such as the sensor noise in satellite images. Thus, the residuals account for most of the storage requirement.

Some of the results from extensive comparison of our MCIA compression algorithm with several existing lossless image coding schemes on heterogeneous satellite images and on classical test images are summarized in **Figure 4**. In general, we have found that the compression rate (ratio of volume of coded data to volume of input data) of MCIA is close to the compression rates of the best commonly available lossless algorithms.

Our scheme has several advantages. First, it has low computational complexity: The cost of reconstructing the original image is directly proportional to the number of pixels. Furthermore, while inverting the DWT, one can generate a multiresolution representation of the image—a sequence of increasingly larger and more detailed approximations, ending with the image itself. (For lossy compression, the final image is a lossy version of the original.) In particular, our scheme yields a dyadic multiresolution pyramid: The original image is at level 0 of the pyramid, and the approximation at each level has half the number of rows and columns of the approximation at the previous level. Reconstructing an approximation at a specified level of the pyramid requires

four times fewer operations than retrieving the one at the immediately lower resolution level. Thus, from the wavelet transform, we can readily generate thumbnails and low-resolution representations of the original image. By using short filters and storing the data on disk using a proprietary scheme, we can also quickly reconstruct arbitrary portions of the image at the desired level of detail. This provides a significant speedup during the query process, since in general we do not need to analyze the whole image for content-based-search purposes. Finally, the space-frequency representation produced by the wavelet transform simplifies the implementation of certain image-processing operations, such as certain types of texture matching [23]. Since the search algorithms can operate on the lossy and highly compressed version of the image, the scheme is amenable to implementations that use hierarchical storage systems, where the relatively high volume of residuals would be stored on tertiary (slow) media and accessed only on user request.

4. Progressive content-based search operation

The primary objective of our system is to provide a framework for automatic search of an image repository based on the image content.

When the user specifies content by means of objects or features that are not extracted *a priori* and stored in the database, the image search engine can extract the desired information from the stored data using a set of image operators, combined through a C-like interpreted programming language.

Combining compression and image-processing operators has proved very effective in reducing the high computational complexity of the feature-extraction task, and sometimes in increasing the accuracy of statistical operators, such as classification.

While the approach has been used in conjunction with a variety of elementary image-processing operators, we discuss in more detail how to apply the progressive framework to three complex operations: template matching, which retrieves content at the pixel level; texture extraction, which retrieves information at the feature level; and classification, which attaches semantics to portions of the images.

- *Template matching*

In a query-by-example framework [4], *template matching* ([2], Chapter 7.5) allows the retrieval of images that contain exactly (or almost exactly) the example (template) specified by the user. Indexing is impossible in this context, since the user is allowed to specify at query time any template of any size. Thus, template matching requires comparing, pixel by pixel, the example (say of size $\Delta x \times \Delta y$) with each $\Delta x \times \Delta y$ subimage at execution time. In practice, template matching is rarely exact, as a result of image noise, quantization effects, and differences in the images themselves. In the satellite-image domain, seasonal changes alone introduce effects that make the matching process difficult. Thus, the goal is to retrieve the subimages that most closely approximate the template. Our experiments have shown that the correlation coefficient, discussed below, is a particularly robust measure of the "distance" between the template and a subimage.

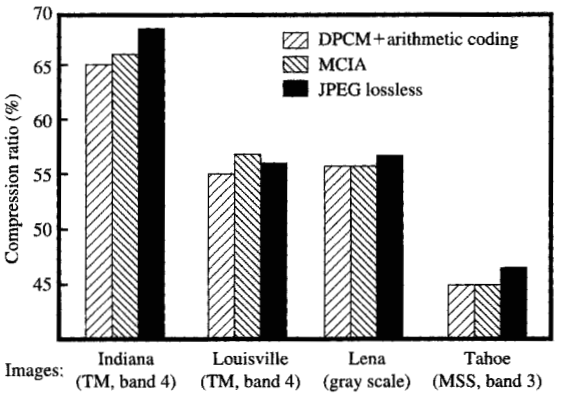


Figure 6

Comparison of MCIA compression ratio with the lossless mode of the JPEG standard and DPCM arithmetic coding. The Indiana and Louisville images were obtained with the LANDSAT Thematic Mapper; Lena is the most commonly used image in the image-compression literature, depicting the head of a woman wearing a feathered hat; the Tahoe image is a portion of a NALC dataset.

Let S denote the image and $S(i, j)$ denote the pixel at location i, j, where $i = 0, \cdots, N-1$ and $j = 0, \cdots, M-1$; let $S_{m,n}^{\Delta x,\Delta y}$ denote the subimage of size $\Delta x \times \Delta y$ starting at location m, n. The correlation coefficient $\rho(m, n)$ is defined in the usual way, as the empirical cross-correlation between properly normalized versions of the subimage $S_{m,n}^{\Delta x,\Delta y}$ and of the template T:

$$\rho(m,n) \triangleq \frac{\sum_{i=0}^{\Delta x-1}\sum_{j=0}^{\Delta y-1}[T(i,j)-\bar{T}][S_{m,n}^{\Delta x,\Delta y}(i,j)-\overline{S_{m,n}^{\Delta x,\Delta y}}]}{\sqrt{\left\{\sum_{i=0}^{\Delta x-1}\sum_{j=0}^{\Delta y-1}[T(i,j)-\bar{T}]^2\right\}\left\{\sum_{i=0}^{\Delta x-1}\sum_{j=0}^{\Delta y-1}[S_{m,n}^{\Delta x,\Delta y}(i,j)-\overline{S_{m,n}^{\Delta x,\Delta y}}]^2\right\}}},$$

where

$$\overline{S_{m,n}^{\Delta x,\Delta y}} \triangleq \sum_{i=0}^{\Delta x-1}\sum_{j=0}^{\Delta y-1}\frac{S_{m,n}^{\Delta x,\Delta y}(i,j)}{\Delta x \cdot \Delta y}, \qquad \bar{T} \triangleq \sum_{i=0}^{\Delta x-1}\sum_{j=0}^{\Delta y-1}\frac{T(i,j)}{\Delta x \cdot \Delta y}.$$

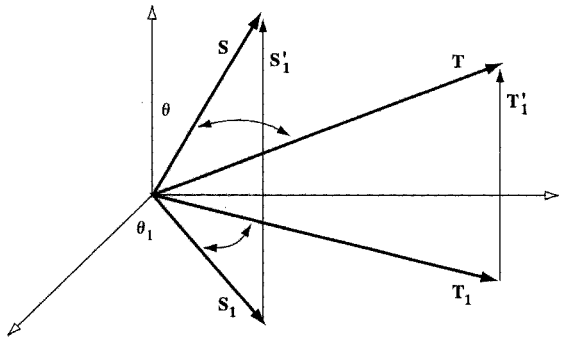


Figure 5

Geometrical interpretation of the progressive correlation coefficient algorithm.

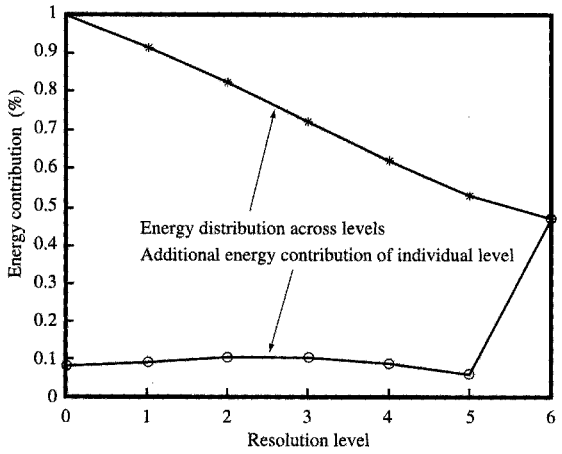


Figure 6

Energy distribution of the wavelet transform of a typical satellite image, as a function of the multiresolution level, and the additional contribution of each level to the energy of a six-level pyramid. Note that the lowest level contains about 50% of the total energy.

From the computational viewpoint, the set of quantities $\{\overline{S_{m,n}}\}$, $m = 0, \cdots, M - \Delta x - 1$, $n = 0, \cdots, N - \Delta y - 1$, can be computed in $O(M \cdot N) + O(\Delta x \cdot \Delta y)$ operations;[1] for very large images and moderate-sized templates, the latter term is negligible. Calculation of the quantities $\bar{T}$ and $\sum_{i=0}^{\Delta x-1} \sum_{j=0}^{\Delta y-1} [T(i, j) - \bar{T}]^2$ requires $O(\Delta x \cdot \Delta y)$ operations. Thus, the real computational cost is associated with the correlation in the numerator of $\rho(m, n)$. By invoking the convolution theorem for the Fourier transform [24], we can compute the correlation coefficient in $O(N \cdot M \cdot \log_2 M \cdot \log_2 N)$ operations, and better algorithms to compute the correlation coefficient exist only if $\Delta x \cdot \Delta y < \log_2 M \cdot \log_2 N$. Thus, the computational cost of the correlation coefficient is superlinear in the number of pixels in the image.

Progressive template matching, by first computing the correlation coefficient on a low-resolution version of the image and then progressively refining the results only around local maxima, allows one to approximate ρ without having to involve the entire image in the computation. More specifically, we start at a low resolution level, say ℓ_0, and compute the correlation coefficient between the level-ℓ_0 approximations of the image and of the template. At each level ℓ we identify regions where a match can occur, by first appropriately thresholding the correlation coefficient result and then finding local maxima. Areas, with size that depends on the level and on the size of the template, are then declared candidate regions. We then proceed to the immediately higher resolution, i.e., level $\ell - 1$, where we analyze with the same algorithm only the candidate regions, and discard the rest of the image. The procedure is recursively repeated until full resolution is reached.

The theoretical foundations of the approach rely on the results in [25] applied to models of real images. While the mathematical details are beyond the scope of the current paper, we give here a simple geometric justification. If we think of the normalized subimage and template as vectors in a $\Delta x \times \Delta y$-dimensional Euclidean space, W_0, the correlation coefficient can be interpreted as the cosine of the angle θ between these two vectors (for instance, the angle θ between **T** and **S** in **Figure 5**). Here, an exact match yields the maximal correlation coefficient, 1. The approximations to the template and the image at level 1 are the projections $\mathbf{T}_1$ and $\mathbf{S}_1$ of **T** and **S**, respectively, onto an appropriate subspace, W_1, defined by the wavelet transform. The correlation coefficient at level 1 is the cosine of the angle θ_1. If the difference vectors $\mathbf{T}'_1$ and $\mathbf{S}'_1$ ($\mathbf{T}'_1 \triangleq \mathbf{T} - \mathbf{T}_1$ and $\mathbf{S}'_1 \triangleq \mathbf{S} - \mathbf{S}_1$) are short, we can closely approximate θ with θ_1. On the other hand, if the difference vectors $\mathbf{T}'_1$ and $\mathbf{S}'_1$ are long, the approximation is poor. Similarly, the approximations at level ℓ can be represented by vectors $\mathbf{T}_\ell$ and $\mathbf{S}_\ell$ in the subspace W_ℓ, where $W_\ell \subset W_{\ell-1} \subset \cdots \subset W_0$, $\mathbf{T}'_j \triangleq \mathbf{T}_{j-1} - \mathbf{T}_j$, and $\mathbf{S}'_j \triangleq \mathbf{S}_{j-1} - \mathbf{S}_j$. Thus, $\mathbf{T} = \mathbf{T}_\ell + \sum_{j=1}^{\ell} \mathbf{T}'_j$; similarly for **S**. The energy of a typical satellite image is distributed across the different resolution levels, as shown in **Figure 6**. It is apparent that the level-3 approximation captures approximately 73% of the energy of the normalized template (the energy is equal to the length of the vector

[1] We use here the Bachmann–Landau O-notation: A function $f(x) = O[g(x)]$ as $x \to x_0$, if there exist finite constants K and $\delta x > 0$, such that for all x satisfying $|x - x_0| < \delta x$, the inequality $|f(x)| < K|g(x)|$ holds.

squared); that is, the length of the vector $\mathbf{S}'_3 + \mathbf{S}'_2 + \mathbf{S}'_1$ is about half the length of $\mathbf{S}_3$. We have developed techniques that allow us to quantify the effects of approximating the angle θ with θ_ℓ without having to convert back to the spatial domain [26]. By appropriately choosing the starting level ℓ_0, we can still guarantee that the results of our search are identical to those obtained when operating on the full-resolution image.

In practice, the larger the template, the better the speedup that can be achieved through this technique. For templates of size 64 pixels × 64 pixels, our experiments indicate an expected speedup ranging between 20 and 50. We have observed that the technique is very robust with respect to sensor noise, being essentially insensitive to the noise levels usually observed in satellite images. Also, for remotely sensed images, the very nature of the correlation coefficient (which does not depend on the length of the vectors) ensures that the effects of changes in sensor gain, sun angle, and seasonal variations in reflectance have little impact on the performance of the technique.

- *Texture analysis*

Texture is a perceptual concept that is rather difficult to formalize. It roughly captures regularity and organization, or lack thereof, of the luminance of neighboring pixels. As such, it is a local property of an image; i.e., different regions can have different texture. Many different approaches have been proposed to capture mathematically or numerically the concept of texture. Usually, the image is divided into square, rectangular, or irregular blocks, possibly overlapping, and certain numerical quantities, called *texture features*, are extracted from each block.

We have compared the effectiveness of different texture features in retrieving satellite images on the basis of similarity [27], and we have identified and included in our system a set of features that outperforms all of the other most commonly used ones. These texture features include fractal dimension [28], coarseness, entropy, spatial gray-level difference (SGLD) statistics [29, 30], and several circular Moran autocorrelation functions [30].

Texture extraction is a rather time-consuming task, since it involves the computation of complex quantities for each block of the image. For interactive situations in which the features cannot be precomputed, such as content-based search of real-time images and video, feature extraction can become the major bottleneck. In our system, we pre-extract texture vectors, store them in files, and generate indexes. Still, given the large amount of data associated with each image, similarity search is a daunting task.

To improve the speed of searching an image or video database for texture similarity, we use an approach called progressive texture matching [31]. The goal is to find a specified number of textures in the database that are most similar to an example that the user provides by specifying one or more blocks of existing images. We assume that a progressive representation of the images and the videos being searched already exists, and in the following we identify it with an L-level multiresolution pyramid created by subband coding or by the wavelet transform. For each resolution level of the pyramid, we can divide the image approximation into regular regions (for example, overlapping square blocks) or into irregular regions according to boundaries determined by edge-detection or image-segmentation techniques, and compute a texture-feature vector for each of the regions. In general, lower-resolution approximations will result in a substantially smaller number of texture vectors. The extracted feature vectors can then be stored separately, level by level, together with other information about the region from which they were extracted, and indexed.

The steps of the progressive texture-matching algorithm are the following:

1. Compute the approximations at level $0, \cdots, L-1$ of the example provided by the user, and calculate a texture feature vector for each level.
2. Retrieve from the level-$L-1$ textures of the images in the database being searched, the Q_L vectors most similar to the level-$L-1$ vector extracted from the example. Call the regions from which the retrieved feature vectors were extracted $R_1^L, \cdots, R_{Q_L}^L$.
3. At the next higher level, ℓ, consider only the level-ℓ texture vectors that were extracted from regions that overlap the regions obtained in the previous step $(R_1^{\ell-1}, \cdots, R_{Q_{\ell+1}}^{\ell+1})$, and from them retrieve the Q_ℓ vectors most similar to the level-ℓ vector extracted from the example, together with the corresponding regions $R_1^\ell, \cdots, R_{Q_\ell}^\ell$.
4. Repeat the previous step until level 0 is reached.

The procedure progressively refines the search space by further pruning it at each level. The method is clearly nonexact, since it relies on the approximations. The choice of the $\{Q_\ell\}$ influences the result and the efficiency of the method. Since detailed description of the selection algorithm is beyond the scope of the present paper, we refer the interested reader to [31].

- *Classification*

Classification of a satellite image is the process of assigning semantic labels to the individual pixels or to distinct regions ([32], Chapter 8). In the remote-sensing community, images are often classified by analyzing each pixel location independently, and using as input to the classifier the spectral bands acquired by the instrument (typically four to two hundred). Different land-cover classes have different "spectral signatures"; for instance,

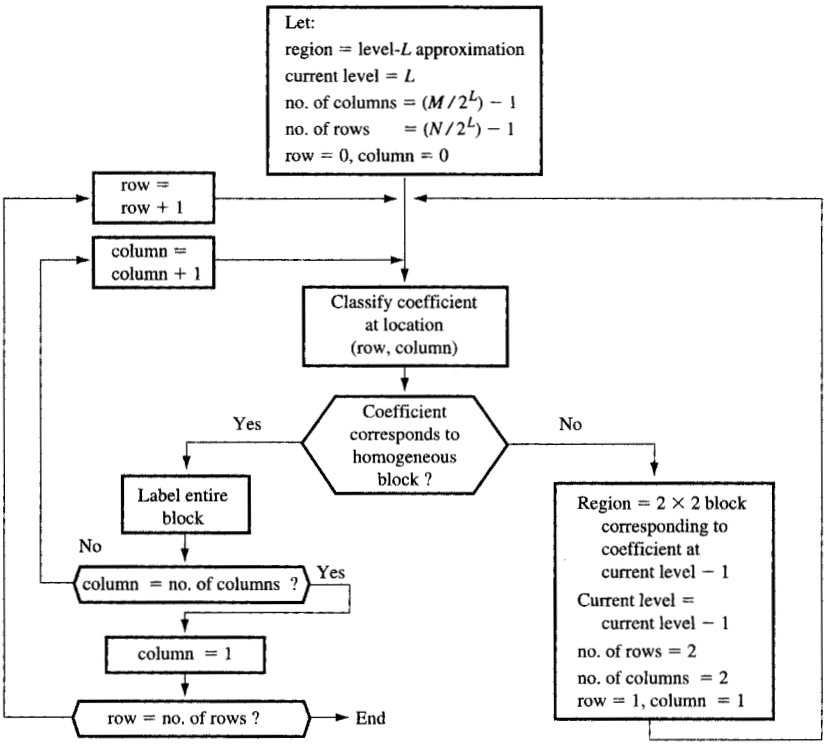


Figure 7

Schematic algorithm of progressive classification.

water has low reflectance across the visible and infrared portion of the spectrum, vegetation has low reflectance in the red and high reflectance in the green and near-infrared bands, and barren terrain has similar, medium-high reflectance in all bands.

Classification is an expensive operation—especially when complex classifiers, such as neural networks [33], are used. The typical size of a satellite image ranges from a million to one hundred million pixels; thus, classical techniques cannot be used as the basis for search operators. Still, classification is one of the essential image-analysis operations in the remote-sensing community, for which the ability to perform searches at the semantic level is essential. Different user groups use different classification taxonomies that cannot be easily reconciled. For instance, the "fuel type" classifications used by the U.S. Forest Service and the American Forest Service are rather different and cannot be reduced to a common denominator. This implies that we can only pre-extract and index semantic content corresponding to very simple taxonomies, and that searching for more complex semantic content must be done at query time.

Our approach, called progressive classification, relies on properties of the multiresolution pyramid to alleviate the computational cost. It is not a new classifier; rather, it is a framework that can be used with essentially all of the existing classifiers.

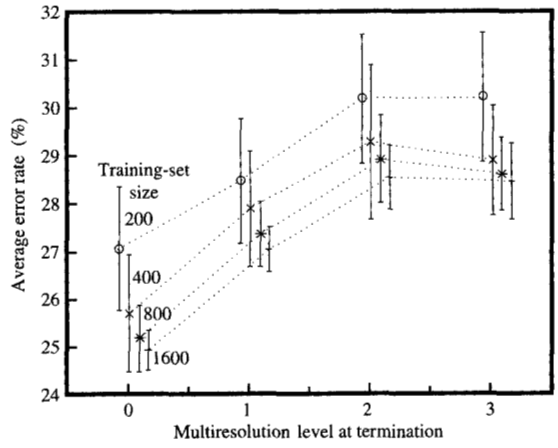
In a multiresolution framework, we can consider a wavelet coefficient at level ℓ "essentially" as the low-resolution representation of a block of 2^ℓ pixels $\times$ 2^ℓ pixels at level 0. A progressive classifier operates as depicted in **Figure 7**: First, the approximation of the image at a predefined resolution level L is labeled by an appropriately trained classifier, which decides whether each coefficient corresponds to a homogeneous or to a heterogeneous $2^L \times 2^L$ block of pixels in the untransformed image. In the former case, the block is assigned a label, while in the latter, the transform is inverted within a small region surrounding the pixel, to yield four coefficients at the immediately higher resolution level, and the classifier further analyzes them independently. The recursive process terminates when either a homogeneous block is detected or full resolution (level 0) is reached.

As proved in [34], this scheme has two advantages. First, it is faster than the full-resolution, pixel-by-pixel approach, since in images from nature there is significant dependence between the labels of adjacent pixels, which allows large portions of the image to be classified at low resolution. Second, somewhat surprisingly, it is more accurate than the pixel-by-pixel approach.

In practical experiments, we have used progressive classification in conjunction with different parametric and nonparametric methods, such as discriminant analysis (based on the assumption of Gaussian behavior, and also known, regrettably, as the maximum-likelihood classifier), k-nearest neighbor, learning vector quantization, clustering-based schemes [2], and CART [35]. Our experiments have shown that the progressive approach is substantially faster than the straightforward classification of the original image. For instance, when the process was started at level $L = 2$, the classification time was consistently reduced by a factor of 5. More detailed descriptions of the construction of a progressive classifier and of experimental results can be found in [36].

Further speedups can be achieved by ending the process at a higher level than full resolution. This results in a higher number of classification errors, but experiments summarized in **Figure 8** show that the increase in error rate from one resolution level to the next is in most cases comparatively small, making this a viable option when the user is willing to trade accuracy for response time. The different "curves" in the figure correspond to different sizes of the training sets used to construct the classifier. Our system has a facility that allows the user to construct a classifier "on-the-fly" by selecting regions of images and assigning labels to them. This process, which is also supported by many image-processing packages commonly used in the remote sensing community, is rather tedious, and constructing large, reliable training sets is a burdensome task. Thus, we decided to capture in the figure the penalty that one incurs by using a small training set.

To generate the figure, we used a satellite image of the Black Hills, acquired by the Landsat multispectral scanner in 1972, for which the "ground truth" is known. For each final resolution and each of the training-set sizes, we randomly selected 20 training sets with which we trained a progressive classifier. With each trained classifier, we labeled the rest of the image (i.e., the part not used for training) and compared the results against the ground truth, thus producing an estimate of the accuracy. For each pair (resolution and training-set size), we have summarized the 20 experiments by plotting the mean and the standard deviation of the probability of error. The *increase* in accuracy that occurs in the change from level 2 to level 3 is partially due to statistical variations, partly because about 75% of the pixels in the image belong to few dominant classes, and the classifier that terminated the execution at level 3 essentially ignored all of the nondominant classes.



Accuracy of progressive classification as a function of the resolution level at which the process is terminated and the training-set size. The data points give the mean error rate, and the vertical lines give the estimated standard deviation for 20 experiments with different training sets and test images.

5. Evaluating the approach

To the best of our knowledge, there are no standardized benchmarks for content-based search that would appropriately capture the benefits of our approach—in particular, the effectiveness of the progressive image-processing operators. Consequently, we have defined, in collaboration with NASA, the following set of benchmark queries. For each query, a region of interest (e.g., Montana), a time frame (e.g., 1970–1990), and a list of satellite instruments (e.g., Landsat TM and Landsat MSS) are specified, and the search is run against all of the images in the data repository that satisfy the constraints.

1. Given a template T, find the best matches and return their locations. (The desired number of matches is specified by the user.)
2. Given a texture pattern T_x, find the two images containing textures that best match T_x. The texture-similarity criterion is defined *a priori*.
3. "Mosaick" a subimage (create a composite) of size $(dx \times dy)$ with the upper left corner at location (x, y) from image A with another image of size $(du \times dv)$ with the upper left corner at location (u, v) from image B, and display the result. (This query is an example of

image-processing operations that do not benefit significantly from the progressive framework.)

4. Classify image S, using the USGS (United States Geological Survey) nine-class taxonomy, identify the centroid of the largest body of water, and return its coordinates.
5. Find all images with less than $p\%$ cloud cover.
6. Find all locations of urban areas adjacent to forests.
7. Find all images covering the region of interest acquired between 1970 and 1990. For each location, consider the oldest and the most recent image. Classify them according to the USGS nine-class taxonomy, identify areas where the land cover has changed, and return the rate of change of the classes. (This query shows the usefulness of our system in global change and other environmental studies.)

To measure the performance, we executed each query both using the progressive framework from within our prototype system and on the original images, as a stand-alone process ("baseline implementation"). The original images are stored in uncompressed format; for multispectral data, each spectral band is stored separately in a flat file. No precomputed metadata or pre-extracted features are used during the execution of the queries; their main role within our system is to prune the search space by limiting the number of images to be analyzed. Thus, for benchmarking purposes, metadata and pre-extracted features are replaced by the specification of the input dataset.

The result of a progressive search operation is usually different from the result of the corresponding operation on the original images. This is partly because of the loss of information due to the lossy compression scheme and partly because of inherent differences between the algorithms. In order to compare the two approaches, we impose the following requirements on the query results: If a query returns a list (possibly ranked), both baseline and progressive implementations must return the same results; if the query returns a statistical estimate, the confidence intervals in the compressed domain must contain the corresponding confidence intervals produced by the baseline implementation.

For each query, we measured the user execution time and the real execution time, for both the baseline and the progressive implementation. Real execution time was measured by means of the UNIX** time command. The user time is a measure of the CPU time spent in executing the user code. Since the computers used for our experiments were devoted exclusively to the benchmarking tasks, the real time comprises user, system, input/output, wait, and application-loading times. On a computer with more than one user, the real time is a less reliable measure. Real-time and user-time speedups were computed as the ratios t_u^b/t_u^c and t_r^b/t_r^c, where t_u^b is the user execution time for the baseline implementation, t_u^c is the user execution time for the compressed-domain implementation, and t_r^b and t_r^c are the corresponding real execution times.

The experiments for query 7b (query 7 on a large image) were run on an IBM RS/6000* Model 590 workstation with 256 MB of RAM and the remaining ones on an IBM RS/6000 43P Model 132 workstation with 160 MB of RAM. The classification schemes used in the baseline and progressive implementations were essentially identical.

A potential problem with the baseline implementation is that with very large images, the memory requirement can exceed the available physical memory, and the resulting page faults can significantly impair the execution (the size of currently available satellite images is such that a similar phenomenon does not occur when progressive searches are executed on the computers used in the experiments). Under such conditions, speedups of over 500 were observed; an example is the real-time speedup for query 7b in **Table 1**. For all of the other queries, we have decided not to modify the baseline implementations to deal with very large images, but to choose the image sizes in such a way that the application resides entirely in physical memory. Had we decided to modify the baseline implementations, thus increasing their complexity, we would have observed additional speedup. The experiments, therefore, depict the worst case for the progressive implementation.

The application of the progressive framework to complex operations such as template matching and classification results in tangible benefits: The real execution times on the mentioned computers are reduced to a few seconds or a fraction of a second per (large) image. In some instances, we have observed smaller improvements—most noticeably in queries 2, 3, and 5. Query 2 was run on sea-ice images, where different types of ice result in subtle differences in the texture vectors. As a consequence, the progressive algorithm was started at level 2 of the pyramid, and lower levels were analyzed frequently. Query 3 demonstrates visualization operations; thus, the result of both progressive and baseline implementations is an image at full resolution. The image-processing operations involved in the query are very simple. The overheads of the progressive implementation (database access, entropy decoding, and load time of the system) then become comparable to, or even bigger than, the operation itself, thus accounting for a large portion of the execution time. Query 5 estimates the cloud-cover percentage and returns only the images with less than $p\%$ clouds. The reported execution times are the averages over four different values of p, namely 10, 20, 30, and 40. The algorithm estimates p and produces a 95% confidence

Table 1 Speedups and measured times for queries involving classification and template matching. The images for queries 1, 4, and 6 have seven spectral bands, while the images used in query 7 have four spectral bands. The real execution time for the progressive version of query 7a (smaller image) is dominated by the application loading time, while the real execution time for the baseline implementation of query 7b (large image) is dominated by page faults, even on a machine with 256 MB of RAM.

Query number	*Image size*	*Baseline*		*Progressive search*		*User time speedup*	*Real time speedup*
		User time (s)	*Real time* (s)	*User time* (s)	*Real time* (s)		
1	2815 × 3082	93.51	277.03	8.2	18.11	11.40	15.29
2*	2815 × 3082	—	—	—	—	4.4	3.7
3*	2815 × 3082	—	—	—	—	2.3	6.8
4	2815 × 3082	208.96	315.15	4.94	10.18	42.30	30.96
5†	1400 × 1470	111.9	242.15	11.68	26.19	9.58	9.25
6	2815 × 3082	145.06	242.15	4.76	8.64	30.47	28.03
7a§	1312 × 1312	26.69	29.4	0.79	2.85	33.78	10.32
7b§	4179 × 3988	297.42	7807.04	10.5	15.27	28.33	511.27

*The actual times for queries 2 and 3 were not computed, since the test program produced speedup figures directly.
†The reported times are for the analysis of 17 images.
§Query 7 involves the analysis of two images.

interval. In the progressive implementation, the images are analyzed at the immediately higher resolution level whenever the confidence interval of the estimate contains the threshold p. In the experiment, we used low-resolution data (2 km per pixel), most of which had cloud cover between 30% and 50%. Then, for $p = 30$ and 40, the progressive implementation analyzes most images at full resolution, thus accounting for the small observed speedup.

6. Discussion and conclusions

There are many important issues associated with providing an infrastructure for integrated multimedia libraries. Perhaps the single most important issue is the ability to perform content-based search. The different groups of users of satellite imagery with which we have interacted have presented us with a large number of different types of content. Thus, pre-extracting information and generating indexes becomes too time-consuming and impractical. We are exploring an approach that allows the user to define new features at query-construction time and to use such features to specify new, arbitrarily complex searches. Within our project, we are developing and demonstrating technologies that further our ability to support this capability.

Clearly, to avoid inefficiencies and reduce the computational cost and associated delays, the design of efficient indexing schemes to manage descriptors of the information in the library is highly desirable. Currently, in our system, we use indexes for the metadata and for precomputed features and objects. Since we have been unable to achieve significant speedups in computing texture features, which are extracted by complex and computationally intensive algorithms, we have identified, extracted, and indexed a set of texture features for each image in the database. The metadata and pre-extracted features are used to prune the search space, usually allowing the search engine to restrict its attention to a limited set of images.

The main contribution of our system is the capability of further narrowing the search results by extracting and manipulating user-defined features at query-execution time from this candidate set. Previously, this process has been regarded as impractical, especially in an interactive system, because of the high computational cost of the image-processing operations involved.

To overcome this difficulty, we propose a progressive framework that combines image representation (in particular, image compression) with image processing. Progressive implementations of image-processing operators rely on the properties of the compression scheme to reduce significantly the amount of data to analyze during the feature-extraction and -manipulation phases.

To measure the benefits of our approach, we have defined a set of benchmark queries and compared the results obtained from the baseline and progressive implementations of the algorithms. The observed speedups are significant: The computational cost has been reduced, on average, by a factor of almost 20, while the response time has been reduced by a factor of 16. We attribute the smaller speedup in response time to the testing environment we have agreed upon with NASA, in which the progressive versions of the queries are executed within our system. Thus, we incur the costs of loading the application and establishing connections to the database. While such overhead is minimal when the response time is of the order of minutes, it becomes substantial when the response time is less than a second, as in query 7.

The progressive operators are capable of extracting user-specified features at query time, and we can use these features to search for new, nonpredefined content, thus adding a new dimension to the flexibility of our query engine. We have shown the ability to analyze a large image (of 3000 × 3000 pixels and seven spectral bands, or 4000 × 4000 pixels and four spectral bands) in less than ten seconds of CPU time and less than 20 seconds of total response time, while the analysis of smaller images (1300 × 1300 pixels and four spectral bands or 1400 × 1400 pixels and two spectral bands) requires less than one second of CPU time.

We feel that the results of our experiments are encouraging, as they show that the significant increase in expressive power resulting from the extraction of new features at execution time can be achieved with reasonable computational cost.

Acknowledgments

The authors wish to acknowledge the contributions of Loey Knapp, Patricia Andrews, Larry Bradshaw, Bob Burgan, Steven Carty, Carolyn Chase, Satheesh Gannamraju, John Gerth, Sharmila Hutchins, Cam Johnston, Bryan Lee, Richard Ryniker, Bernice Rogowitz, Jerald Schoudt, Barbara Skelly, Alex Thomasian, Lloyd Treinish, and Joel Wolf, during the development of our prototype system. This work was funded in part by NASA/CAN Grant No. NCC5-101.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc., Moving Picture Expert Group, or The Open Group.

References

1. J. C. Tilton and M. Manohar, "Earth Science Data Compression Issues and Activities," *Remote Sensing Rev.* **9,** 271–298 (1994).
2. Richard O. Duda and Peter E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc., New York, 1973.
3. A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *SIGMOD Record* **14,** No. 2, 47–57 (1984).
4. W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape," *Storage Retrieval for Image and Video Databases, Proc. SPIE* **1908,** 173–187 (1993).
5. A. Pentland, R. W. Picard, and S. S. Sclaroff, "PhotoBook: Tools for Content-Based Manipulation of Image Databases," *Storage and Retrieval for Image and Video Databases, Proc. SPIE* **2185,** 34–47 (1994).
6. J. R. Smith and S.-F. Chang, "VisualSeek: A Fully Automated Content-Based Image Query System," *Proceedings of ACM Multimedia '96*, Boston, November 1996, pp. 87–98.
7. J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, and R. Jain, "The Virage Image Search Engine: An Open Framework for Image Management," *Storage and Retrieval for Still Image and Video Databases, Proc. SPIE* **2670,** 76–87 (1996).
8. T. Y. Hou, P. Liu, A. Hsu, and M. Y. Chiu, "Medical Image Retrieval by Spatial Feature," *Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, 1992, pp. 1364–1369.
9. T. Y. Hou, A. Hsu, P. Liu, and M. Y. Chiu, "A Content Based Indexing Technique Using Relative Geometry Features," *Image Storage and Retrieval Systems, Proc. SPIE* **1662,** 29–68 (1992).
10. B. Holt and L. Hartwick, "Visual Image Retrieval for Applications in Art and Art History," *Storage and Retrieval for Image and Video Databases, Proc. SPIE* **2185,** 70–81 (1994).
11. E. Deardorff, T. D. C. Little, J. D. Marshall, and D. Venkatesh, "Video Scene Decomposition with the Motion Picture Parser," *Digital Video Compression on Personal Computers: Algorithms and Technologies, Proc. SPIE* **2187,** 44–55 (1994).
12. H. Zhang and S. W. Smoliar, "Developing Power Tools for Video Indexing and Retrieval," *Storage and Retrieval for Image and Video Databases, Proc. SPIE* **2185,** 140–149 (1994).
13. H. Zhang and S. W. Smoliar, "Content Based Video Indexing and Retrieval," *IEEE Multimedia* **1,** No. 2, 62–72 (1994).
14. F. Arman, A. Hsu, and M. Y. Chiu, "Image Processing on Compressed Data for Large Video Database," *Proceedings of ACM Multimedia '93*, 1993, pp. 267–272.
15. F. Arman, A. Hsu, and M. Y. Chiu, "Feature Management for Large Video Database," *Proc. SPIE* **1908,** 2–12 (1993).
16. G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic, Theory and Applications*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1995.
17. William B. Pennebaker and Joan L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993.
18. Joan L. Mitchell, D. Le Gall, and C. Fogg, *MPEG Video Compression*, Chapman & Hill, New York, 1996.
19. Oliver Rioul and Martin Vetterli, "Wavelets and Signal Processing," *IEEE Signal Process Magazine* **8,** No. 4, 14–38 (1991).
20. Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory* (Wiley Series in Telecommunications), John Wiley & Sons, Inc., New York, 1991.
21. Ingrid Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.
22. Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic Coding for Data Compression," *Commun. ACM* **30,** No. 6, 520–540 (1987).
23. J. R. Smith and S.-F. Chang, "Quad-Tree Segmentation for Texture-Based Image Query," *Proceedings of ACM Multimedia '94*, October 1994, pp. 278–286.
24. Ronald N. Bracewell, *The Fourier Transform and Its Applications* (McGraw-Hill Electrical and Electronic Engineering Series), McGraw-Hill Book Co., Inc., New York, 1978.
25. P. P. Vaidyanathan, "Orthonormal and Biorthonormal Filter Banks as Convolvers, and Convolutional Coding Gain," *IEEE Trans. Signal Processing* **41,** No. 6, 2110–2130 (1993).
26. C.-S. Li, J. J. Turek, and E. Feig, "Progressive Template Matching for Content-Based Retrieval in Earth Observing Satellite Image Databases," *Proceedings of SPIE Photonics East, Proc. SPIE* **2606,** 134–144 (1995).
27. Chung-Sheng Li and Vittorio Castelli, "Deriving Texture Feature Set for Content-Based Retrieval of Satellite

Image Database," *Proceedings of the 1997 IEEE International Conference on Image Processing*, November 1997, pp. 567–579.
28. A. R. Rao, *A Taxonomy for Texture Description and Identification*, Springer-Verlag, New York, 1990.
29. J. Weszka, C. Dyer, and A. Rosenfeld, "A Comparative Study of Texture Measures for Terrain Classifications," *IEEE Trans. Systems, Man, Cybernet.* **SMC-6,** No. 4, 269–285 (1976).
30. Z. Q. Gu, C. N. Duncan, E. Renshaw, C. F. N. Mugglestone, M. A. Cowan, and P. M. Grant, "Comparison of Techniques for Measuring Cloud Texture in Remotely Sensed Satellite Meteorological Image Data," *IEE Proc.* **136,** No. 5, 236–248 (1989).
31. C.-S. Li and M.-S. Chen, "Progressive Texture Matching for Earth Observing Satellite Image Databases," *Proceedings of SPIE Photonics East, Proc. SPIE* **2916,** 150–161 (1996).
32. John A. Richards, *Remote Sensing Digital Image Analysis*, Springer-Verlag, New York, 1993.
33. Samir R. Chettri, Robert F. Cromp, and Mark Birmingham, "Design of Neural Networks for Classification of Remotely Sensed Imagery," *Telematics & Informatics* **9,** No. 3/4, 145–156 (1992).
34. Vittorio Castelli, Ioannis Kontoyiannis, Chung-Sheng Li, and John J. Turek, "Progressive Classification: A Multiresolution Approach," *Research Report RC-20475*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1996.
35. Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks/Cole, Pacific Grove, CA, 1984.
36. Vittorio Castelli, Chung-Sheng Li, John J. Turek, and Ioannis Kontoyiannis, "Progressive Classification in the Compressed Domain for Large EOS Satellite Databases," *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 1996, pp. 2201–2204.

Received November 13, 1996; accepted for publication March 28, 1997

Vittorio Castelli *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (vittorio@watson.ibm.com).* Dr. Castelli received a "Laurea" degree in electrical engineering from the Politécnico di Milano in 1988, with a thesis in bioengineering. He received an M.S. in electrical engineering in 1990, an M.S. in statistics in 1994, and a Ph.D. in electrical engineering in 1995 from Stanford University, with a dissertation in information theory and statistical classification. From 1995 to 1996, he was a Postdoctoral Fellow at the IBM Thomas J. Watson Research Center, where he is currently a Research Staff Member. His main research interests include image processing and image compression, statistical data analysis, statistical classification, and database mining. Dr. Castelli is a member of the IEEE Information Theory Society, the American Statistical Association, and Sigma Xi.

Lawrence D. Bergman *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (bergman@watson.ibm.com).* Dr. Bergman is a Research Staff Member at the IBM Thomas J. Watson Research Center, where his work focuses on content-based query and retrieval from image archives. He received his Ph.D. in 1993 from the University of North Carolina at Chapel Hill in computer science, with a focus on computer graphics and visualization. His research interests include visualization, user interfaces, and graphical languages.

Ioannis Kontoyiannis *Information Systems Laboratory, Stanford University, Durand 141A, Stanford, California 94305 (yiannis@isl.stanford.edu).* Mr. Kontoyiannis received an M.S. in pure mathematics from Cambridge University in 1993, graduating with a distinction in Part III of the Cambridge Mathematics Tripos. Since then he has been a Research Assistant to Professor Tomas M. Cover at Stanford University, where he received an M.S. degree in statistics in 1997, and expects to receive his Ph.D. in electrical engineering in June 1998. From June to December 1995 he worked as a co-op at IBM Research in Hawthorne, New York, where he took part in the image-processing project reported on in this work. His academic work has been in the areas of universal data compression, entropy theory of stationary processes and random fields, wavelet theory and signal processing applications, nonparametric statistics, and applied probability.

Chung-Sheng Li *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (csli@watson.ibm.com).* Dr. Li received his B.S.E.E. degree from the National Taiwan University, Taiwan, R.O.C., in 1984, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley in 1989 and 1991, respectively. He has worked in the Computer Science Department of the IBM Thomas J. Watson Research Center as a Research Staff Member since September 1991. His research interests include digital library, optical chip interconnects, optoelectronics, all-optical networks, broadband switching architectures, and high-speed analog/digital VLSI circuit design. He has co-initiated several research activities in IBM on fast tunable receivers for all-optical networks and content-based retrieval in the compressed domain for large image/video databases. He is currently the co-investigator of a satellite-image database project funded by NASA. Dr. Li received a Research Division Award from IBM in 1995 for his major contribution to the

tunable receiver design for WDMA, and numerous invention and patent-application awards. He is serving as the technical editor and feature editor for the *IEEE Communication* magazine. He has authored or coauthored more than 80 journal and conference papers and received one of the best-paper awards from the IEEE International Conference on Computer Design in 1992. He is a senior member of the IEEE Laser Electro-Optic Society, the Communication Society, the Computer Society, and the Circuit and System Society.

John T. Robinson *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (robnson@watson.ibm.com, http://www.research.ibm.com/people/r/robnson/).* Dr. Robinson received the B.S. degree in mathematics from Stanford University in 1974, and the Ph.D. degree in computer science from Carnegie-Mellon University in 1982. Since 1981, he has been with the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. His research interests include database systems, operating systems, parallel and distributed processing, and design and analysis of algorithms. He is a member of the ACM and the IEEE Computer Society.

John J. Turek *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (jjt@watson.ibm.com).*