RC 20583 (91227) (10/17/96)
Computer Science/Mathematics

IBM Research Report

Progressive Search and Retrieval in Large Image Archives

L. Bergman, V. Castelli, S. Gannamraju, C.-S. Li, J. Robinson, R. Ryniker, J. Turek,

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

S. Carty, S. Hutchins, L. Knapp, J. Shoudt, B. Skelly,

IBM Government Systems
6300 Diagonal Highway
Boulder, CO, 80301

J. Gerth, I. Kontoyiannis

Department of Electrical Engineering
Stanford University
Stanford, CA 94305

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Progressive Search and Retrieval in Large Image Archives [1]

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

S. Carty
scarty@betvmic1.vnet.ibm.com
(303)924-0168

V. Castelli*
vittorio@watson.ibm.co
(914) 784-7665

S. Gannamraju
c1sat@watson.ibm.com
(914) 784-7467

J. Gerth
gerth@watson.ibm.com
-

S. Hutchins
sharmila@betvmic1.vnet.ibm.com
(303) 924-0647

L. Knapp
knappl@betvmic1.vnet.ibm.com
(303) 924-0482

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

R. Ryniker
ryniker@watson.ibm.com
(914) 784-7954

J. Shoudt
jschoudt@betvmic1.vnet.ibm.com
(303) 924-0614

B. Skelly
skelly@betvmic1.lexington.ibm.com
(303) 924- 0646

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

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 at the IBM T.J. Watson Research Center 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 the image processing algorithms required in extracting the content during the query execution. As a result, this framework is more scalable to a large image archive.

1 Introduction

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. 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. These raw data generated by various EOS platforms will be processed and stored in distributed active archive centers (DAACs) located throughout the United States [1]. Other examples are in the seismic and medical imaging areas in which terabytes of data are continuously acquired and stored. As a result, 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 the detail of the image and video data, it would be 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.

Most existing image or video databases fall into one of the following three categories based on the mechanisms used to provide content search:

- *Flat file:* In a flat file database each image is treated as a single array attribute or binary large object (BLOB); the database does not have any knowledge about the semantics of the image [2]. Content-based queries such as template matching can be executed as user-defined functions and are performed through all of the images in the database. Due to the sequential nature of the search, this type of database does not scale well with either the size of the images or the number of images stored in the database.
- *Graph-based:* In a graph-based database, the semantics of each image is extracted through a set of sophisticated image understanding processes. This information is then stored as a semantic graph and content-based queries are translated into membership or similarity search on the graphs [3]. This type of database scales well with both the size of the images and the number of entries in the database. However, since the semantics are defined by the graphs that represent the images, this approach is effective for domain specific problems at best. Graph-based image databases are sometimes combined with model-based object generation, in which additional content of the image (typically shape) is discovered through the guidance of a set of preclassified models [4].
- *Feature-based:* In a feature-based database, features of the image such as texture, shape, and color are extracted and stored as attributes of the image. Content-based queries can then be translated into feature matching. This approach has been demonstrated as an effective means for retrieving photographic images such as the IBM QBIC [5]. It has also been successfully demonstrated for medical images [6], art work [7] and satellite imagery [8]. This concept has also been applied to video browsing. Feature-based databases can be further classified as *relational*, *spatial* or *object-oriented* depending on the methodology that is used to provide indexing on the attributes. [9].

In general, there exist three different levels of abstraction at which image or video objects can be defined and searched, namely, pixel, feature and semantic level. Most of the existing image databases only allow the objects to be search at one level. Feature matching at the feature level is more efficient than,

for instance, template matching at the pixel level since there exist very efficient indexing techniques (such as R-tree) to search the pre-extracted features. However, feature-based search alone is not sufficient. The feature extraction process is by its very nature lossy and cannot represent all of the content contained in the image. Furthermore, the processing involved in deriving the attributes can be quite expensive; one needs techniques to make this processing more efficient.

This paper describes a project currently in progress at the IBM T.J. Watson Research Center under joint sponsorship by NASA, Goddard Space and Flight Center that proposes to explore some of these challenges. In this paper, we describe the architecture and implementation of an extensible framework where the object(s) to be searched is specified in terms of constraints at one or more different abstraction levels. From the analysis of the performance of a complex query, we conclude that most commonly used features, feature indices, and object semantics should be pre-extracted and stored in a database to maximize the retrieval efficiency. Meanwhile, user-defined features and objects is still necessary at query time to maximize the flexibility and extensibility. The query-time content extraction is performed by means of a sequence of image processing operators. These operators can be computationally expensive, especially when applied to large satellite images. To overcome this problem, we propose a progressive framework that combines data compression, which is a basic requirement due to the large amount of data typically in this kind of environment, and image analysis. 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. This approach yields a significant increase in speed over the more conventional implementations, that operate on the full images.

The remainder of this paper is organized as follows: Section 2 describes the abstraction levels in content-based search. The current system architecture is discussed in Section 3. Section 4 describes data representation and outlines the ways in which we use data representation to make the processing more efficient. Sections 5 and 6 are devoted to the navigation and content-search operators. Finally, in Section 7 we conclude the paper. The discussion in this paper is focused on image databases. Nevertheless, the concepts presented herein are also applicable to other multimedia databases.

2 Preliminary

Content-based retrieval of image and video databases usually involves comparing a query object (also called *target* object), with the objects stored in the data repository. The comparison is usually based on *similarity* rather than on exact match, and the retrieved results are ranked according to a *similarity metric*. For instance, when objects are represented as vectors of features (such as $\{u_i\}$ and $\{v_i\}$, the most commonly used similarity metric is the Euclidean distance between these vectors:

$$D = \sum_{i=1}^{n} (u_i - v_i)^2 \tag{1}$$

or, in general, the L^p distance metric which is defined as

$$D = \sum_{i=1}^{n} (u_i - v_i)^p \tag{2}$$

The objects of an image or video database can be represented at different abstraction levels, as shown in Figure 1.

1. *Raw Pixels*: The lowest abstraction level refers to the raw pixels from the image. Comparison between objects or regions is done pixel-by-pixel and common indicators of similarity are the correlation and the L^p distances. Similarity is then defined in terms of threshold: an image region is considered as a candidate when its correlation with the template is larger than the threshold, or when its L^p distance is smaller than the threshold. Comparison at pixel level is very specific, and therefore is only used when a relatively precise match is required.

2. *Feature*: Features are often used to represent images at the next higher abstraction level. Each image can usually be segmented into regions consisting of homogeneous spectral or texture features. Similarity search in the feature space thus consists of comparing the target feature with the stored features in the database. These features can be predefined and pre-extracted, or user-defined and pre-extracted, or even user-defined and extracted at query time. Both global and local features such as texture, spectral histogram, and wavelet or Fourier coefficients of an image region have been used for characterizing an image region. Various spatial indexing schemes also exist to facilitate feature space indexing.

3. *Semantic*: This is the highest abstraction level at which a content-based search can be performed. Semantic information from an image is usually

extracted from a pre-trained classifier or supplied through human interpretation. This information could include the type of land cover of a specific area such as water, forest, or urban. A semantic network can be constructed to assist generalization.

Note that an object in a content-based query can be defined by means of constraints at all of the abstraction levels. As a result, the ranked results from the similarity search at each individual abstraction level need to be combined. A simple linear combination of the similarity (or distance) metrics can be used as final similarity metric for a search involving D abstraction levels,

$$S_F = \sum_{i=1}^{D} w_i S_i, \tag{3}$$

where S_F is the final score, w_i is the weighting of each abstraction level, and S_i is the score at the i^{th} level.

3 System Architecture

In this section, we describe the architecture of the content-based search system which is currently being developed at IBM T. J. Watson Research Center. As shown in Figure 2, our system has a client-server architecture, and the communication is based on the http protocol.

When the URL of the image server address is specified on a browser (such as the netscape browser), a Java applet is downloaded to the client site. The client interface serves the purposes of allowing the user to navigate through the image database, to construct content-based queries and to specify the format used in visualizing the query results.

The user can visually navigate at different resolutions through the stored images in the repository using the geographical coordinates, the instrument, the time of acquisition and the spectral response of the sensor as selection criteria.

Content-based image queries are specified through a drag-and-drop interface, and are expressed in a quasi-natural language. This interface performs syntax checking during the query construction and only syntactically correct query strings can be submitted to the server across the Internet.

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 the relation definition module.

The typical content-based query that our system is designed to satisfy is, for example, *Find me all the bodies of water that are within 50 kilometers of*

any fire where the fire started less than 3 days ago and the diameter of the fire is more than 100 kilometers and the fire is in the selected geographical region.

In this example, *body of water* and *fires* are objects, that could be extracted through the use of a pretrained classifier, but, as mentioned, a query can also include user-defined objects and features.

Thus, a typical query usually consists of constraints on

- *metadata*, specified for instance, as the temporal range between two dates or as type of sensor that acquired the data;
- *predefined image objects*, such as fires and body of water;
- *user-defined image objects using predefined features*, for instance "forest at high risk of fire" can be defined as areas with land cover type equal to forest, with relative greenness index very low and with fire weather index high (note that here the different features "land cover type", "relative greenness index" and "fire weather index" could be obtained from different data sources);
- *user-defined image objects using user-defined features*, such as the class labels of a user-trained classifier, and
- *spatial or temporal relationships among objects* such as the forest A is upwind from the fire B.

The user can therefore specify content at all three levels of abstraction defined in section 2, and can generate arbitrarily complex queries.

Apparently, it is not feasible to extract a priori all possible objects from all the images, store them in a database and generate appropriate indexes. On the other hand, extracting all the desired features only at run time from every image that satisfies the basic metadata requirements, whenever a new query is issued, is clearly impractical, extremely time consuming, and prevents an interactive use of the system.

Our solution consists of combining indexing on feature and object extracted a priori, with query-time image analysis.

The server architecture (Figure 2) 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 query parser also issues database queries involving metadata to our database engine in the form of SQL statements. The current version of the system interfaces with DB2 version 2. These queries, by identifying a set of candidate images, effectively result in an initial pruning of the search space,

We pre-extract the most commonly used and most descriptive features and objects, that will allow for an efficient pruning of the search space, we store them in a database, and construct appropriate (multidimensional) indexes. The search engine translates queries involving pre-extracted objects into SQL statements, and issues them to DB2.

For all other features and objects, e.g., the user-defined new object classes, the query parser invokes the image search engine, that applies signal processing operators to the candidate images stored in the repository, computes the desired feature and detects the relevant objects at query time.

We have developed a framework that combine appropriately modified image processing algorithms with the data representation resulting from our compression scheme, in order to achieve a significant speedup and efficiency. The accuracy of the resulting operators is always comparable and sometimes better than the accuracy of the corresponding operators applied to the full-resolution original images. This framework is described in detail in the following section, while in the rest of the current session we further describe database issues.

Recall that we use the database to store information at all three abstraction levels. 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.
- *Object table*, which records the object type and geographic coordinates of all the pre-extracted objects.
- *Feature table*, which records the feature vectors extracted from the images. Currently, the table stores eight different texture features including fractal dimensions.

The metadata search is performed by the query parser on the metadata table. The image search, on the other hand, is performed by the search engine on the object table, the feature table, or the raw images.

The results, which are usually a ranked list from each abstraction level, are then combined by the image search engine, and the final search results are rendered by the visualization engine.

4 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 the 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 will indeed become more efficient, and pruning of the search space can often be accomplished by 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 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 and lossy depending on whether the coding scheme preserves the original data. Transformed coding such as discrete cosine transformation (DCT) used in both JPEG and MPEG and discrete wavelet transformation (DWT) is commonly adopted as the first step for compression due to its capability of concentrating the information in the first few coefficients. A quantization scheme can then be applied to eliminate those coefficients which are close to zero.

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, the 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.

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 (DWT) coding 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.

The transform coding is based on DWT; this step yields a multiresolution representation of the original image. (For a simple introduction to Wavelets see the article [10] and the references therein.) We use short biorthogonal filters, such as the Daubechies biorthogonal symmetric ones [11] of order 1 to 5. If the original data is stored in integer format, [2] these filters allow perfect reconstruction, i.e., the transform step is perfectly invertible.

We quantize 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), followed by a fixed-model, two pass, arithmetic code. 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 4. 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 complex-

[2] 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.

ity: 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 general we do not need to analyze the whole image for content-based search purposes.

The space-frequency representation induced by the wavelet transform in itself 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 at the user request.

This algorithm is amenable to efficient implementations on modern superscalar computer architectures, such as the IBM RS/6000 family.

5 Progressive Image Navigation

Image databases frequently make use of *browse* products for displaying the results of queries. For example, a 512×512 photograph can be represented by a 64×64 low resolution representation of the same image. The overhead associated with storing the browse product is small as it contains fewer than 2% of the number of pixels contained by the original image.

However, when dealing with complex images, such a small browse product does not allow one to visually evaluate some of the image features. Thus, we would like to be able to *navigate* through the image. Figure 6 shows a portion of a TM image of the Indiana/Kentucky border. The figure shows a scenario in which the users zooms in on the area of interest. We start with a 128x128 pixel representation of a 2048x2048 portion of the image. Here, we can clearly see the Ohio river but cannot identify any other relevant features. The user starts zooming in. At each step, the resolution is doubled, while the size of the browse product remains unchanged. As the resolution increases, it becomes easier to

identify features such as mountains, cities and roads.

Currently, the system is capable of allowing the user to navigate through images by using a wavelet representation. Figure 6 was, in fact, extracted from our database. The synthesis of the image product is done in software. By representing the image as a single object rather than partitioning the image into blocks, as is currently done in many GIS systems [12], we eliminate much of the management overhead associated with mosaicing the individual blocks or the need to deal with the boundary effects that occur when applying image operators across the subblocks. It is worth noting that the dominant cost for image retrieval in our implementation is the cost of sending the image product back to the client across our local area network.

6 Progressive Content-based Search Operation

While the ability to visually navigate through the images is an important part of our current system, our primary objective is to provide a framework for automatic search of the data 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.

Much as we can navigate through the image when we are doing a visual inspection, we can analyze the image progressively. The resulting framework has proved very effective in reducing the high computational complexity of the feature extraction task, and sometimes in increasing the accuracy of the operators.

We briefly overview how this progressive technique is applied to three different operations, template matching, that extracts content at the pixel level, texture extraction, that retrieve information at the feature level, and classification, that attaches semantics to portions of the images.

Template Matching

One of the fundamental methods for detecting objects within an image is *template matching* whereby a template of size $n \times n$ is compared pixel by pixel with each $n \times n$ subimage. The objective is to find those regions which minimize the difference. Template matching is rarely exact as a result of image

noise, quantization effects and differences in the images themselves. Seasonal changes alone introduce effects that make the matching process difficult. Our experiments have shown that the correlation coefficient a particularly resilient measure of the "distance" between the template and a subimage. 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, of size $dx \times dy$. 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 θ between these two vectors. In this scenario, an exact match yields the maximal correlation coefficient of 1.

Progressive template matching allows one to approximate the angle θ 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 of $nm^2 \log nm$, 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 [13]. Heuristics that make it effective are

- natural signals often have a power spectrum that decreases as 1/frequency,
- large, distinctive features are well represented in lower resolution approximations of the images.

The difficulties that get introduced are twofold:

- As the number of coefficients is reduced, dimensionality of the search space is reduced, resulting in a many-to-one mapping. Thus, the number of false hits increases.
- Alignment errors are introduced at lower resolution, since the wavelet transform is not translation-invariant. As a result, the maximum of correlation coefficient does not guarantee an exact match any more.

We have developed techniques that allow us to quantify the latter effect without having to convert back to the spatial domain [14]. The resulting progressive approach allows us to 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 also 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

Texture is frequently used to describe two dimensional variations of an image with a characteristic repetitiveness and is, therefore, a good candidate for classification and feature recognition in a subimage devoid of sharp edges. A texture-based image retrieval process usually involves the following steps:

1. Given a target template, which usually shows the texture of the desired image, extract the corresponding feature vector.
2. Divide the images into homogeneous regions using some form of spatial decomposition.
3. For each region, compute texture features such as coarseness, brightness, entropy, and energy. The features computed from each region are expressed as a feature vector.
4. Compute the distance between the feature vector of the target image and the feature vectors of all the regions of an image.
5. Retrieve the regions having feature vectors that are most similar to the target image.

Since texture extraction has to be applied to all the regions of all the images in a database, predefined texture features are usually preextracted by processing off-line the images stored in the database. The efficiency of retrieving similar texture feature vectors can be significantly improved by building spatial index tree (such as R-tree) for the extracted feature vectors.

However, texture extraction process can still computationally expensive when the content-based query involves user-defined texture features. One way of speeding up the texture extraction process under this circumstance is to apply the texture extraction and comparison algorithm progressively for a given target image [15].

In the progressive framework, the texture features are extracted first from a lower resolution version of the image, and compared to the texture vector obtained from the target. When the accumulated Euclidean distance between the target feature vector and the image region vector is above a certain (resolution-dependent) threshold, a mismatch is declared and the region of the image is no longer considered a candidate. On the other hand, if the accumulated distance

is smaller than the threshold, texture features are extracted at the immediately higher resolution level and the process is is continued until the full resolution is reached. Using this approach, the speed of performing combined texture extraction and feature comparison is expected to improve by a factor of 2 to 8 depending on the nature of the image.

Classification

Classification of a multispectral image is the process of assigning labels to the individual pixels or to distinct regions.

In a wavelet multiresolution framework, we can consider a coefficient at level ℓ "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 7: 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 [16], 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.

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 [17], and CART (Classification and Re-

gression Trees) [18].

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 [19].

Further speedups can be achieved by ending the process at a higher level than the full resolution: this is a viable option when the user is willing to trade accuracy for response time.

7 Lessons Learned

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. Our project is focused on developing and demonstrating technologies that further our ability to support this capability. In building a prototype, we have reached several conclusions on how to build such a system:

- *Build indices whenever possible.* The design of efficient schemes to manage descriptors of the information in the library is highly desirable. It not only helps the user of the library in the formulation of queries but significantly reduces the cost of the search. Currently, the indices are built for the metadata (based on the image ID). Spatial indices are also built for the geographic coordinates of each image as well as the texture features. Unfortunately, one cannot build every index. Even if one could afford the space associated with storing all relevant metadata it is unlikely that one can foresee every possible use of the data.
- *Data organization can help reduce the cost of searching for data.* Given that many searches will require additional filtering of the returned "hits", the next technique is to organize the data in such a way that the cost of the remaining part of the search can be further reduced. This is one of the primary contributions of our project. We are studying various compression and data representation schemes. Not only does compression allow us more efficient storage of the data, but it also allows us, in many cases, more efficient filtering of the data. Of course, one can note that the organization of the dataset also implies a certain level of knowledge as to how the data set will be used. In particular, we have not fully avoided the

bane of context. On the other hand we have made significant progress in covering a set of conditions that had not been identified *a priori*.

- *The search for data needs to be interactive.* Involving the user in the search process is important. It is the user that defines the context, thus it is the user that must be able to specify the semantics and fine tune the query. There is a need to provide a query language that allows the user to capture these semantics without being constrained a priori to a specific domain. Visual support is crucial as we need to be able to identify "features" of interest so as to specify the query. Visual support plays a crucial role in providing a mechanism for viewing and interacting with the intermediate results.

This testbed provides us with a platform for understanding and evaluating the interactions between different components of the search process. It also provides us with a mechanism for interacting with users in order to better understand those user interface issues associated with image search.

Thus far, the focus of our work has been almost entirely on the use of satellite imagery. The multitude of sources of satellite imagery makes this data set extremely attractive in terms of understanding the issues associated with different kinds of images as well as understanding the issues associated with allowing for interoperability between the different sources.

Acknowledgements: We would like to acknowledge 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] M. Stonebraker. Sequoia 2000: A reflection on the first three years. *IEEE Computer Science Engineering*, 1(4):63–72, Winter 1994.

[3] A. Kitamoto, C. Zhou, and M. Takagi. Similarity retrieval of noaa satellite images by graph matching. *Proc. SPIE*, 1908, Storage and Retrieval for Image and Video Databases, Feb. 1993.

[4] J. K. Wu, F. Gao, and P. Yang. Model based 3d object recognition. *Proc. ICARCV*, 1, PCV 6.4/1-6, 1992.

[5] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying im-

ages by content using color texture, and shape. *Proc. SPIE*, 1908, Storage Retrieval for Image and Video Databases:173–187, 1993.

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

[7] B. Holt and L. Hartwick. Visual image retrieval for applications in art and art history. *Proc. SPIE*, pages 70–81, Feb. 1994.

[8] R. Samadani, C. Han, and L. K. Katragadda. Content-based event selection from satellite images of the aurora. *Proc. SPIE*, 1908, Storage and Retrieval for Image and Video Databases:50–59, February 1993.

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

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

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

[12] D. DeWitt, N. Kabra, J. Luo, J. Naughton, J. Patel, and J.-B. Yu. Client-server paradise. *Proceedings of the Very Large Data Base Conference, Santiago, Chile*, Aug. 1994.

[13] 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.

[14] 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.

[15] C.-S. Li and M.-S. Chen. Progressive texture matching for earch observing satellite image databases. *Proc. of SPIE Photonics East*, November 1996.

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

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

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

[19] 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.

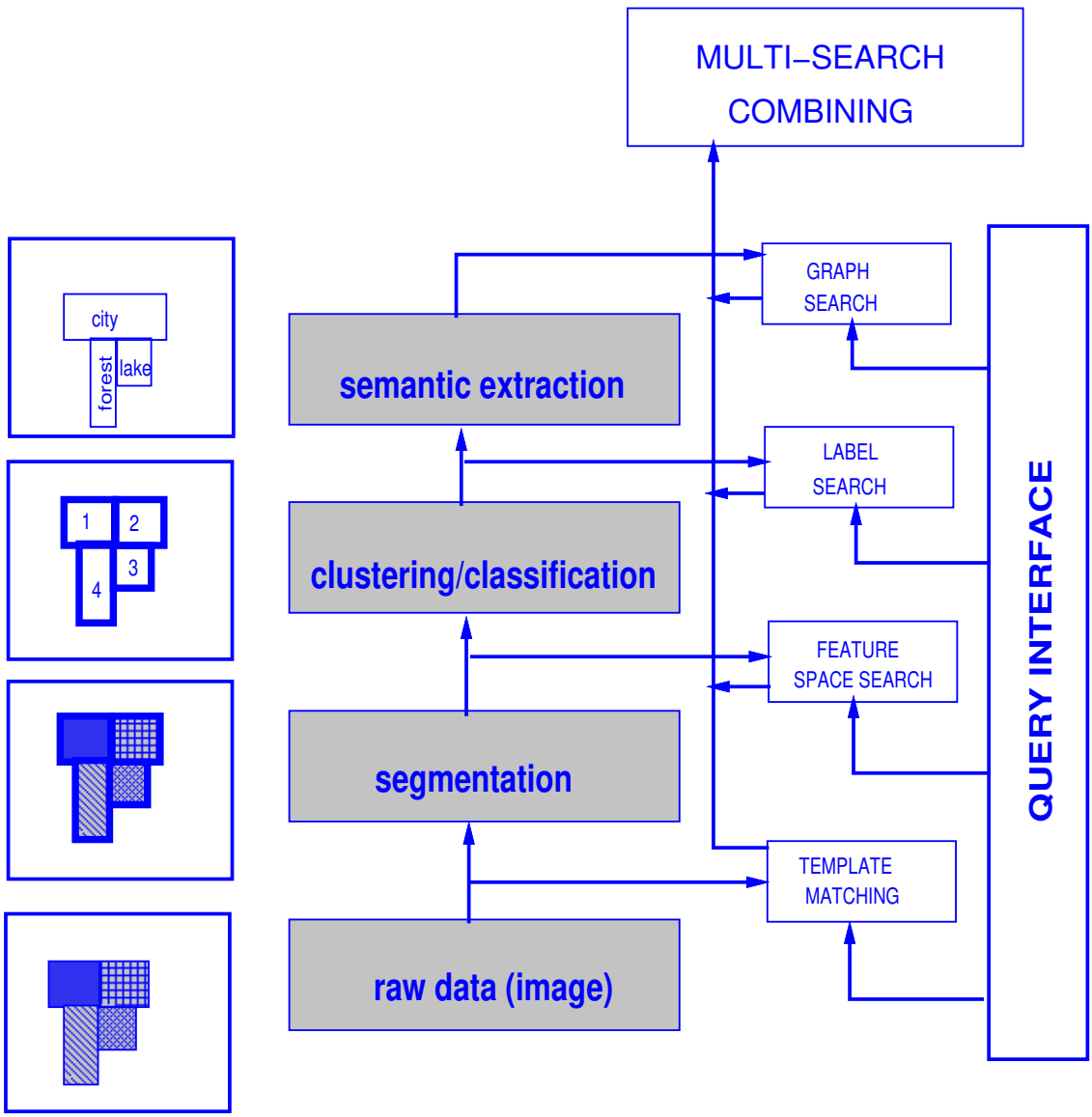

Figure 1: Abstraction levels of an image.

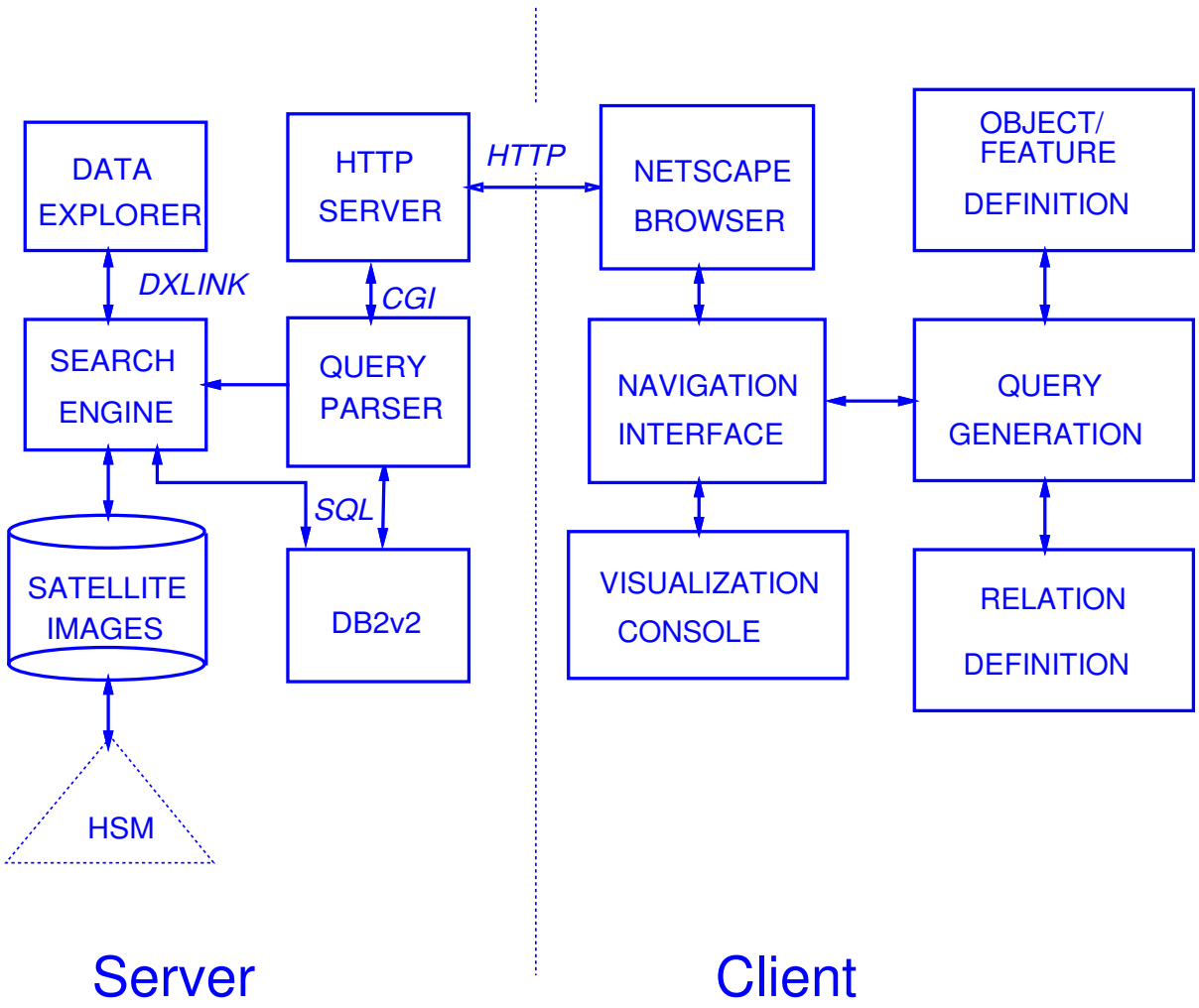

Figure 2: Architecture of the content-based retrieval system.

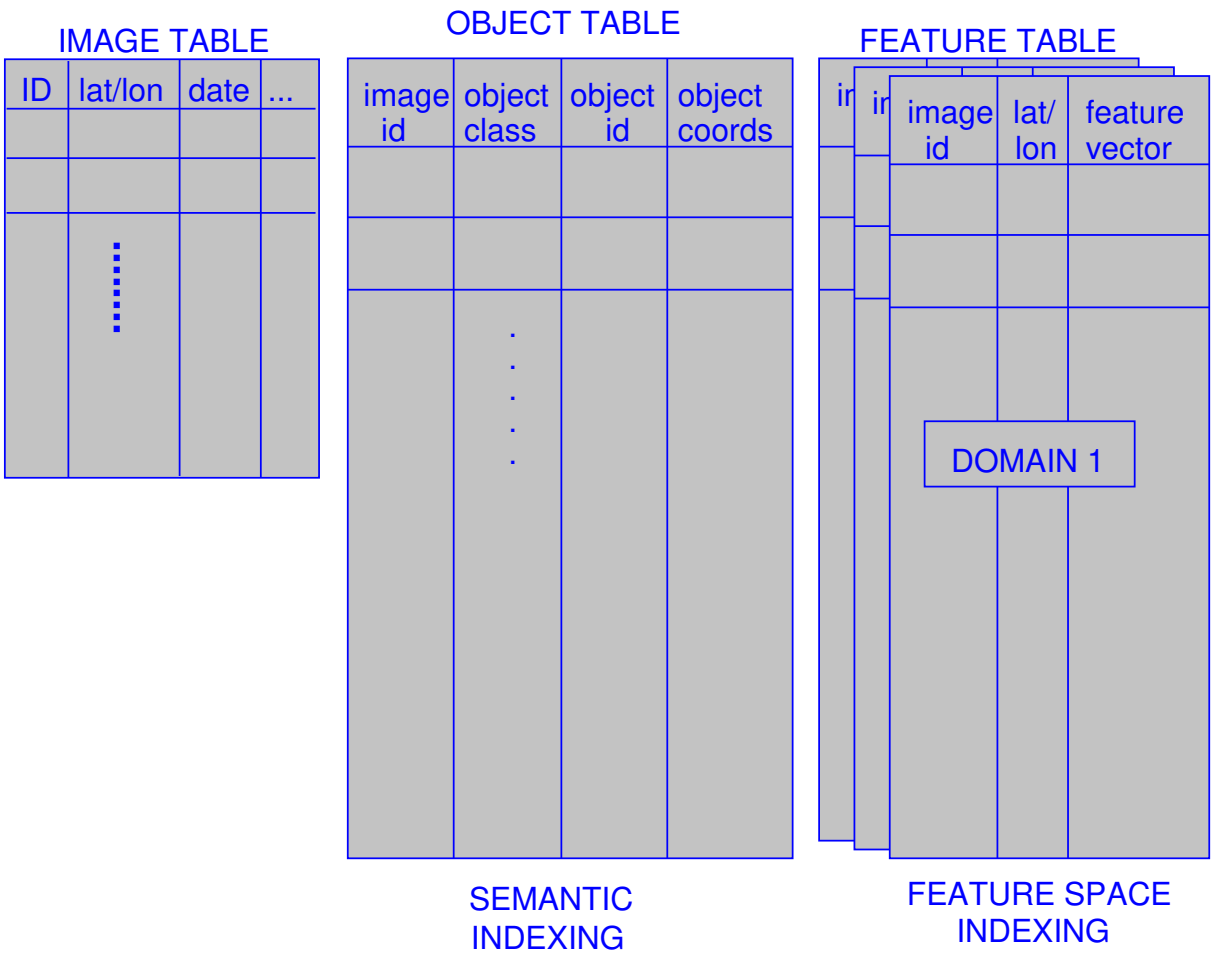

Figure 3: Schema of the database.

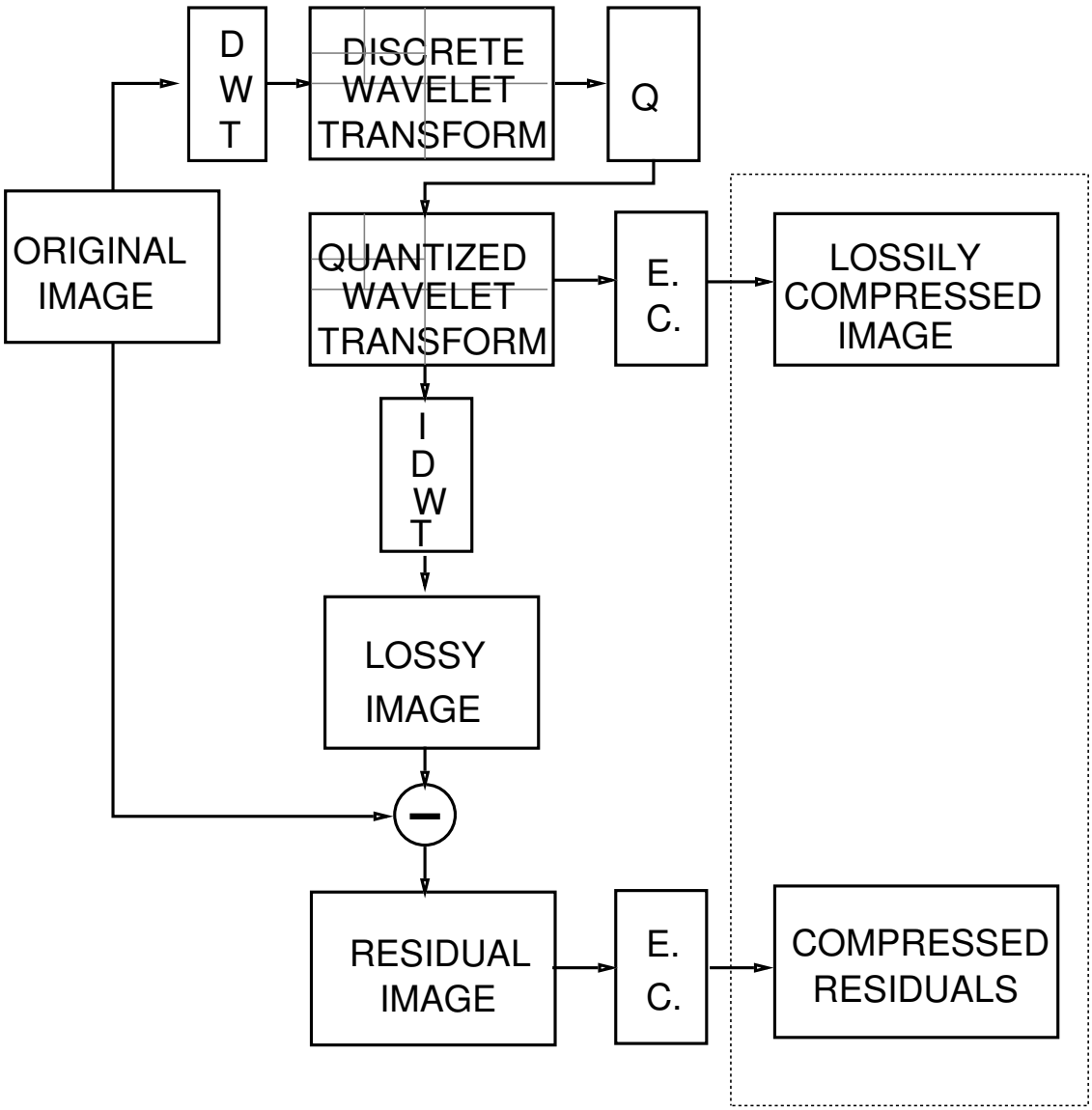

Figure 4: Overall structure of the compression algorithm. The image is transformed, quantized (**Q**), entropy coded (**E.C.**) and stored. The difference of the original and the lossy data (residuals,) is entropy coded and stored. The combination of compressed lossy data and residuals yields lossless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

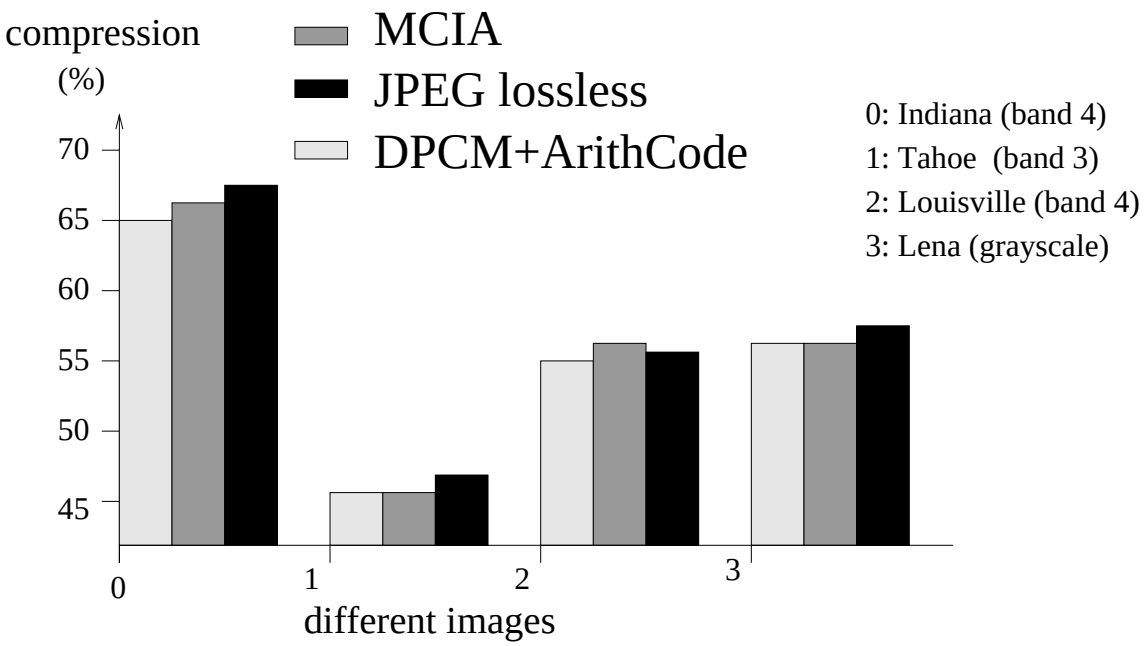

Figure 5: Comparison of MCIA with the lossless mode of the JPEG standard, and DPCM with arithmetic coding. The Indiana and Louisville images were obtained with the LANDSAT Thematic Mapper, the Tahoe image is a portion of a NALC dataset.

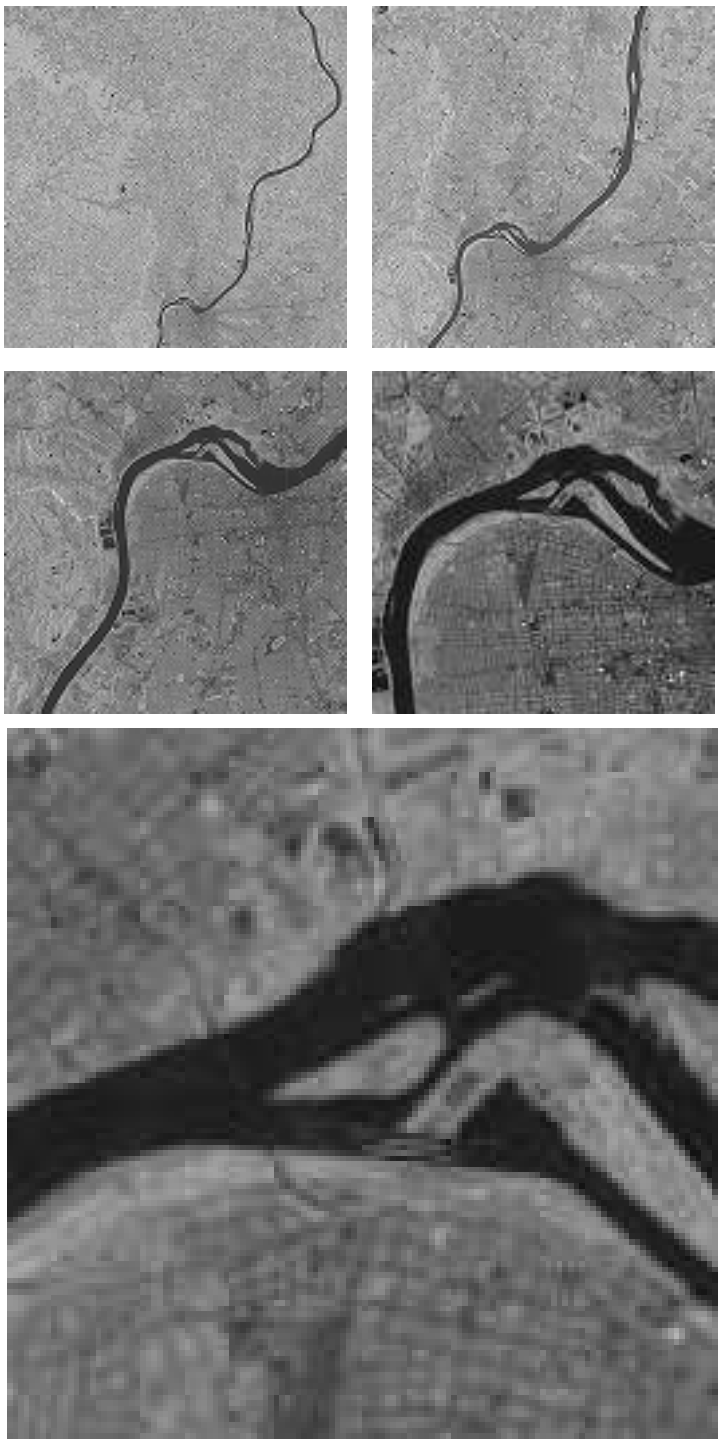

Figure 6: Homing in on Louisville

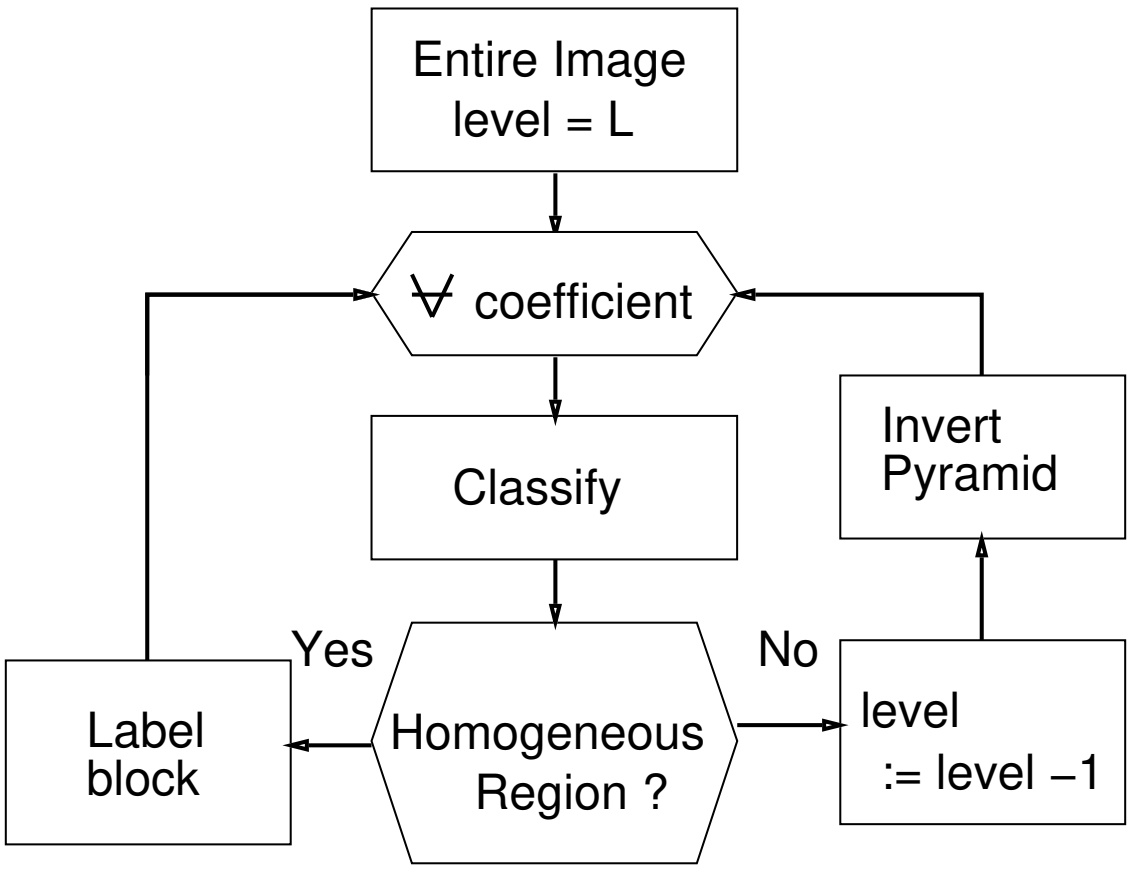

Figure 7: Progressive classification.