# Progressive Search and Retrieval in Large Image Archives [1]


V. Castelli*
vittorio@watson.ibm.co
(914) 784-7946

L. Bergman
bergman@watson.ibm.com
(914) 784-7946

I. Kontoyiannis
yiannis@isl.stanford.edu
(415) 723-4544

C.-S. Li
csli@watson.ibm.com
(914) 784-6661

J. Robinson
robnson@watson.ibm.com
(914) 784-7127

J. Turek
jjt@watson.ibm.com
(914) 784-7591



[1] **Work funded in part by NASA/CAN grant no. NCC5-101**



* Vittorio Castelli is the **primary contact.**

## Abstract

Technological advances over the last several years have resulted in digital image and video libraries that, today, comprise tens of terabytes of on-line data. Due to the continued proliferation of image data, these libraries will grow significantly larger over the next few years. Consequently, the need for databases that can effectively support storage, search, retrieval and transmission of this kind of nontraditional data will grow significantly. This paper describes a project currently in progress under joint sponsorship by NASA, GSFC, that explores some of these challenges. 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 three abstraction levels: pixel, feature and semantic. This framework incorporates a progressive methodology that allows a computationally efficient implementation of image processing algorithms, yielding the efficient extraction and manipulation of user-specified featured and content during the query execution.

# 1 Introduction

The last several years have seen the advence 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 *non-traditional* data, these libraries will grow significantly larger over the next few years. As an example, the instruments on the first two Earth Observing System (EOS) platforms, to be launched in 1998 and 2000, will generate data at a rate of 281 GB/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. Unlike conventional text-based digital libraries and databases, search of image and video libraries cannot be realized simply through the search of text annotations. Due to 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, the challenge is to develop mechanisms that extract meaning from this data and characterize the information contents in a compact and meaningful way – in other words, provide support for *content-based search* – and to ensure that these mechanisms scale well with the number of users, size of the library, and the 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 - pixel, feature, or semantic. Object at each level can be either pre-extracted or evaluated at query time. Current systems tend to pre-extract as much information as practical to allow efficient indexing and access to the desired information.

A pixel-level object consists of a subimage which is to be matched using low-level operations such as correlation (template matching). Due to the fact that it is not practical to pre-extract all possible subimages, such operations must be performed at query time. This has been prohibitively expensive, resulting in most systems precluding pixel-level techniques on large archives.

Feature-based search targets particular features, such as texture. The user specifies feature-based search parameters either by providing sample images, from which feature vectors for search are extracted, or by explicit specification of feature values or ranges. For example:

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

or

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

Feature-based objects are often pre-extracted, either by blocking the image, or employing clustering and segmentation techniques. Efficient indexing techniques such as R-trees can then be used to optimize the search for particular feature values.

A semantic search takes the form of (for example):

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

Semantic objects are typically predefined using classification algorithms on image features (such as texture or spectral histograms). Note that the primary distinction between semantic objects and feature objects is that semantic objects are labeled (with labels 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 each 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 desirable to support search at any of these abstraction levels.

At the time of this writing, there are numerous examples of systems that allow images or video indexing through the use of low-level image features such as shape, color histogram, and texture. Prominent examples for photographic images include IBM QBIC [2], the MIT PhotoBook [3], VisualSeek from Columbia University [4], and the Multimedia Datablade from Infomix/Mirage [5]. These techniques have also been applied to specific application domains, such as medical imaging [6, 7], art work[8] and video clips [9, 10, 11, 12, 13]. All these examples rely on a preprocessing stage in which appropriate features (such as shape, color histogram and texture) are extracted and indexed.

Although pre-extracted features and/or semantic objects allow for efficient indexing schemes, they fail to capture 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 NASA, Goddard Space and Flight Center. Our goal is to explore technologies that will facilitate the storage, query and retrieval of images by a diverse community of users from large digital libraries, and to demonstrate the use of these technologies in a functional testbed. It is our thesis that, although useful and necessary, the use of a predefined schema will be insufficient to adequately support content-based search. It is necessary to provide users the capability to define and extract features dynamically, and specify the target for content-based search interactively on the image data. To this end, we have developed an extensible framework where the objects to be searched are specified in terms of constraints at one or more different abstraction levels.

In this paper, we describe elements of our system that maximize retrieval efficiency, thereby enabling a broad range of query capability. In particular we propose a progressive framework that that is designed to combine data compression with image analysis in order to gain substantial speed-ups, In this progressive framework, the compression scheme decorrelates the information contained in the images and reorganizes it so that the analysis operators can effectively be applied on small, selected portions of the data.

To evaluate the effectiveness of the approach, we propose a set of benchmark queries. We observe that conventional implementations of image processing operators result in unacceptable execution times, while the progressive approach yields a significant increase in speed, that makes it appealing even in an interactive system. The net effect is to make a wider range of image processing and 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 set of queries that measure the benefits of our approach. Discussion and conclusions are in Section 6. The discussion in this paper is focused on image databases. Nevertheless, the concepts presented herein are also applicable to other multimedia databases.

# 2 System Architecture

In this section, we describe the architecture of our content-based search system. As shown in Figure 1, our system has a client-server architecture, with communication between client and server over the internet based on the http protocol.

The client program is a Java applet with allows the user to navigate through the image database, to construct content-based queries and to specify the format for visualizing query results.

The user can define new features and new object types by means of a feature/object definition module, and specify complex relations between simple objects using a relation definition module.

The server architecture (Figure 1) is centered around a query parser, that operates on features and objects that are both pre-extracted and computed at query execution time. The client issues queries that are translated into programs (written in an internal representation language) and parsed by the query parser. The control flow of object retrieval is diagrammed in figure 2. For each search operator, the query parser either retrieves pre-extracted objects, segments pre-extracted features into objects, or extracts features from objects and synthesizes object definitions.

Pre-extracted objects or features are stored in a database, currently DB2 version 2. The schema of the database is shown in Figure 3, and consists of the following three components:

- *Metadata table*, which contains information such as the coordinates of the bounding box of the image, instrument type, date of generation, and other relevant information. This information is used to perform initial pruning of the search space based on location, date, and resolution.

- *Object table*, which records the object type and geographic coordinates of all the pre-extracted objects.

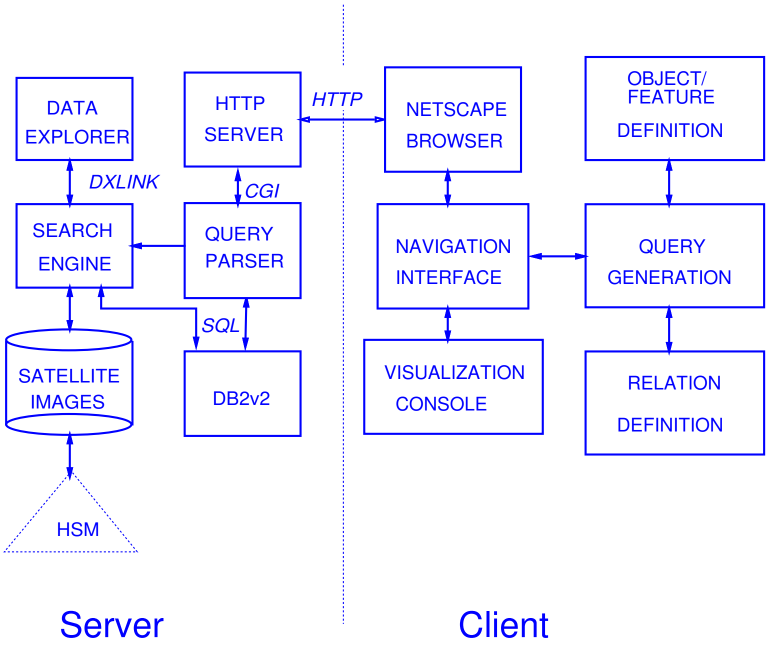


Figure 1: Architecture of the content-based retrieval system.

- *Feature table*, which records the feature vectors extracted from the images. Currently, the table stores eight different texture features including fractal dimensions.

When objects or features need to be synthesized at run-time, the query parser invokes the image search engine, which applies signal processing operators to the candidate images stored in the repository.

Each operator invokes the appropriate database and/or signal processing operators required to produce candidate objects for the search. Fuzzy boolean operators may be used to combine the ranked results from search operators into a final ranked list of object definitions, which are then used by the visualization engine to create output images.

# 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 results 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 that it is possible to increase the speed of searching through images

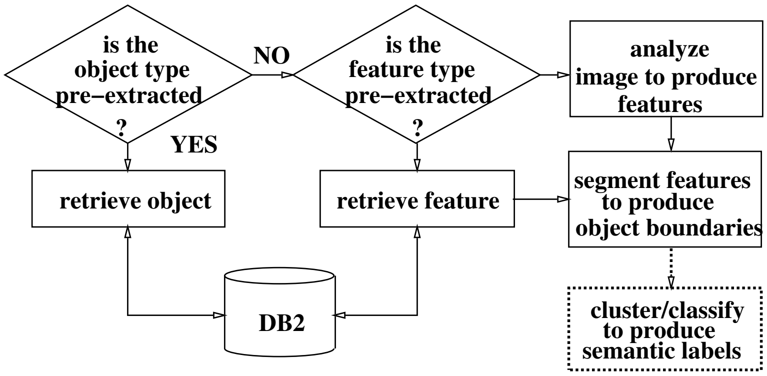


Figure 2: Object retrieval for a content-based query. If the objects are pre-extracted and indexed, the retrieval is translated into a SQL query. Otherwise, the objects are generated at query time: if the features used in the definition of the object are pre-extracted, they are retrieved from the database, otherwise they are computed from the image data. The features are then used to segment the image(s) into homogeneous regions (objects) and an optional classifier assigns semantic labels to each region. Similarity search is then performed on the preextracted objects, on the segmentation results or on the classification results.

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

Source coding can be categorized as lossless or lossy depending on whether the coding scheme preserves the original data. Transformed coding, such as discrete cosine transformation (DCT) used in both JPEG [14] and MPEG, and discrete wavelet transformation (DWT) [15], is commonly adopted as the first step for compression due to its capability of concentrating the information in the first few coefficients. A thresholding scheme can then be applied to eliminate the coefficients that are close to zero, and a quantization step is customarily employed to further reduce (compress) the information.

Quantization tables 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 I/O bandwidth is significantly reduced;
- The features and properties of the data can be emphasized by the transformed-based compression.

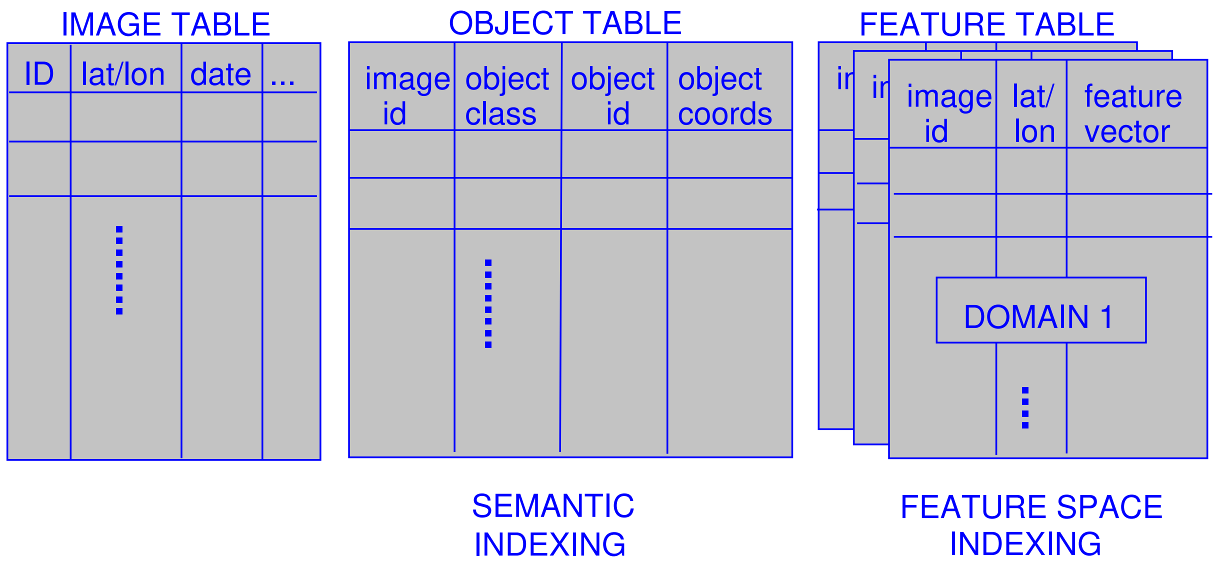


Figure 3: Schema of the database.

In particular, query operations (e.g., retrieval, evaluation, transmission and visualization) on image or video data can be staged progressively to minimize the total execution time by selectively and adaptively processing limited amounts of information. The execution schedule is adaptively determined to best benefit from specific properties of the image or video clip of the query.

## Wavelet-based compression

Our system uses a transform-based image coding scheme, called MCIA, that yields both lossless and lossy compression. The structure of our algorithm is shown schematically in figure 4. It consists of a lossy component and a lossless component; the lossy component uses Discrete Wavelet Transform coding (For a simple introduction to Wavelets and the Discrete Wavelet Transform, or DWT, see the article [15] 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 and the original image. The resulting *residual* image is also losslessly encoded [16, Chapters 5 and 6].

The transform coding is based on DWT; this step yields a multiresolution representation of the original image. We use short orthogonal or biorthogonal filters, [2] such as the Daubechies biorthogonal symmetric wavelets [17] of order 1 to 5, (although the system supports a large

[2]From a practical viewopoint, the wavelet filters we use are short Finite Impulse Response filters, and the filtering operation is the convolution between the original data and a short sequence, containing the impulse response of the filter.

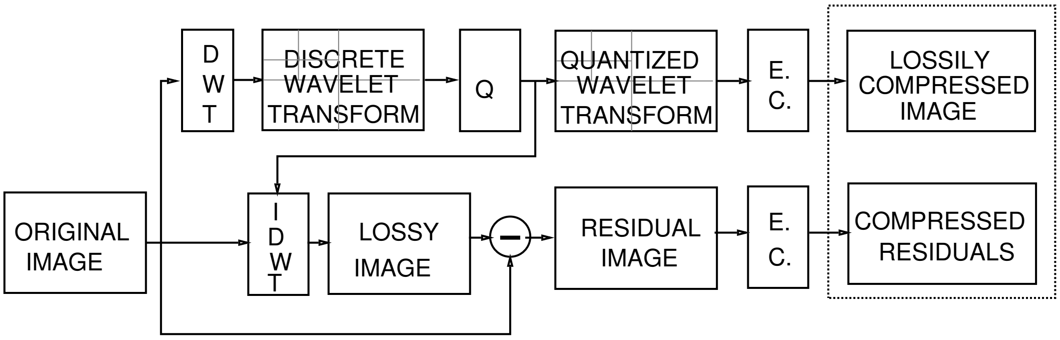


Figure 4: Overall structure of the compression algorithm. The image is transformed, quantized (**Q**), entropy coded (**E.C.**) and stored. The difference of the original and the lossy data (residuals,) is entropy coded and stored. The combination of compressed lossy data and residuals yields lossless compression.

variety of wavelets.) If the original data is stored in integer format, [3] these filters allow perfect reconstruction, i.e., the transform step is perfectly invertible.

We quantize [16, Chapter 13] this coefficient matrix using a uniform scalar quantization scheme. A different number of bits for each coefficient is allocated for each subband. Quantization results in loss of information: this step is the (only) lossy portion of our coding scheme. The quantized subbands are then independently losslessly encoded using predictive coding (DPCM [14, Chapter 5.2.1],) followed by a fixed-model, two pass, arithmetic coding [18, 14]. The result is a representation of the image from which we can reconstruct a lossy version of the original data.

The residual is computed by inverting the DWT of the quantized wavelet coefficients, and calculating the difference between that and the original image; the residual is then losslessly encoded, using DPCM followed by arithmetic coding.

We have extensively compared our compression algorithm with several existing lossles image coding schemes on heterogeneous satellite images and on classical test images such as Lena, and some of the results are summarized in figure 3. In general we have found the compression rate of MCIA closely compares to the compression rates of the commonly available lossless algorithms.

This scheme has several advantages. First, it has low computational complexity: the cost of decoding the data to the maximum resolution level (that is, reproducing the original image exactly) is linear in the number of pixels in the image. One can easily extract a multiresolution pyramid from the wavelet transform, essentially at the same cost of fully inverting the transform: as we will see later, this property makes the scheme appealing for applying search operators in a progressive fashion. By using short filters we can selectively reconstruct arbitrary portions of the image at the desired level of detail by accessing appropriate parts of the transform. This gives a significant speedup during the decoding process since, in

[3] 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.

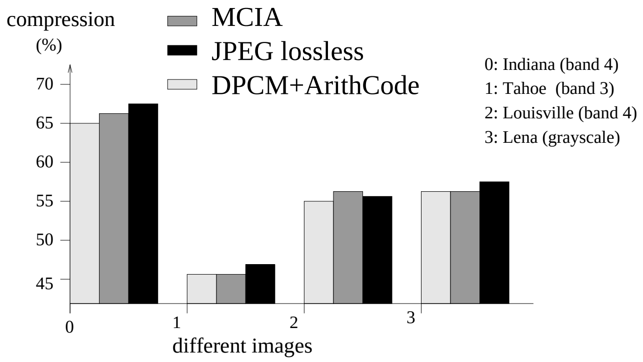


Figure 5: Comparison of MCIA with the lossless mode of the JPEG standard, and DPCM with arithmetic coding. The Indiana and Louisville images were obtained with the LANDSAT Thematic Mapper, the Tahoe image is a portion of a NALC dataset.

general, we do not need to analyze the whole image for content-based search purposes. The space-frequency representation induced by the wavelet transform simplifies the implementation of certain image processing operations, such as specific types of texture matching. Since the search algorithms can operate on the lossy and highly compressed version of the image, the scheme is amenable to implementations using hierarchical storage systems, where the relative high volume of residuals would be stored on tertiary (slow) media, and only accessed 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 the operators.

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

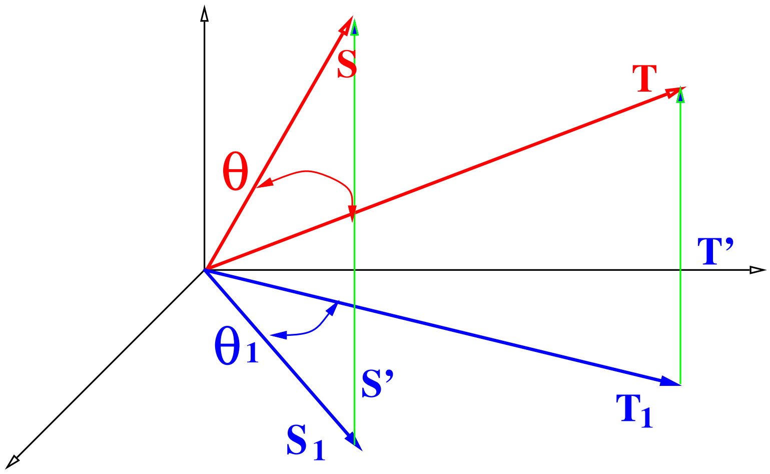


Figure 6: Geometrical interpretation of the progressive correlation coefficient algorithm.

## Template Matching

In a query-by-example framework [2], *template matching* [19, Chapter 7.5] allows the retrieval of images that contain exactly (or almost exactly) the example specified by the user. Indexing, in this context, is impossible, 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 $dx \times dy$) with each $dx \times dy$ 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 is a particularly resilient measure of the "distance" between the template and a subimage. Let $S$ denote the image, let $N$ and $M$ be the number of rows and of columns of $S$, and let $S_{m,n} = S_{m,n}(dx, dy)$ denote the subimage of size $dx \times dy$ starting at location $m, n$. The correlation coefficient is defined in the usual way, as the empirical cross-correlation between properly normalized versions of the subimage $S_{m,n}$ and of the template $T$,

$$\rho_{m,n} = \left[ \sum_{i=0,j=0}^{dx,dy} \left(T_{i,j} - \overline{T}\right) \left(S_{m+i,n_j} - \overline{S_{m,n}}\right) \right] \left[ \sum_{i=0,j=0}^{dx,dy} \left(T_{i,j} - \overline{T}\right)^2 \sum_{i=0,j=0}^{dx,dy} \left(S_{m+i,n_j} - \overline{S_{m,n}}\right)^2 \right]^{-\frac{1}{2}}$$

where $\overline{S}_{m,n} = \sum_{i=0,j=0}^{dx,dy} S_{m+i,n+j}/(dx\,dy)$. From the computational viewpoint, the set of quantities $\{\overline{S_{m,n}}\}_{m,n=0}^{M-dx-1,N-dy-1}$ can be computed in order $O(M \cdot N) + O(dx \cdot dy)$, which, for very large images, is essentially $4MN$; The quantities $\overline{T}$ and $\sum_{i=0,j=0}^{dx,dy} \left(T_{i,j} - \overline{T}\right)^2$ require $O(dx \cdot dy)$ operations. Thus, the real computational cost is associated with the correlation at the numerator of $\rho_{m,n}$: by using the convolution theorem for the Fourier Transform [20], we can compute the correlation coefficient in order $O(N \cdot M \cdot \log M \log N)$ and better algorithms exist only if $dx \cdot dx < \log M \log N$. Thus, the computational cost of the correlation coefficient is superlinear in the size of the image.

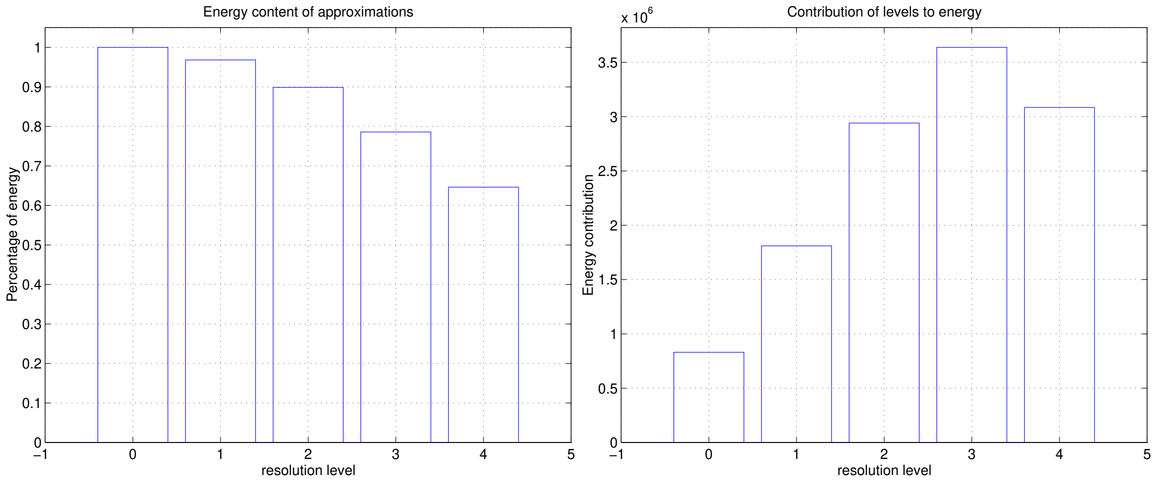


Figure 7: The energy distribution (left) of the wavelet transform of an image, as a function of the multiresolution level. The figure on the right shows that the approximation at level 6 already captures most of the overall energy: the wavelet transform concentrates most of the energy of real images in very few coefficients, and the contributions due to lower levels decreases quickly.

Progressive template matching allows one to approximate $\rho$ without having to consider all of the coefficients. The approach consists of computing the correlation coefficient on a low-resolution version of the image first, and refining progressively the results only around local maxima. This reduces significantly the computational complexity, which, for all but very small templates, is of the order $O\left(nm \log n \log m\right)$, where $n$ and $m$ are the numbers of rows and of columns in the target image. The theoretical foundations of the approach rely on the results in [21] 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 of the approach. If we think of the normalized subimage and template as vectors in a $n^2$-dimensional Euclidean space, then the correlation coefficient can be interpreted as the cosine of the angle $\theta$ between these two vectors (for instance $\mathbf{T}$ and $\mathbf{S}$ in figure 4.) In this scenario, an exact match yields the maximal correlation coefficient of 1. The level-1 approximations to the image and the template are the projections $\mathbf{T}_1$ and $\mathbf{S}_1$ of $\mathbf{T}$ and $\mathbf{S}$ onto an appropriate subspace defined by the wavelet decomposition. Similarly, the level-$i$ approximations can be represented by vectors $\mathbf{T}i$ and $\mathbf{S}_i$ in the subspace $W_i$. The correlation coefficient at level 1 is the cosine of the angle $\theta_1$. If the the difference vectors $\mathbf{T}^{'}$ and $\mathbf{S}^{'}$: are short, then we can closely approximate $\theta$ with $\theta_1$. On the other hand, if the difference vectors capture a significant portion of the energy, the approximation is poor. The energy of a typical normalized image is distributed across the different resolution levels as shown in figure 7: It is apparent that the level-3 approximation captures almost 80% of the energy of the normalized template, that is, the length of the vector $T^{'}$ in figure 4 about half the length of $T_3$. We have developed techniques that allow us to quantify the effects of approximating the angle $\theta$ with $\theta_i$ without having to convert back to the spatial domain [22]. Then, at each stage $i$ of the progressive template matching, we can identify candidate regions to be analyzed at the immediately higher resolution and 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 our experiments indicate an expected speedup of between 20 and 50. We have observed that the technique is very robust to sensor noise: it is essentially insensitive to the noise levels usually observed in satellite images. Also, the very nature of the correlation coefficient 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

The texture features that are considered in our system include:

- Fractal dimension,
- Coarseness,
- Entropy,
- Circular Moran autocorrelation functions,
- Spatial grey-level difference (SGLD) statistics.

The reason for choosing these features is to facilitate the extraction of the most prominent texture features at various resolution, which is essential in the progressive feature extraction algorithm.

Fractal dimension of a random function $I(x)$ is defined as $T+1-H$ where $T$ is the topological dimension of $I(x)$, while $H$ is a parameter of the fractal Brownian function $I(x)$ if for all $x$ and $\Delta x$:

$$Pr\left(\frac{I(x+\Delta x)-I(x)}{\|\Delta x\|^H} < y\right) = F(y) \tag{1}$$

is satisfied. Fractal dimension one means of measuring disordered texture. In particular, it is useful to measure the surface roughness. It is chosen in our system because of its invariance to linear transformation of the data and the transformation of the scale. For our system, the reticular cell counting method [23], in which the number of cubes at different scales that are passed by the image surface are counted, is used to compute the fractal dimension.

Coarseness is defined as

$$C = 1 - \frac{1}{1+S_D} \tag{2}$$

It is related to the dispersion of the image, $S_D$, defined as

$$S_D = \sum_{i=0}^{L-1} (i - S_M)^2 h[i] \tag{3}$$

where $S_M$ is the mean of the histogram, $h[i]$, of the image, and $L$ is the number of levels used in accumulating the histogram.

Entropy is defined as

$$S_E = -\sum_{i=0}^{L-1} h(i) log_2 h[i] \tag{4}$$

Spatial gray-level difference based statistics (SGLD) are commonly used for extracting texture from remote-sensed images [24, 25]. Let $G(m,n)$ be the gray level image within a block, then for any displacement $(\Delta_m, \Delta_n)$, we can define the gray-level difference as:

$$G_{\delta,\theta}(m,n) = |G(m,n) - G(m + \Delta_m, n + \Delta_n)| \tag{5}$$

where $\Delta_m$ and $\Delta_n$ are integers. Possible choices of $\Delta_m$ and $\Delta_n$ are $-d \leq \Delta_n, \Delta_m \leq +d$, so that the direction $\theta$ in which the pixel pairs are measured may be 0, $\pi/4$, $\pi/2$, $3\pi/4$ when $d = 1$. From the Gray level difference of a region, the histogram $h_{\delta,\theta}[i]$ associated with a specific value of $(\delta, \theta)$ can then be determined. In other words,

$$h_{\delta,\theta}[i] = \sum_m \sum_n count(G_{\delta,\theta}(m,n) = i) \tag{6}$$

This histogram function can then be used to define various SGLD-based statistics:

$$Mean(\delta,\theta) = \frac{1}{G} \sum_{i=0}^{G-1} i \frac{h_{\delta,\theta}[i]}{P} \tag{7}$$

where $G$ is the number of gray levels in the image (256 for one-byte pixels), and $P$ is the total number of pixel pairs that are separated by distance $\delta$ in direction $\theta$.

Similarly,

$$Contrast(\delta,\theta) = \sum_{i=0}^{G-1} i^2 \frac{h_{\delta,\theta}[i]}{P} \tag{8}$$

$$Angular\ Second\ Moment(\delta,\theta) = \sum_{i=0}^{G-1} (\frac{h_{\delta,\theta}[i]}{P})^2 \tag{9}$$

$$Entropy(\delta,\theta) = -\sum_{i=0}^{G-1} \frac{h_{\delta,\theta}[i]}{P} log(\frac{h_{\delta,\theta}[i]}{P}) \tag{10}$$

A set of directionally unbiased spatial autocorrelation functions called *circular Moran autocorrelation function* [25] are also used in our system. Denoted by $Circ_i$ $(i = 1,..,4)$, the autocorrelation of the mean-corrected image pixels in the $i^{th}$ circular band is evaluated by multiplying each value by the central value before summation. The structure of the $i^{th}$ circular band is:

$$\begin{array}{ccccccccc} - & - & 4 & 4 & 4 & 4 & 4 & - & - \\ - & 4 & 3 & 3 & 3 & 3 & 3 & 4 & - \\ 4 & 3 & 2 & 2 & 2 & 2 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 1 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & * & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 1 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 2 & 2 & 2 & 2 & 3 & 4 \\ - & 4 & 3 & 3 & 3 & 3 & 3 & 4 & - \\ - & - & 4 & 4 & 4 & 4 & 4 & - & - \end{array} \tag{11}$$

Finally, the sum is divided by the sum of squares of the central pixel values and by the number of image pixels within the band in order to normalize the value of the circular autocorrelation functions. As an example, the circular autocorrelation of the first band can be expressed as:

$$\begin{aligned} Circ_i = \sum_i \sum_j G(i,j)\{G(i-1,j-1) + G(i-1,j) + G(i-1,j+1) + \\ G(i,j-1) + G(i,j+1) + G(i+1,j-1) + G(i+1,j) + \\ G(i+1,j+1)\}/(8 \sum_i \sum_j G(i,j)^2) \end{aligned}$$

From the procedures listed in Section 2, it is apparent that the texture extraction step is the most time-consuming step, as this step has to be applied to all the regions of all the images in the database. For interactive situations in which the features cannot be precomputed, such as content-based filtering of real-time images and video, feature extraction becomes the major bottleneck. Therefore, it is desirable to speed up the feature extraction step so that the texture matching operation scales well with the size of the database. One way of speeding up the texture extraction process is to apply the texture extraction and comparison algorithm progressively to the given images from the database. We assume that a progressive representation of the images and the videos already exists. A progressive representation of an image and video can be obtained from applying subband coding or wavelet transformation to the image or video. The progressive version of the image retrieval algorithm based on texture features is as follows:

1. Divide the transformed images into regular or irregular regions at each resolution according to the boundaries extracted by edge detection or image segmentation. We assume that the segmentation at each resolution is similar. As a result, a $B \times B$ block at level 0 corresponds to a $B/2 \times B/2$ block at level 1, and corresponds to a $B/4 \times B/4$ block at level 2.

2. Compute the normalized texture feature vector of each region at the $L^{th}$ resolution level for each block of the image using the same techniques outlined in Section 2. $L$ is

the starting resolution of the image that the progressive image retrieval algorithm is applied.

3. Compute the normalized texture feature vector of each block of the target template at the $L^{th}$ resolution level using the same techniques outlined in Section 2.

4. Compute the similarity between the blocks in the target template and the blocks from the image at resolution level K. The candidate regions of blocks are then sorted according to a distance metric between the feature vectors of the target template and that from the image using the same techniques outlined in Section 2. A total of $R$ regions whose feature vectors are closest to those extracted from the target template are selected for further comparison at the next higher resolution. The value $R$ is determined by the final number of matches, $Q$, required by the query. In general, $R \geq Q$ in order to ensure that the final results are a subset of the intermediate results.

5. The feature vector of those regions at resolution level $L-1$ are then computed and compared to those extracted from the target template at the same level.

6. This process is continued until level 0 of the image is reached.

## Classification

Classification of a multispectral image is the process of assigning labels to the individual pixels or to distinct regions [26, Chapter 8]

In a wavelet multiresolution framework, we can consider a coefficient at level $\ell$ "essentially" as the low-resolution representation of a block of $2^\ell \times 2^\ell$ pixels at level 0. A progressive classifier operates as depicted in figure 8: first the approximation at a predefined resolution $2^{-L}$ of the image is labeled with an appropriately trained classifier, which decides whether each coefficient $w_{i,j}^L$ corresponds to a homogeneous or to an heterogeneous $2^L \times 2^L$ block of pixels in the untransformed image. When the classifier decides that the coefficient corresponds to a homogeneous regions of pixels of class $k$, all the pixels at full resolution that belong to this region are labeled with the label $k$. Otherwise, the transform is inverted locally, to yield 4 coefficients at the immediately higher level, and the classifier analyzes them independently. The recursive process terminates either when a homogeneous block is detected or when full resolution is reached.

As proved in [27], this scheme has two advantages: it is faster than the full-resolution, pixel-by-pixel approach, since in natural images there is significant correlation between the labels of adjacent pixels, allowing large portions of the image to be classified at low resolution, and it is more accurate than the pixel-by-pixel approach.

The main downside is that the results of the classification sometimes appear somewhat blocky, especially near the borders between regions of spectrally similar classes.

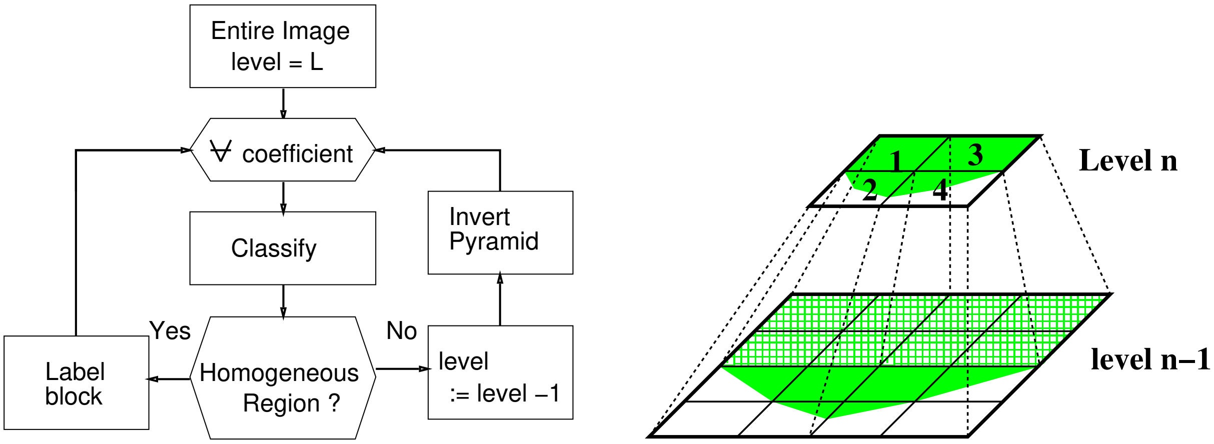


Figure 8: Schematic algorithm of progressive classification (left) and two-class example (right). The example depicts two adjacent levels of the multiresolution pyramid. At level $n$ we consider 4 pixels, of which pixels 1 and 3 correspond to a homogeneous region of class 1 in the full-resolution image. Then the progressive classifier labels the corresponding blocks at level $n-1$ as class 1. Pixels 2 and 4 correspond to a non-homogeneous region, and are expanded into 4 pixels and analyzed at level $n-1$.

Note that no specific classification scheme is mentioned in the description of the algorithm. Indeed, progressive classification is not a new classifier, rather, it is a framework that can be used with different types of classifiers.

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 Gaussianity, also known, regrettably, as Maximum Likelihood classifier), k-Nearest Neighbor, Learning Vector Quantization, clustering-based schemes [19], and CART (Classification and Regression Trees) [28].

Our experiments have shown that the progressive approach is substantially faster than the straightforward classification of the original image. For instance, if the process is started at level $L = 2$, the classification time is 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 [29].

Further speedups can be achieved by ending the process at a higher level than the full resolution: as seen in figure 9 the increase in error rate from one resolution level to the next is in most cases limited, making this a viable option when the user is willing to trade accuracy for response time.

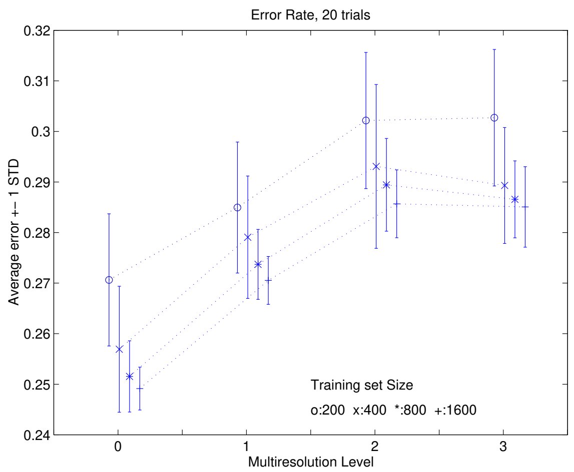


Figure 9: Accuracy of progressive classification as a function of the resolution level and of the training set size. The figure depicts the mean error rate and the standard deviation estimated over 20 experiments with different training sets and test images.

# 5 Evaluating the approach

To our knowledge, there are no standardized benchmarks for content-based search that would appropriately capture the benefits of our approach. Consequently, we have defined in collaboration with NASA the following set of benchmark queries

1. Determine the best match in an image $I$ for a given template $Te$ (such as an airport, GCP, etc) and the location(s) of the match.
2. Given a texture pattern $T_x$, find the two images containing textures that best match $T_x$. The texture similarity criterion is defined statically.
3. Return a subimage with size $(dx, dy)$ with upper-left corner at location $(x, y)$ in image A, composited with another image with size (du,dv) with upper left corner at location $(u, v)$ in image B.
4. Classify scan S using USGS classification as a base and return centroid of the largest body of water.
5. Find all images with less than $p\%$ cloud cover.
6. Find all locations of urban areas adjacent to forest. By definition two regions are adjacent if their boundaries interst.
7. Find the rate of change of all 9 USGS land usage categories in fixed region(s) between 1970 and 1990.

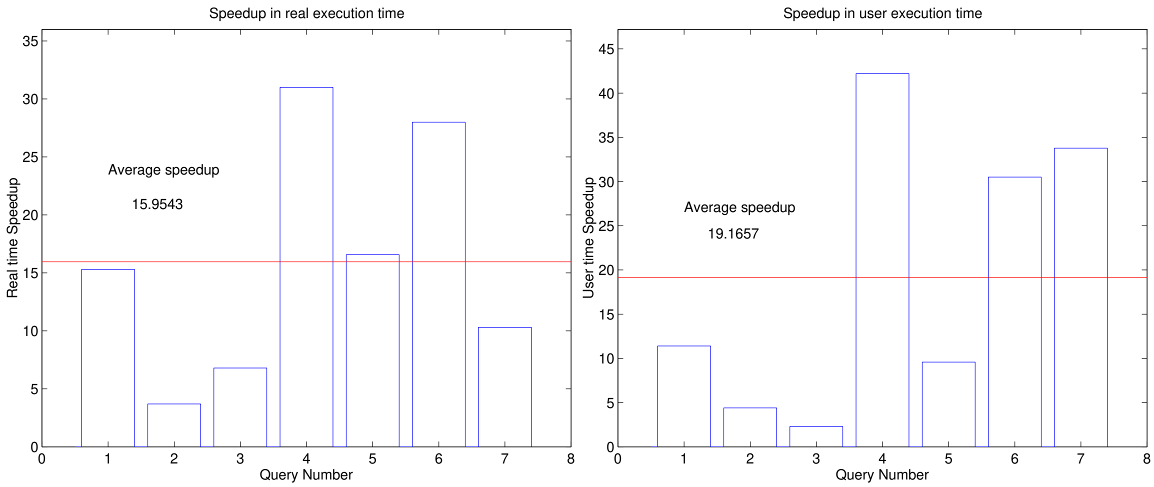


Figure 10: Speedups in real time (left) and execution time (right) for each of the 7 benchmark queries.

For each query we define a set of input images, a set of constraints and the format of the results. To measure the performances, each query is executed both in the compressed domain, from within our system, and on the original images, as a stand-alone process. The original images are stored in uncompressed format; for multispectral data, each spectral band is stored separately in a flat file. No precomputed medatada or preextracted 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 purpose they are replaced by the specification of the input dataset.

The results of an operation in the compressed domain is usually different from the result of the corresponding operation on the original images. This is due in part to the loss of information resulting from the lossy nature of the compression scheme, in part to inherent differences in the algorithms. In order to compare the two approaches, we impose the following requirements on the query results: if a query returns a (possibly ranked) list, both baseline and compressed domain 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 measure the user and the real execution times, for both the baseline and the compressed domain implementation. A real time and a user time speedups are 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. By taking the averages of the ratios over the benchmark queries, we obtain two measures of the benefits of our approach.

Figure 10 shows the results of applying the compressed domain approach to the benchmark queries. The experiments were run on an IBM computer model 590 with 256MB of RAM and on an IBM computer model 43P with 160MB of RAM. The real and user times were measured using the UNIX `time` command. The user time is a measure of the CPU time

| Query number | image size | baseline | | progressive | |
|---|---|---|---|---|---|
| | | user time | real time | user time | real time |
| 1 | $2815 \times 3082$ | 93.51 | 277.03 | 8.20 | 18.11 |
| 4 | $2815 \times 3082$ | 208.96 | 315.15 | 4.94 | 10.18 |
| $5^*$ | $1400 \times 1470$ | 111.90 | 242.15 11.68 | 26.19 | |
| 6 | $2815 \times 3082$ | 145.06 | 242.15 | 4.76 | 8.64 |
| 7 | $1312 \times 1312$ | 26.69 | 29.40 | 0.79 | 2.85 |
| $7^+$ | $4179 \times 3988$ | 297.42 | 7807.04 | 10.52 | 15.27 |

$*$ The reported times are for the analysis of 17 images of size $1400 \times 1470$.

$+$ The query involves the analysis of two images

Figure 11: Measured times in seconds for queries involving classification and template matching. The input image sizes are specified. The images for query 1, 4 and 6 have seven spectral bands, while the images used in query 7 have four spectral bands. Note that the real execution time when the progressive version of query 7 is applied to the smaller image is dominated by the application loading time, while the real execution time when the baseline implementation is applied to the larger image is due to page faults.

spent in executing the user code in user space. Since the computers were devoted only 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. The algorithms used in the baseline and progressive implementations are 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 significantly impair the execution. Under such conditions, speedups of 500 were observed; an example is the first results for query 7 in figure 11. 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 a worst-case scenario 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 are reduced to few seconds or fraction of second per (large) image. In some instances we have observed smaller benefits, most noticeably, in queries number 2 and number 5. Query number 2 requires very simple operations, thus it is easily performed on the full resolution image. The overheads of the progressive implementation (database access, entropy decoding, 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 number 5 estimates the cloud cover percentage and returns only the images with less than $p\%$ clouds. The reported execution times are the averages over 4 different values of $p$, namely 10, 20, 30 and 40. The algorithm estimates $p$ and produces a 95% confidence 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, most of which had cloud cover between 30% and 50%. Then, for $p = 30$ and $p = 40$ 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 user groups of satellite imagery with which we have interacted have presented us with numerous, heterogeneous definitions of content. In such situations, preextracting information a nd generating indexes becomes too onerous 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, extremely expressive, 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 indices 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 preextracted features are used to prune the search space, usually allowing the search engine to restrict the 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. In general this process has been regarded as impractical, especially in an interactive system, due to 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 "classical" and the progressive implementations of the algorithms. The observed speedups are significant: the computational cost has been reduced on the 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: 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 number 7.

We have demonstrated the capability of extracting user-specified features at query time and of using these feature to search for content, thus adding a new dimension to the flexibility of our query engine. We have shown the ability to analyze a large image (of size $3000 \times 3000$ pixels and 7 spectral bands, or $4000 \times 4000$ pixels and 4 spectral bands) in less than 10 seconds of CPU time and less than 20 seconds of total response time, while the analysis of smaller images ($1300 \times 1300$ pixels and 4 spectral bands or $1400 \times 1400$ pixels and 2 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 limited computational cost.

**Acknowledgements:** We would like to acknowledge Loey Knapp, Steven Carty, Satheesh Gannamraju, John Gerth, Sharmila Hutchins, Richard Ryniker, Jerald Schoudt, Barbara Skelly, Bernice Rogowitz, Alex Thomasian, Lloyd Treinish and Joel Wolf for their contribution during the development of the system.

# References

[1] J. C. Tilton and M. Manohar. Earth science data compression issues and activities. *Remote Sensing Reviews*, 9:271–298, 1994.

[2] W. et al. Niblack. The qbic project: Querying images by content using color texture, and shape. In *Proc. SPIE*, volume 1908, Storage Retrieval for Image and Video Databases, pages 173–187, 1993.

[3] A. Pentland, R.W. Picard, and S.S Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proc. SPIE*, volume 2185, Storage and Retrieval for Image and Video Databases, pages 34–47, February 1994.

[4] J.R. Smith and S.-F. Chang. Visualseek: A fully automated content-based image query system. In *Proc. International Conference on Image Processing*, Lausanne, Switzerland, 1996.

[5] J.R. Bach and al. The virage image search engine: An open framework for image image management. In *Proc. SPIE*, volume 2670, Storage and Retrieval for Still Image and Video Databases, pages 76–87, 1996.

[6] T.Y. Hou, P Liu, A. Hsu, and M. Y. Chiu. Medical image retrieval by spatial feature. In *Proc. IEEE International Conference on System, Man, and Cybernetics*, pages 1364–1369, 1992.

[7] T.Y. Hou, A. Hsu, P Liu, and M. Y. Chiu. A content based indexing technique using relative geometry features. In *Proc. of SPIE*, volume 1662, Image Storage and Retrieval Systems, pages 29–68, 1992.

[8] B. Holt and L. Hartwick. Visual image retrieval for applications in art and art history. In *Proc. SPIE*, volume 2185, Storage and Retrieval for Image and Video Databases, pages 70–81, February 1994.

[9] E. Deardorff, T. D. C. Little, J. D. Marshall, and D. Venkatesh. Video scene decomposition with the motion picture parser. In *Proc. SPIE*, volume 2187, Digital Video Compression on Personal Computers: Algorithms and Technologies, pages 44–55, February 1994.

[10] H Zhang and S. W. Smoliar. Developing power tools for video indexing and retrieval. In *Proc. SPIE*, volume vol. 2185, Storage and Retrieval for Image and Video Databases, pages 140–149, February 1994.

[11] H Zhang and S. W. Smoliar. Content based video indexing and retrieval. In *IEEE Multimedia*, volume 1, nr 2, pages 62–72, Summer 1994.

[12] F. Arman, A. Hsu, and M. Y. Chiu. Image processing on compressed data for large video database. In *Proc. ACM Multimedia 93*, pages 267–272, 1993.

[13] F. Arman, A. Hsu, and M. Y. Chiu. Feature management for large video database. In *Proc. SPIE*, volume 1908, pages 2–12, 1993.

[14] William B. Pennebaker and Joan L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.

[15] Oliver Rioul and Martin Vetterli. Wavelets and signal processing. *IEEE Signal Process Magazine*, 8(4):14–38, October 1991.

[16] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley and Sons, 1991.

[17] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[18] Ian H. Witten, Radford M. Neal, and Cleary John G. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[19] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[20] Ronald N. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill Electrical and Electronic Engineering Series. McGraw-Hill, 1978.

[21] P.P Vaidyanathan. Orthonormal and biorthonormal filter banks as convolvers, and convolutional coding gain. *IEEE Transactions on Signal Processing*, 41(6):2110–2130, June 1993.

[22] C.-S. Li, J. J. Turek, and E. Feig. Progressive template matching for content-based retrieval in earch observing satellite image databases. *Proc. of SPIE Photonics East*, November 1995.

[23] A. R. Rao. *A Taxonomy for Texture Description and Identification.* Springer-Verlag, New York, 1990.

[24] J. Weszka, C. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classifications. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-6(4):269–285, 1976.

[25] Z. Q. Gu, C. N. Duncan, E Renshaw, C. F. N. Mugglestone, M.A.and Cowan, and P. M. Grant. Comparison of techniques for measuring cloud texture in remotely sensed satellite meterorological image data. *IEE Proceedings*, 136(5):236–248, October 1989.

[26] John A. Richards. *Remote Sensing Digital Image Analysis.* Springer-Verlag, 1993.

[27] Vittorio Castelli, Ioannis Kontoyiannis, Chung-Sheng Li, and John J. Turek. Progressive classification: A multiresolution approach. *Research Report nr RC 20475*, 06/10/1996.

[28] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth & Brooks/Cole, 1984.

[29] Vittorio Castelli, Chung-Sheng Li, John J. Turek, and Ioannis Kontoyiannis. Progressive classlification in the compressed domain for large eos satellite databases. *Proc. of 1996 IEEE International Conference on Acoustics, Speech and Signal Processing*, 4:2201–2204, May 1996.