Automatic Feature Extraction and Indexing for Content-Based Visual Query

Shih-Fu Chang, John Smith, and Hualu Wang

Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University
New York, NY 10027

Tel: 212-854-6894
Fax: 212-316-9068
Email: sfchang, jrsmith, hwang@ctr.columbia.edu

keywords: content-based image query, image indexing, feature extraction, image database, video server.

Abstract:

Content-based indexing and query has been considered as a powerful technique for accessing large visual information systems (databases and video servers). By extracting and indexing the visual contents of the images such as texture, color, and shape, users may search desired images by specifying the image contents directly. However, a practical and economical solution cannot afford extensive user involvement. This paper proposes an approach in which image processing technology is explored to the limit of its capability for *automatic* extraction and indexing of image features. We relax the criterion of image content recognition that concrete 3D objects can be successfully segmented from 2D images. Instead, we aim to index the prominent image regions with distinctive features only. By combining multiple modalities of useful signal features, we hope to characterize the prominent image objects efficiently and effectively. In order to further explore the maximum synergy between feature extraction and other image processing tasks required in image databases, we also propose to extract the visual features directly from the compressed images. For existing large image archives, decoding of compressed image data is hence not necessary. For new image database design, merging feature extraction intelligence into the compression algorithm has recently been recognized as an important issue for image processing research. We use texture, shape, and video scene change detections as examples in describing this compressed-domain approach. All the proposed research are being incorporated into practical applications in Columbia University's Multimedia testbed.

1.0 Introduction

Content-based indexing and query has been considered as a powerful technique for accessing large visual information systems (VIS, such as image databases and video servers) [1]. It complements well the existing, more or less mature text-based indexing technique. One crucial component of content-based query is recognition and extraction of "content" from the visual data. In describing the content of images and video, one may find many different levels of possible interpretation and description, as shown in figure 1. Entities in different levels of the hierarchy are associated with different levels of attributes, ranging from low-level signal features to high-level context-specific semantics. Starting from the bottom level, we may associate attributes such as illumination intensity and color to individual image pixels. Image regions comprise of a group of pixels with distinctive attributes such as texture, color, and shape. In the object level, semantics begin to appear. An image object usually corresponds to a physical object which actually exists, such as a person, a car, or a building. Such objects usually are composed of several regions (for example house is composed of door, window, and roof). Some exceptions do exist (such as background objects, sky, sea, grass field) and cannot be easily characterized by separate disjoint physical components. On top of objects, an image is composed of several image objects, and video is composed of a sequence of images, usually associated with some camera control operations (such as scene cut, zoom, and pan).

In another dimension of content interpretation, relationships between entities or occurrences of entity mentioned above play an important role. Actions may be associated with each individual object (e.g., a person is running), and relationships may exist between objects (e.g., person A is talking to person B). Spatial and temporal relationships are also typical between entities or occurrences of entity in every level. For example, object A on the right of object B, picture-in-picture, a large white graphic area in the bottom are all typical spatial relationships. Compositions of scenes in video (scene cut, dissolve) are typical temporal relationships.

All these aspects of image contents are fundamental to human's perceptions towards information of visual form. Human can recognize the attributes and relationships of different visual entities naturally and easily. However, for computers, many of these tasks still remain very difficult, if not impossible at all. Our effort in this paper aims to answer the following question, how far can we push for computers to recognize the visual contents *automatically* without human interventions. Study of image feature extraction for pattern recognition has been studied for many decades, but the application to VIS for content-based query is still in its infancy. We envision that all semantic-level contents need to be provided by users. We also envision that with 2D image representation of 3D real-world objects, computers will not be able to recognize objects automatically in general unrestricted settings. However, we believe that pushing upward in the content hierarchy of figure 1 by applying innovative image processing techniques is still worthwhile and will provide promising avenues in accessing and indexing images and video.

Specifically, our goal is to explore the maximum usefulness of applying image processing techniques in automatic visual feature extraction. Although today's technologies still cannot teach computers to recognize image objects from 2D images, we think it will suffice to some extent if "*prominent regions*" with "distinctive features" can be extracted and indexed. For example, we will extract regions with specific texture and color from green grass field images, even without recog-

nizing its semantics. The definition of "prominence" can be based on some attributes of the region, such as size, feature uniqueness. The immediate advantage is that prohibitive cost of manual indexing is now avoided. In addition, we can minimize the dependence of feature indexes on specific applications and make them more universally applicable. In some sense, we only aim at low-level signal features opposed to high-level semantics. On the other hand, we do not envision this automatic feature indexing method to supplant higher-level user-assisted alternatives. Instead, to achieve the highest system performance, we think this low-level feature recognition framework should be integrated with high-level alternatives, such as knowledge-based assistance and user provided semantics, which most people consider indispensable.

Another goal of our research is to explore the maximum synergy among different aspects of image database design — feature extraction/indexing, compression, manipulation, and storage. Traditional study of image compression only concerns with performance of rate reduction, signal quality distortion, and implementation complexity. Traditional image manipulation techniques also focus on efficiency and image quality only. Little attention has been paid to the integration of these tasks in the context of image databases. To overcome this shortcoming, we propose an important principle here — *extract visual features in the compressed domain whenever possible*. Imagine that huge collections of images and video need to be compressed when being stored in the databases. It will greatly reduce the implementation cost if the image features are extracted directly from the compressed format. Not only that decoding of existing compressed data can be avoided, moving feature extraction to the compressed domain also allow us to take full advantages of the compressed domain, in which data size is much reduced and visual features usually have been somewhat revealed through signal decomposition and information filtering. We will describe techniques for defining texture and shape features for still images, and scene cut features for video in the compressed domain in later sections.

Content-based image query has been first demonstrated in an comprehensive way in [2], where image features are extracted with the assistance of user intervention. Although using a similar set of features (texture, shape, and color) for still images, we explore two new distinctive research fronts — (1) automatic tools for feature extraction and region recognition, and (2) compressed-domain feature extraction whenever possible.

Content-based visual query will not be deemed practical without actual evaluation in practical concrete applications. Techniques described in this paper are being prototyped in actual image databases and video servers in Columbia University's multimedia projects, such as medical, art image databases, and the video on demand testbed [3].

2.0 Image Features

Texture, shape, and color are considered important underlying primitives in human visual perceptions of the real world. We discuss techniques for extracting prominent regions with distinctive texture, shape, and color in this section. We will demonstrate the feasibility of extracting visual features such as texture and shape from the compressed image formats, and the feasibility of extracting prominent features without user assistance.

2.1 Texture

Texture is an important element to human vision. Texture features have been used to identify contents of aerial imagery such as bodies of water, crop fields, and mountains. Textures may be used to describe content of many real-world images; for example, clouds, trees, bricks, hair, fabric all have textural characteristics. Particularly when combined with color and shape information, the details of image objects important for vision are provided.

Texture Discrimination:

Two major approaches have been used in the study of texture discrimination — statistical and structural. Psychophysical studies have also shown that signal decomposition by Gabor filter banks can approximate the mechanisms of human vision in texture discrimination [8]. In order to explore the advantage of deriving texture features from the compressed image format, we use several typical image compression techniques, such as wavelet transform, Discrete Cosine Transform (DCT), and subband transform, to approximate the feature extracted from the Gabor filter banks. All these image compression techniques decompose the image into different frequency bands, from which energy distribution and statistics parameters can be used to define the texture feature. For example, texture feature sets can be extracted from the wavelet images by measuring the first-order and the second-order statistics of each wavelet subband. For a 3-level wavelet decomposition, feature vectors with 20 terms will be produced.

Additional techniques, such as Fisher Discriminant Analysis, may be further used to maximize the separation capability among different textures [10]. In order to get texture classes as complete as possible in the process of determining the optimal texture discriminant function, we obtain the complete set of 112 Brodatz texture images [9]. We hope that using the complete set of Brodatz textures will construct a discriminant function general enough to discriminate between new and unknown textures. The Mahalanobis distance in the transformed feature space was used to measure the similarity between textures. In response to a "Query-by-texture," all textures in the database will be sorted by distance from the texture-key. This doesn't require that a threshold to be established. However, to decide whether two textures are similar or not, as in the quad-tree segmentation described in the next section, does require a threshold in distance.

As mentioned above, the above texture extraction technique can be applied to various image compression algorithms, such as uniform subband and the popular DCT. Using more than 2000 texture cuts from the Brodatz collection, we found that all these signal compression algorithms produce satisfactory classification accuracy rate (about 90%), while wavelet transform does slightly better. The wavelet compression algorithm also provides an ideal platform for supporting multi-resolution image representation in the image database. A subset of the encoded data can be used to reconstruct the image with a smaller size, lower quality, or lower resolution. We will also describe a technique using the wavelet compression technique in extracting the shape feature in the next section. Figure 2 shows a texture classification scenario from our texture image database including more than 2000 texture cuts from the Brodatz collection.

Prominent Texture Region Extraction:

A natural image usually consists of many different regions with distinctive textures. To extract these texture regions, we use the texture discriminant function defined above to match blocks within each image to perform the *texture segmentation*. Because the goal of this segmentation is to provide indexing of images, we relax the constraint that segmentation provides perfect boundary extraction. Texture segmentation is considered successful in this context when texture regions of the images are represented accurately enough to provide the expected matches upon a "Query-by-texture". More accurate boundary information can be obtained in a boundary-sensitive feature such as shape.

The spatial quad-tree provides an efficient representation for region segmentation. Each quad-tree node points to a block of image data. Child nodes are merged when the discriminant function indicates that the child blocks contain sufficiently similar textures. For each quad-tree node, signal decomposition for texture derivation can be performed independently, or obtained directly from the decomposition of the full image. If the decomposition of the full image is already available (as in the case where images have been compressed), decomposition of each quad-tree image block can be readily obtained.

Typically in a top-down approach, a full quad-tree data structure is formed by splitting a single parent node into four children, then recursively splitting all descendants into four children, until some minimum size is reached. The quad-tree decomposition can be used to perform image segmentation by assigning a condition by which nodes are split. Conversely, in a bottom-up approach using a fully-grown tree, conditions may be assigned by which children are merged. In general, the two approaches will not give identical results in application towards image segmentation. Post-processing routines for adjoining similar spatially adjacent nodes with different parents may be added. Also, modification of the quad-tree structure to allow each parent node to have two, three or four children may be beneficial. When all four children cannot be merged together, subsets of the children may be paired horizontally or vertically depending on which arrangements group the most similar children.

Choosing the distance threshold for determining whether two textures are sufficiently similar is not trivial. A fixed distance threshold will not be optimal for all types of textures. Based on the examination of the training texture cuts taken from the Brodatz collection, we have found that the distance threshold depends heavily on *the block size* from which the features are extracted and on *the energy of the feature set*. There is an inverse relation between block size and the accuracy in statistical measures produced from the blocks. Since smaller texture blocks will contain fewer data points from which to derive statistical features, there will be a larger deviation in features extracted from these blocks of similar textures. This results in a greater variance in distance between possibly similar textures necessitating a higher distance threshold for comparing textures of smaller block size. Likewise, features extracted from larger blocks produce smaller within-class variation, necessitating a lower distance threshold for comparing textures of larger block size.

Texture-Based Query:

A "Query-by-texture" can be formulated to search through the image database, returning images found to contain regions of similar texture. Here there is no restriction that the textures belong to predefined classes. The texture key may be produced by the user's cutting an arbitrarily

shaped region from an image or selecting from samples of provided textures; it may be synthesized through texture descriptions; or it may be drawn using graphics tools.

One implementation trick is worth mentioning for improving the query efficiency. The texture key provided by the user to initiate a texture-based search can be examined for information on *the scale of the texture process*. The texture process scale is determined by the size of the fundamental elements which are repeated in some way to form the texture. By performing a quad-tree based texture segmentation on the key, ideally a single quad-tree node will be produced, indicating one texture present. Starting from some initial small block size, and performing quad-tree bottom-up decomposition on the texture key at successively larger starting block sizes, eventually a complete merge will result. If this is not the case, then it is most likely that the texture key does not contain a single texture, or the scale of the texture process is larger than the texture key size. If a single texture is found, then the smallest block size that produces a full merge of the texture key indicates the minimum block size in the database to be examined. This scale information obtained by processing the texture key can reduce the number of searches for a "Query-by-texture". Blocks in the database that are smaller than the minimum block size obtained will not have to be considered. This "minimal block size" concept is illustrated in Figure 3.

We have conducted some experiments on the composite image database to test the above proposed approach to texture-based image matching. A typical result from a "Query-by-texture" using a cut from Brodatz texture D101 is indicated in Figure 3. Currently, we are in the process of applying this texture region extraction technique to an art image database.

2.2 Color

Psychophysical research has been studying how human vision system discriminates different colors. There also has been research applying color features as query keys to image database applications. Typical approaches use *single color, color pairs, and color histograms* to index the color information contained in the images. Each approach has different advantages and disadvantages. Usually, given the well-defined image objects, color histograms can well describe characteristics of the unique color distributions of the physical objects. For example, the relative amounts of brown and white, combined with oval shape description, may describe what may be nearly uniquely, a "football." However, today's image processing technologies still cannot generate concrete image objects corresponding to physical objects without human assistance. As an alternative, we relax the requirement and only aim to index prominent image segments that have consistent color distributions but may not accurately correspond to actual physical objects. In addition, as an answer to the need for automatic color feature extraction and quick and efficient indexing, we have also developed a single color extraction method for image data indexing.

2.2.1 Color Space Consideration

One major issue every color study faces is the choice of color space. In the following descriptions about single color region extraction and color histogram definition, appropriate representation of colors (i.e. color space), sampling/quantization in the color space, and similarity measurement are required. There are many choices for color space, each providing some benefit in terms of image representation and color differentiability. While image data is usually presented in RGB space, such as to allow display on color monitors, a quantized RGB space does not provide

a visually uniform sampling of colors. As a result, the distance between any two colors is not based on Euclidean distance. Other color spaces give better color space compaction, such as YCrCb and YIQ, whereas RGB provides none. As such, there is greater opportunity of attaining a desirable sampling of the color space, however the distance function in these spaces is not simply Euclidean. On the other hand, color spaces, such as CIE-Lu*v*, CIE-La*b* and Munsell have been derived to approach a uniform sampling of the visible color gamut, and use a Euclidean distance function to determine the visible difference between colors. While these color spaces have these desirable properties, the forward transform from RGB color space is highly non-linear. The reverse transforms are not straightforward or do not exist in closed form.

2.2.2 Single Color Query and Indexing

The single color indexing technique described here is most similar to the file inversion approach of data indexing. This works well for image colors, because the full gamut of visible colors can be represented without much visible distortion using a fixed finite set of colors. For great accuracy, thousands of colors may be needed, but usually hundreds of colors will suffice. Since there will be a finite total number of colors, the file inversion approaches requires that a query find through one index the right color and then the matches within the database may be read off straightforwardly. This technique can be easily extended to extract occurrences of pairs of colors.

When the user can describe a specific color region to be contained in an image, the images can be quickly retrieved using the single color index. Here the identification of a single color region should be distinguished from the general presence of a color in an image. For example, a color histogram may identify that a color occurs with certain frequency within an image, but does not determine whether the color was dispersed throughout the image or indicates that a region exists that contains a single color. The single color index will point to regions within the image that contain single colors. The regions however, will not correspond directly to objects. But, the flexibility exists to allow the color regions to represent isolated objects, in terms of further allowing queries to be formulated that specify spatial relationships between color regions. For example, queries can be formulated such as, “Select images containing a blue region above and touching a red region,” or “Select images containing a black region inside a hazel brown region inside a white region,” such as to describe hazel brown eyes. To execute this query, the images containing these three color regions will be selected, then the spatial characteristics will be examined to identify matches. This approach provides a different type of query expression than using color histograms of image segments. In the complete system, the techniques complement each other in providing a full set of tolls for color-based queries.

Color Region Extraction:

The color regions are extracted by sampling the images using the quantized gamut colors. Once the collection of colors is produced, such as by uniform sampling of CIE-La*b* color space, any distance function may be used. The extraction process proceeds as follows, also see Figure 4: first, a color histogram is generated for each image using the bins that correspond to the quantized color gamut. Then by iterating over the non-zero bins, the color distance is compute between the image and each non-zero bin color. Again the distance function may be chosen to match the human visual system. This produces a series of distance images, see Figure 5(b) for distance from the color pink. Simple thresholding of each distance image gives a binary mask that

when fitted over the original image identifies the occurrences of the color within the image. However, immediate application of the mask does not always give good results. This is because noise, lighting and other artifacts give spots within color regions and outside of color regions. To compensate for the spatial aberrations, we use a rank-order filter on the distance image. This has the quality of eliminating isolated points of color, and closes over regions that have holes or are disconnected. Linear filters, such as low-pass filters are not appropriate for this task because while artifacts might disappear, important information is also lost, such as boundary information being blurred. The thresholded, rank-order filtered distance image appears in Figure 5(c).

Once the filtered distance image is obtained, it is thresholded. The threshold may be constant for all colors, if the distance function and color space give uniform spacing between visibly similar colors. Otherwise, the threshold will be a function of the color. After thresholding, a mask image is obtained. Next follows a sequential labelling routine, whereby unconnected regions are distinguished and labelled. This also provides for computation of region attributes, such as area. Finally, a threshold is applied at this stage, based on region area. This threshold is not a function of color, but more likely a constant for the database. This threshold corresponds for the minimum size of the areas the color indexes should point to. An area threshold, of approximately 256 seems to provide desirable results. The size of the color region index for the database is inversely proportional to this threshold. A labelled image corresponding to all pink regions appears in Figure 5(d).

Color Region Indexing and Color Agents:

After the color regions have been identified using the above process, the database is populated with the color region data. The color, an image pointer, position and size of each color region are kept as attributes of each color region. The database query is initiated with the user selecting a color from a color picker, see Figure 6, or from image. A color index is then searched for the match to the selected color. The user may also specify tolerances for the color, see Figure 6(b), such as for the channels of a particular color spaces. We call the definition of the color and the tolerances, *the color agent*. For example in YIQ space, the user may specify a color agent that has a larger tolerance in the luminance (Y) channel, if the brightness of the desired color is not a significant factor. This may allow, say all shades of the “blue” to be retrieved. The values of the tolerances will determine how many and which colors will be selected in the index. Once the colors in the index have been identified, the file inversion now allows the images containing matches to be read off directly. Therefore, the only delay in query execution will be the initial determination of the colors in the index that match the user specified color agent.

2.2.3 Color Histogram Indexing and Query

Color histograms can provide more complete color descriptions than other alternatives, such as single color, average color, and color pairs. Actually, average color and color pairs can both be derived from the color histogram information. However, color histograms also pose new implementation issues. First of all, it’s more difficult for users to directly specify color histograms quantitatively. While users could describe a color histogram by pointing out some dominant colors and their percentages in the image, or by using a graphical interface to adjust the shape of the color histogram, it is still more feasible for users to simply use sample images or image segments.

Furthermore, one big concern of color histogram design is the huge dimensionality. A 24-bit RGB space without quantization will produce histograms with 2^{24} dimensions. Quantization of color space is hence needed to reduce the dimensionality of color histograms. As a compromise between complexity and accuracy, we quantize the La*b* space uniformly into 64 bins. If we use RGB space instead, we will have larger quantization errors because of the color nonuniformity. Lastly, standard transformation from RGB to La*b* requires RGB values to be transformed into XYZ components, followed by the XYZ to La*b* transform. Since XYZ to La*b* transform is non-linear, we have to carefully determine the dynamic ranges of the three new components to get the optimal quantization and normalization effects. We also found that another uniformity-preserving color space Lu*v* will produce color histogram distributions similar to that for the La*b* space.

Determining the optimal distance functions for measuring the similarity between color histograms is also critical. Both color histogram intersection and color pairs approach have been proposed as good measures. An effective approach is proposed in [7], which uses a full quadratic form of the color histogram difference and adds a modulating matrix. It guarantees positive distances and, most importantly, takes into account the crosstalk between different quantized colors. The distance between two histograms $\grave{x}$ and $\grave{y}$ can be defined as follows

$$d(\grave{x},\grave{y}) = (\grave{x}-\grave{y})^{t} A (\grave{x}-\grave{y}) = \sum_{i,j} (x_i - y_i)\, a_{i,j}\, (x_j - y_j) \qquad \text{(EQ 1)}$$

where elements $a_{i,j}$ of matrix A represents the “cross correlation” between color i and color j. If A is the identity matrix, then (EQ 1) becomes the standard Euclidean distance between. The compensation by $a_{i,j}$ coefficients is important because usually colors are not orthogonal. For example, the distance between red and orange should be smaller than that between red and blue. Note that using uniformity-preserving color space also provides advantages in simplifying the determination of A matrix.

Using color histograms to discriminate colors provides much flexibility. For example, the *average color* of an image region can be obtained easily by calculating the mean of the color histogram. In addition, intersections of color histograms can provide useful measurement of color similarity, such as “30% of the color distribution of image A is similar to the color distribution of image B”.

Color-Histogram Based Region Extraction:

As in texture region extraction, we want to extract image regions with consistent color histograms and use them as image indexes. The quad-tree based segmentation technique proposed above and the distance function defined above can be used to perform color “segmentation”. Neighboring image blocks pointed by corresponding quad-tree nodes are merged whenever their color histograms are similar enough.

One nice feature of using quad-tree segmentation for color indexing is that we can index each intermediate node and terminal node of a quad-tree with its representative color histogram. Each terminal node of a quad-tree points to an image region with homogeneous color distribution. In other words, all the image blocks below this terminal node have color histograms within some distance threshold and are thus merged. For the intermediate nodes, although the underlying

child nodes do not have homogeneous color patterns, the color histograms are still useful in some applications, such as queries of “30% red, 70% green”.

There are different ways to segment images into square blocks. We can choose different block size or use adaptive block size, or we may choose overlapping blocks of the image instead of non-overlapping segments. At this stage, we segment the whole image into non-overlapping blocks of 64 pixels x 64 pixels and generate color histograms for each block. The block size should be large enough to maintain the accuracy and reduce the sensitivity to noises. The block size cannot be too large either, for we will then lose the accuracy of image region boundaries.

Query Examples and Results:

Now the color histogram query proceeds as follows. Given a sample image segment provided by users, we transform it into La*b* color space, compute its color histogram, and threshold small value bins to reduce noises. The distance function in EQ 1 is used to calculate the color histogram distance between the input image key and all possible candidate image regions, which are extracted and populated during the indexing stage. As matches to the similarity query, images containing most similar image regions are returned. Note that there may be more than one single match regions occurring in the same image. Figure 7 illustrates the procedure. Figure 8 shows two query examples. The image region with red border highlight is the one with the highest color histogram similarity. Other match image regions are highlighted with green borders, with brightness proportional to the degree of similarity.

There are still many open issues in the color histogram approach toward image content-based query. One shortcoming of the color histogram approach is that it does not contain any spatial information of color distribution in an image. Also, unlike the single color for which we can use file inversion to avoid exhaustive search, color histograms do not work well with file inversion. The index file size is simply too large if we do not quantize bin values. Accuracy requirement, however, prohibits any substantial histogram quantization.

2.3 Shape

Shape feature is extremely useful in many image databases (such as electronics schematic matching) and pattern matching applications (such as military target recognition). Traditional approaches for recognizing shapes in the images can be classified into edge based, region based, or feature based. Edge based approaches detect edge points by finding the intensity discontinuity. Concrete shapes are then obtained by performing closing and filtering operators on the edge points. We use this approach here, however, applying the edge/shape extraction algorithms in the compressed domain.

We have mentioned in section 2.1 that the wavelet transform provides a suitable compression format for multi-resolution image representation and texture feature extraction. Actually, wavelet decomposition has been used as an effective technique for detecting edges as well. Mallat and Zhong used the derivative of Gaussian function as the wavelet function and detect edge points at the maxima or zero-crossing points in different scales [11]. The same technique has been applied to signal approximation and image coding as well [11]. Many works of edge detection in the multi-scale space have been reported also. These prior efforts have provided an excellent framework for shape extraction in the wavelet compressed domain. Using multi-scale edge

detection also provides additional advantages for image database applications. Usually, edges and shapes obtained from the higher scales are more clean and less sensitive to noises. It fits the characteristics of image database applications well, where prominent shape features are required, as opposed to highly accurate boundary information.

In determining the similarity between different shapes, usually higher-level attributes are used, such as area, center position, perimeter, orientation, or X-Y aspect ratio [12]. These high-level representations of shapes can also grant nice invariance property with respect to size, rotation, shifting. However, they don't really capture the characteristics of shapes accurately. Intermediate-level representations and similarity measurement such as Fourier Descriptors, Chain Code, Hough transform, and principle component decomposition may alleviate these shortcomings to some extent at the cost of sacrificing invariance property. We think the choice of optimal representation and similarity measurement in the context of content-based image query is worth further study. Currently, we are experimenting the wavelet-compressed-domain shape recognition technique for content-based image query application.

2.4 *Integrated Feature Map* for Content-Based Visual Query

With all the above low-level signal features available in indexing the prominent regions of images, the following concept of ***integrated feature map*** becomes very interesting and useful.

First, independent signal features can now be integrated to detect and index image objects more effectively. Consistency in segmentation results from different features increases the confidence level of the object extraction results. For example, combining the brown/white color histogram region with the oval shape, we should be able to recognize a football more easily. Indexing each image object by multiple features also provides higher flexibility in the query stage. For example, the same object can be searched by any or all of the associated features. A query example could be

"find images containing regions with this texture AND/OR this color pattern AND/OR this shape".

Secondly, boundary alignment between objects extracted by different features provides a new arena of image query. Some image regions may have well-aligned segmentation boundaries from different features; some objects may not, depending on the physical appearance of the image objects. For example, with the integrated feature map, the system will allow users to issue queries like

"find image regions with this texture, this homogeneous color pattern in the center, but a random color distribution around the boundary",

or

"find all image regions with homogeneous texture and well enclosed color patterns within the boundary (or with a color pattern spread over several texture regions)".

Region alignment can be easily implemented if a consistent data structure (such as the modified quadtree described earlier) in region extraction and representation. Color regions, texture regions, and shapes can all be represented by the quadtree structure, for which various efficient manipulation operations are already available.

Lastly, feature extraction results from different channels may complement each other in improving the individual accuracy. For example, inference from segmentation results by features like color and texture can improve the edge accuracy and find the missing edge segment of an image object.

3.0 Video Features

Video carries much richer information than still images. Research on video indexing and content-based access is still in its infancy. As shown in the content hierarchy of figure 1, besides the above features typical of still images, a video sequence can be further characterized by two additional channel of "features" — (1) how the video is made (the camera manipulation functions)? (2) how still image features change over time (e.g., object motion and temporal relationships)?

Applying the compressed-domain feature extraction principle, we have developed a set of algorithms and an interactive system for users to detect scene changes (direct and dissolve) directly from the compressed video format. Assuming that the content during each scene is more or less consistent, this "divide and conquer" approach can be used to filter and index the tremendous amount of visual information in video. Once a video sequence is segmented into different scenes, we index each individual scene with its representative features, such as texture, color, shape, and motion. Other high-level attributes such as scene length can be automatically indexed as well.

In several practical video coding standards such as MPEG-1/MPEG-2 and H.261, image sequences are converted to the motion vectors (for prediction of current image blocks based on neighboring image frames) and frequency decomposition of prediction errors. Ideally, the former variable can be used to model the motion of the image objects or regions, while the latter the dissimilarity between image frames. Some video sequence features can thus be readily derived from the compressed data directly.

Our compressed-domain video feature extraction technique takes advantages of the above characteristics of video compression technologies. The distribution of motion vectors and transform of the motion compensated residual errors in the compressed streams are used as cues in detecting the dissimilarity between image frames. Since a full decode of the compressed bitstream is not necessary, computation time can be thus greatly reduced. There have also been recent research studying detecting other camera operations (e.g., zooming, panning, and wiping) from the compressed streams as well. Figure 9 shows the graphic interface of our interactive video on demand system in which the above video indexing technique has been incorporated.

4.0 Application in Columbia's Multimedia Testbed

All the above mentioned research and proposed technologies are being incorporated into Columbia University's Multimedia testbed, which is a research and development platform for accommodating various multimedia applications such as digital libraries, Video-on-Demand, and interactive multimedia systems. The underlying infrastructure includes campus-wide ATM networks and a graphic supercomputer as servers. Projects have also been initiated to apply the proposed feature-based indexing and access techniques to a practical art image database including about 2000 art images from museums and galleries.

5.0 Conclusions and Future Work

Content-based image query will be an extremely powerful technique for accessing future huge image/video archives. However, if every single image still needs to be inspected by users for content extraction and indexing, the implementation cost is too high. Recognizing the fact that current image processing technologies still cannot perform automatic image object recognition successfully, this paper describes a practical and efficient approach aiming at automatic extraction and indexing of prominent image regions. Although high-level semantics will not be able to be captured at this level, automatic generation of multi-channel visual features such as texture, color, and shape can provide an efficient and flexible foundation to be connected with high-level alternatives. Also, by combining multiple modality of features into an integrated feature map, we can improve the effectiveness of each individual feature channel and move further upward in the visual content hierarchy.

We envision this low-level signal feature indexing technique will complement nicely high-level image indexing methods, such as user-provided descriptions, semi-automatic image segmentations and feature indexing, and domain knowledge applications.

Another technical thrust discussed is feature extraction directly in the compressed image format. There is great synergy between image compression and image feature extraction. Many existing image compression algorithms have provided potential platforms for defining useful signal features. We used texture, shape, and video scene changes as examples in the context.

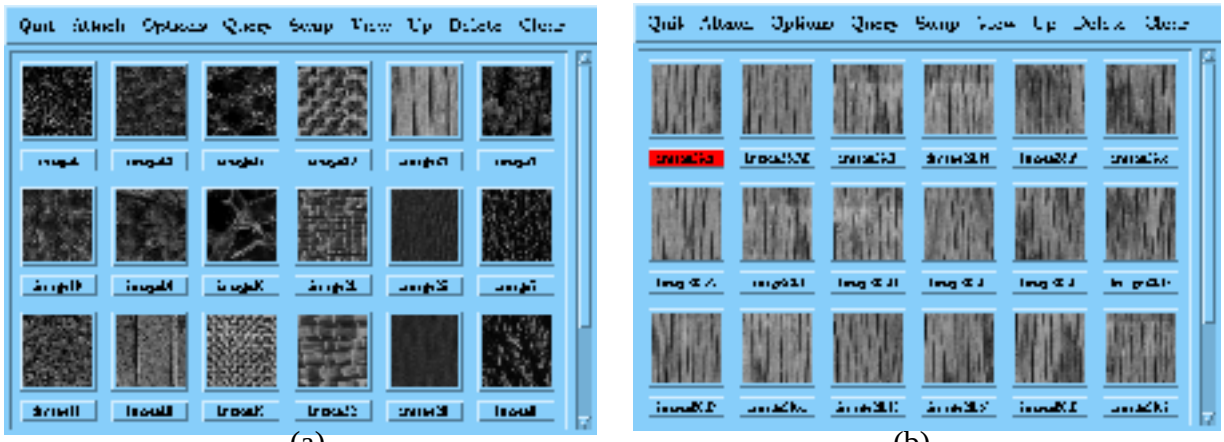
Many new research issues arise in this challenging area. Among them are efficient data structures for representing multi-dimensional signal features, efficient comparison and search algorithms in the feature space, subjective quality based distance measurement, and integration with knowledge-based applications.

6.0 References

1. Ramesh Jain, *NSF Workshop On Visual Information Management Systems*, Redwood, CA, Feb. 24-25, 1992.
2. W. Niblack, et al, “The QBIC Project: Querying Images by Content Using Color, Texture, and Shape,” IBM Research Report, RJ 9203 (81511) Feb. 1, 1993.
3. S.-F. Chang, D. Anastassiou, A. Eleftheriadis, J. Meng, S. Paek, S. Pejhan, and J.R. Smith, “Development of an Advanced Video on Demand Testbed,” IEEE Workshop on Visual Signal Processing and Communications, New Brunswick, NJ, June, 1994, also submitted to Journal of Image Communication, Special Issue on Video on Demand and Interactive TV, 1994.
4. A. Netravali and B. Haskell, *Digital Pictures*, Plenum Press, 1988.
5. M. Swain and D. Ballard, “Color Indexing,” *International Journal of Computer Vision*, 7:1, 1991, p. 11 -- 32
6. A. Nagasaka and Y. Tanaka, “Automatic Video Indexing and Full-Video Search for Object Appearances,” *Visual Database Systems, II*, IFIP, Elsevier Science Publishers B. V., October, 1992, p. 113 -- 127.
7. Mikihiro Ioka, “A method of defining the similarity of images on the basis of color information”, Technical Report RT-0030, IBM Tokyo Research Lab, 1989.
8. A.K. Jain and F. Farrokhia,” Unsupervised Texture Segmentation Using Gabor Filters,” Pattern Recognition, Vol. 24, No. 12, pp. 1167-86, 1991.
9. P. Brodatz, Textures: a Photographic Album for Artists and Designers, Dover, New York, 1965.
10. J.R. Smith and S.-F. Chang, “Texture Classification and Discrimination Using Wavelet Subband Features for Image Databases,” Intern. Conference on Image Processing, Austin, Nov. 1994.
11. S. Mallat and S. Zhong, “Characterization of Signals from Multiscale Edges,” IEEE T-PAMI, Vol. 14, No. 7, July 1992, pp. 710-32.
12. Keinosuke Fukunaga, *Introduction to statistical pattern recognition*, 2nd ed, Academic Press, Boston, 1990.

Entities	Attributes	Relationship
Video	source (news, movie, commercial), camera control (scene change, zoom, pan, wiping) type (action, drama, news, medical)	spatial/temporal
Image	source (art, satellite, scenery), perceptual quality	spatial/temporal
Object	person, car, building	spatial/temporal, action, compositional
Region	texture, shape, color	spatial/temporal
Pixel	intensity, color	

FIGURE 1. A hierarchy of different levels of visual content description.



(a) (b)

FIGURE 2. (a) Interface Showing Texture Keys, (b) Results of Query-by-Texture using Brodatz texture 23 as the texture-key.

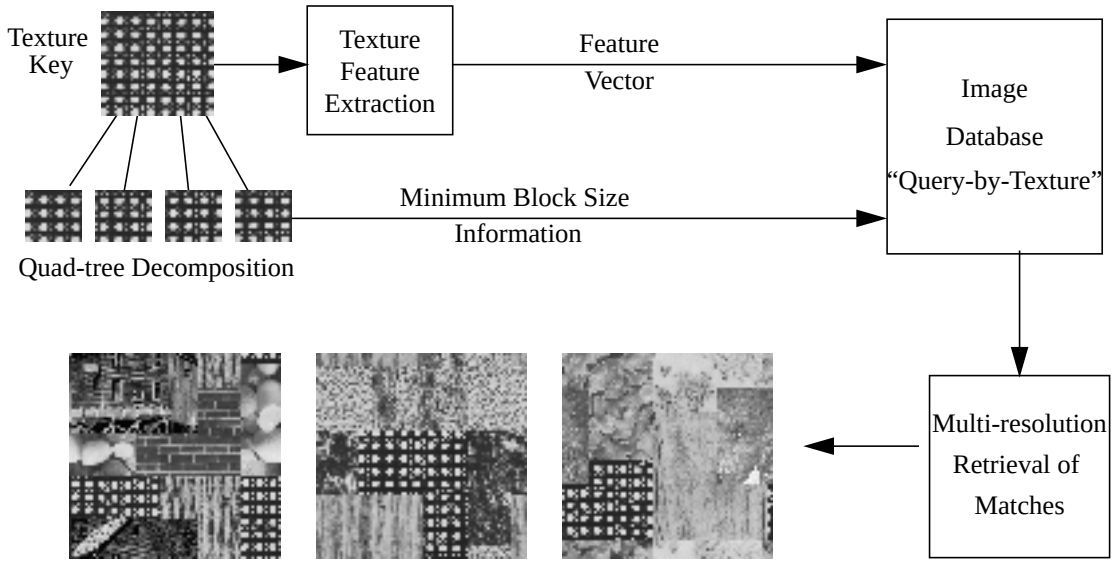


FIGURE 3. "Query-by-Texture" -- using a cut from Brodatz texture D101 as a texture-key to search through the image database. The three closest matches shown at the bottom are returned to user using multi-resolution retrieval.

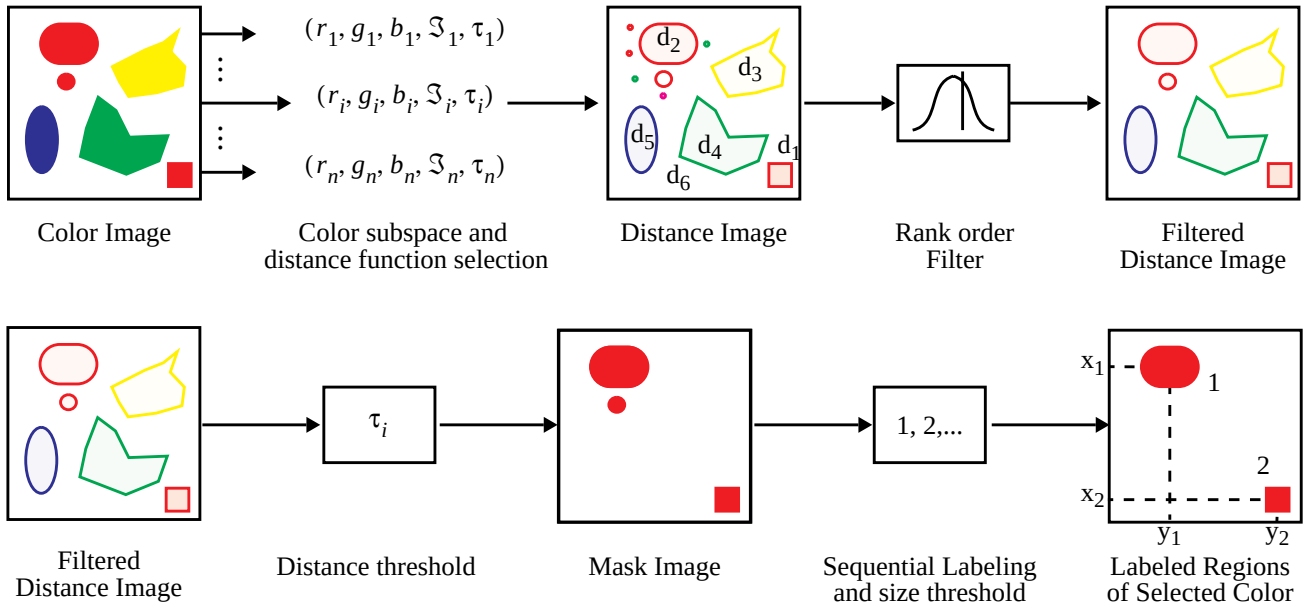


FIGURE 4. Procedure of single color indexing.

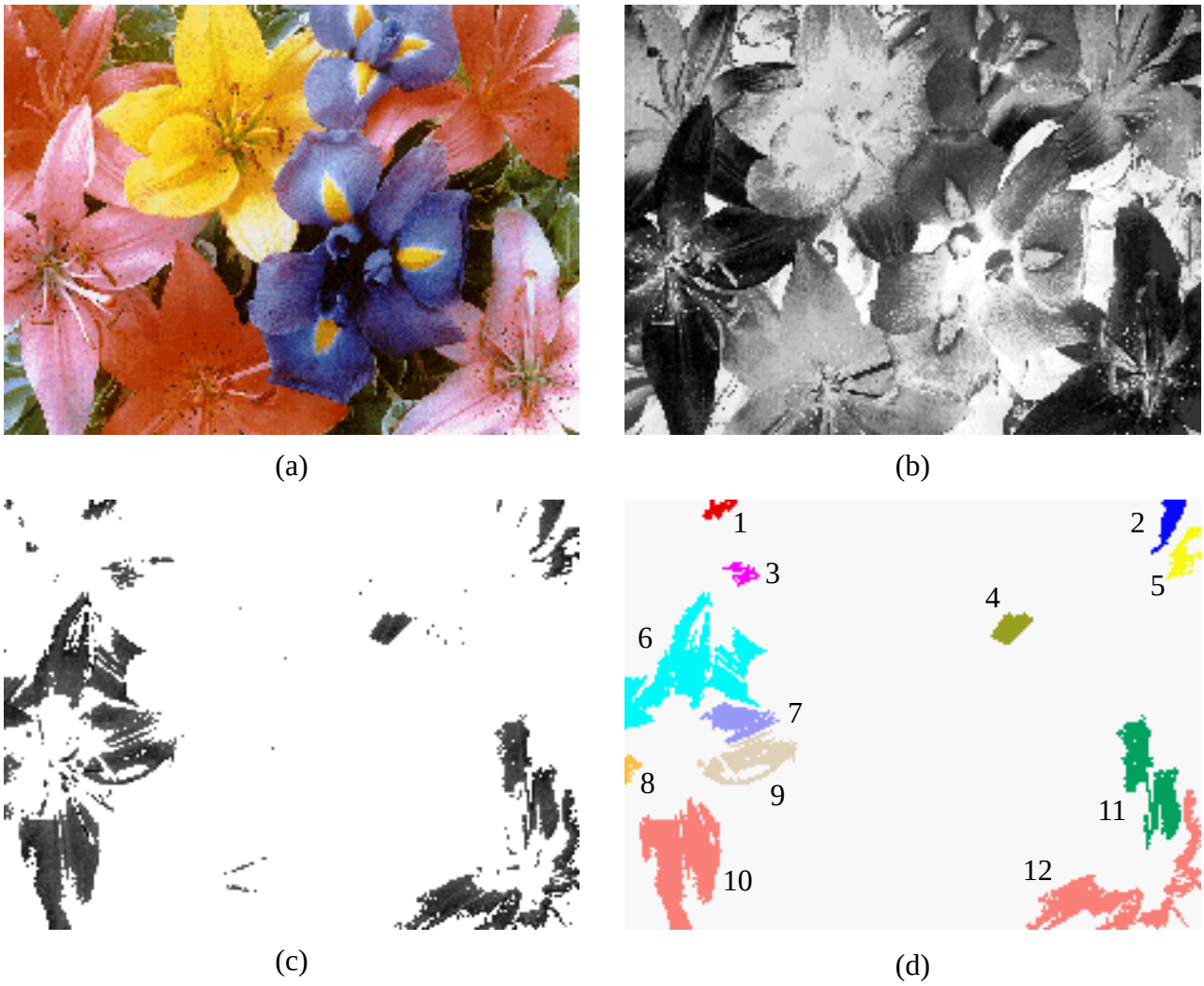
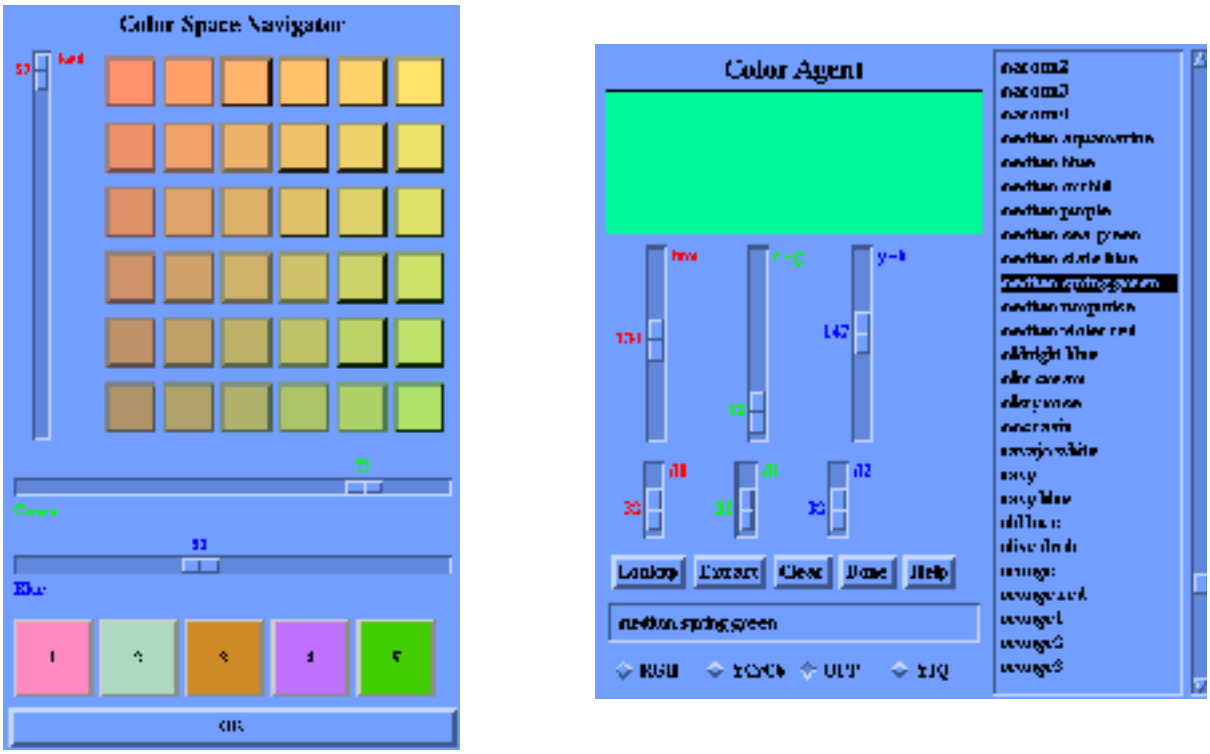


FIGURE 5. (a) Color Flowers image, pink regions to be extracted, (b) distance image from flowers image to pink point in color space, (c) after rank-order filtering and distance thresholding of distance image, (d) after sequential labelling and size thresholding -- isolated color regions now used to populate the database.



(a) (b)

FIGURE 6. (a) Color Space Navigator — a tool that allows users continuous navigation through 3-D color space; the 6x6 color pane displays local space. (b) Color Agent — another tool that allows user to define single color query by using text description and multiple color space navigation.

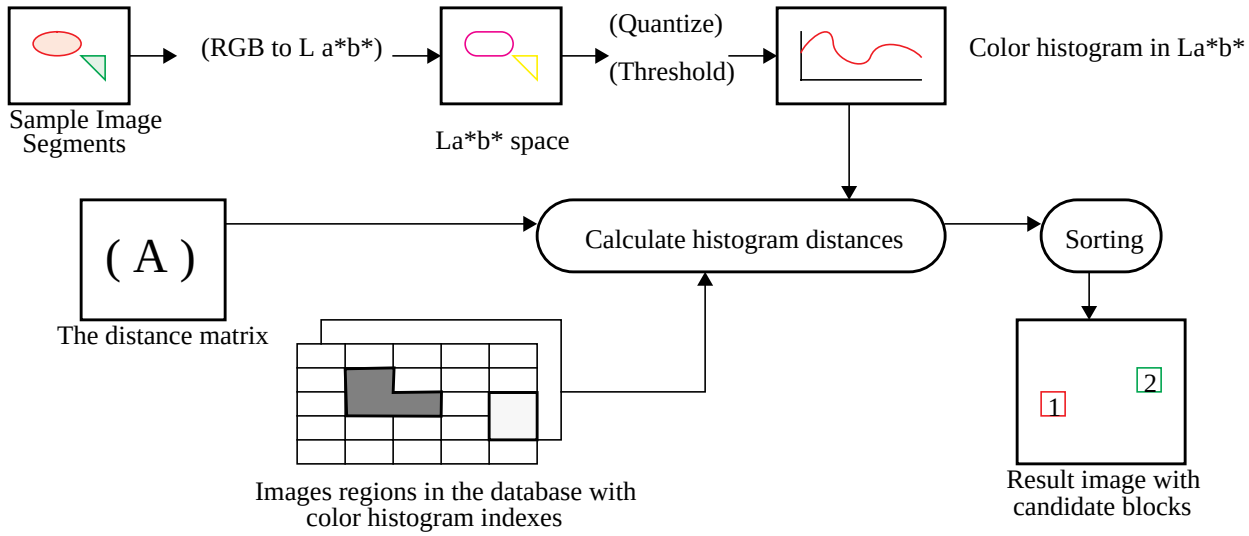
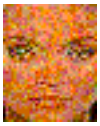


FIGURE 7. Procedure for color histogram indexing.

(a)

(b)



(c)

(d)

FIGURE 8. Query examples and results. (a) query sample - human face, (b) query result- abba.ppm, (c) query example - launching shuttle, (d) query result - gerboth.ppm.

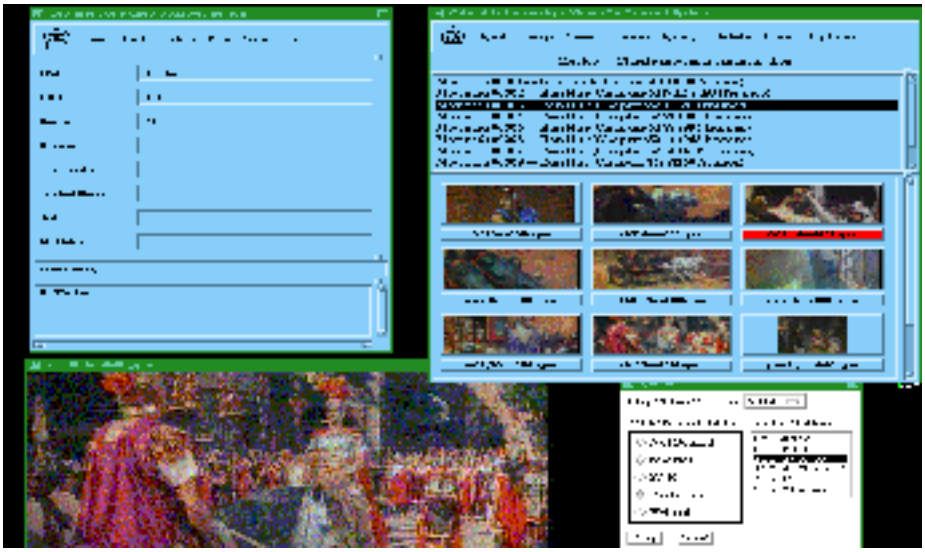


FIGURE 9. Current user interface of Columbia's interactive video browsing tools. This snapshot shows the video scene cut browser, the video playback interface, the QoS setup panel, and the bibliographic search database.