Query by Image and Video Content: The QBIC System

Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker

IBM Almaden Research Center

QBIC* lets users find pictorial information in large image and video databases based on color, shape, texture, and sketches. QBIC technology is part of several IBM products.

*To run an interactive query, visit the QBIC Web server at http://wwwqbic.almaden.ibm.com/.

Picture yourself as a fashion designer needing images of fabrics with a particular mixture of colors, a museum cataloger looking for artifacts of a particular shape and textured pattern, or a movie producer needing a video clip of a red car-like object moving from right to left with the camera zooming. How do you find these images? Even though today's technology enables us to acquire, manipulate, transmit, and store vast on-line image and video collections, the search methodologies used to find pictorial information are still limited due to difficult research problems (see "Semantic versus nonsemantic" sidebar). Typically, these methodologies depend on file IDs, keywords, or text associated with the images. And, although powerful, they

- don't allow queries based directly on the visual properties of the images,
- are dependent on the particular vocabulary used, and
- don't provide queries for images similar to a given image.

Research on ways to extend and improve query methods for image databases is widespread, and results have been presented in workshops, conferences,[1,2] and surveys.

We have developed the QBIC (Query by Image Content) system to explore content-based retrieval methods. QBIC allows queries on large image and video databases based on

- example images,
- user-constructed sketches and drawings,
- selected color and texture patterns,

Semantic versus nonsemantic information

At first glance, content-based querying appears deceptively simple because we humans seem to be so good at it. If a program can be written to extract semantically relevant text phrases from images, the problem may be solved by using currently available text-search technology. Unfortunately, in an unconstrained environment, the task of writing this program is beyond the reach of current technology in image understanding. At an artificial intelligence conference several years ago, a challenge was issued to the audience to write a program that would identify all the dogs pictured in a children's book, a task most 3-year-olds can easily accomplish. Nobody in the audience accepted the challenge, and this remains an open problem.

Perceptual organization—the process of grouping image features into meaningful objects and attaching semantic descriptions to scenes through model matching—is an unsolved problem in image understanding. Humans are much better than computers at extracting semantic descriptions from pictures. Computers, however, are better than humans at measuring properties and retaining these in long-term memory.

One of the guiding principles used by QBIC is to let computers do what they do best—quantifiable measurement—and let humans do what they do best—attaching semantic meaning. QBIC can find "fish-shaped objects," since shape is a measurable property that can be extracted. However, since fish occur in many shapes, the only fish that will be found will have a shape close to the drawn shape. This is not the same as the much harder semantical query of finding all the pictures of fish in a pictorial database.

0018-9162/95/$4.00 · 1995 IEEE

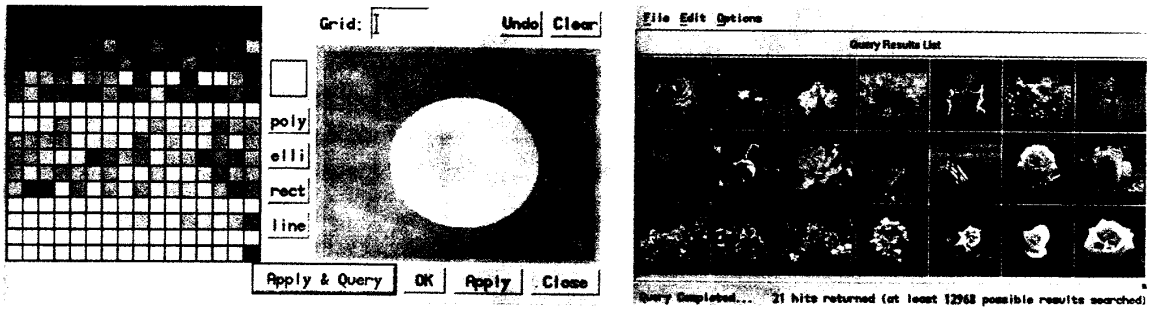


Figure 1. QBIC query by drawn color. Drawn query specification on left; best 21 results sorted by similarity to the query on right. The results were selected from a 12,968-picture database.

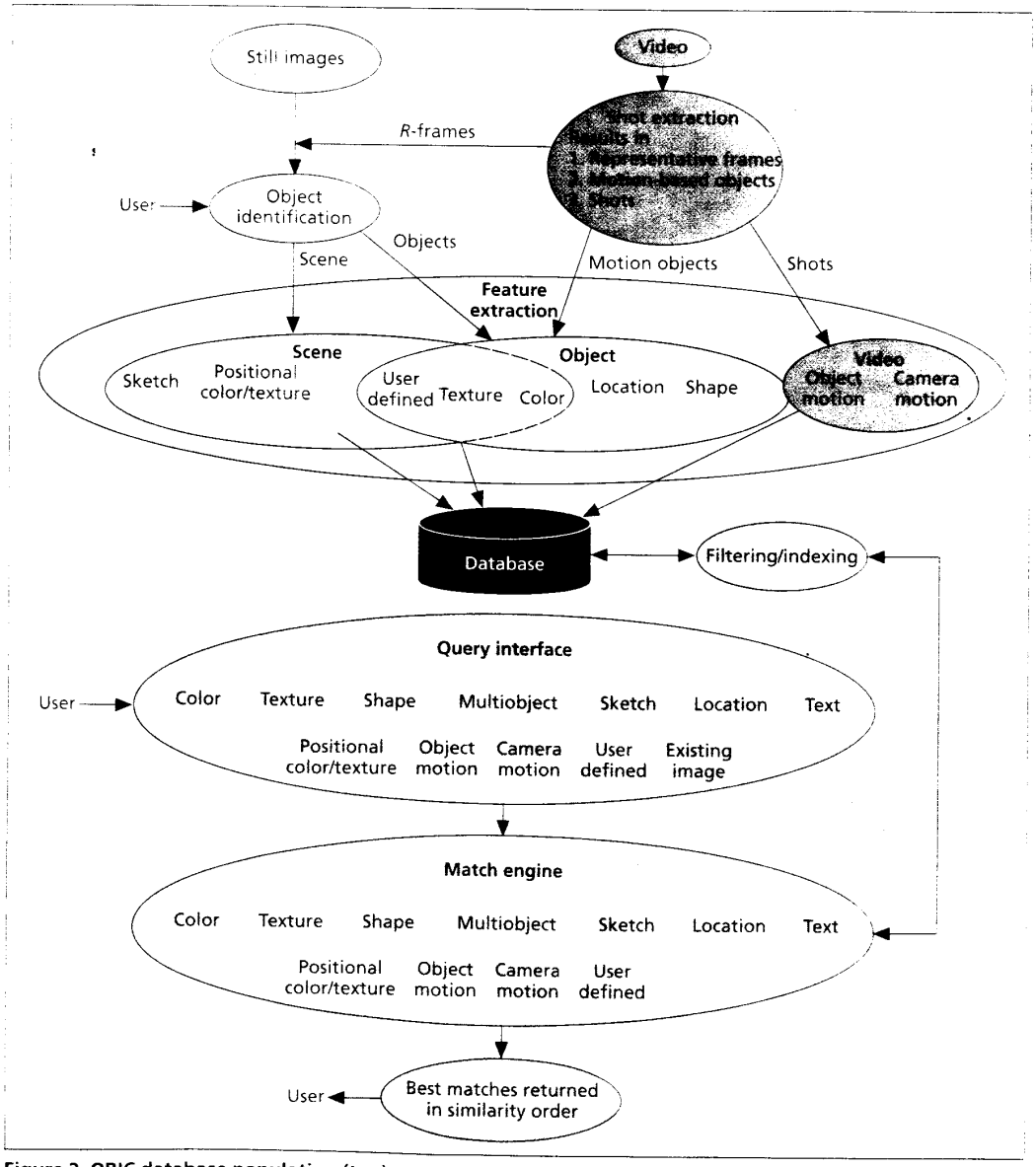


Figure 2. QBIC database population (top) and query (bottom) architecture.

- camera and object motion, and
- other graphical information.

Two key properties of QBIC are (1) its use of image and video content—computable properties of color, texture, shape, and motion of images, videos, and their objects—in the queries, and (2) its graphical query language in which queries are posed by drawing, selecting, and other graphical means. Related systems, such as MIT's Photobook[3] and the Trademark and Art Museum applications from ETL,[4] also address these common issues. This article describes the QBIC system and demonstrates its query capabilities.

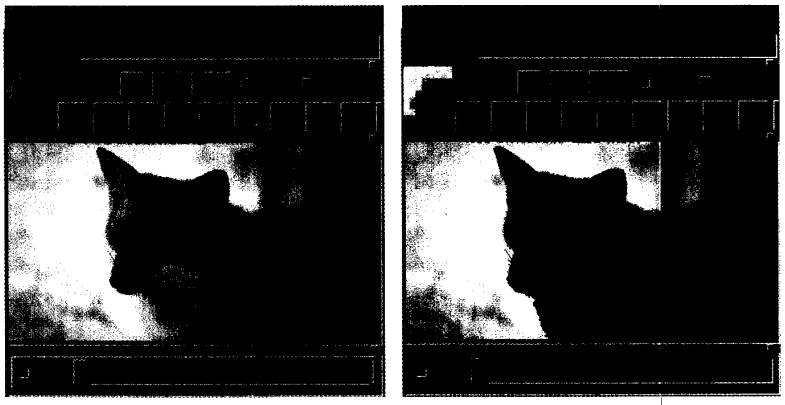


Figure 3. QBIC still image population interface. Entry for scene text at top. Tools in row are polygon outliner, rectangle outliner, ellipse outliner, paintbrush, eraser, line drawing, object translation, flood fill, and snake outliner.

QBIC SYSTEM OVERVIEW

Figure 1 illustrates a typical QBIC query.* The left side shows the query specification, where the user painted a large magenta circular area on a green background using standard drawing tools. Query results are shown on the right: an ordered list of "hits" similar to the query. The order of the results is top to bottom, then left to right, to support horizontal scrolling. In general, all queries follow this model in that the query is specified by using graphical means—drawing, selecting from a color wheel, selecting a sample image, and so on—and results are displayed as an ordered set of images.

To achieve this functionality, QBIC has two main components: database population (the process of creating an image database) and database query. During the population, images and videos are processed to extract features describing their content—colors, textures, shapes, and camera and object motion—and the features are stored in a database. During the query, the user composes a query graphically. Features are generated from the graphical query and then input to a matching engine that finds images or videos from the database with similar features. Figure 2 shows the system architecture.

Data model

For both population and query, the QBIC data model has

- still images or scenes (full images) that contain objects (subsets of an image), and
- video shots that consist of sets of contiguous frames and contain motion objects.

For still images, the QBIC data model distinguishes between "scenes" (or images) and "objects." A scene is an image or single representative frame of video. An object is a part of a scene—for example, the fox in Figure 3—or a moving entity in a video. For still image database population, features are extracted from images and objects and stored in a database as shown in the top left part of Figure 2.

Videos are broken into clips called shots. Representative frames, or *r*-frames, are generated for each extracted shot. *R*-frames are treated as still images, and features are extracted and stored in the database. Further processing of shots generates motion objects—for example, a car moving across the screen.

Queries are allowed on objects ("Find images with a red, round object"), scenes ("Find images that have approximately 30-percent red and 15-percent blue colors"), shots ("Find all shots panning from left to right"), or any combination ("Find images that have 30 percent red and contain a blue textured object").

In QBIC, similarity queries are done against the database of pre-extracted features using distance functions between the features. These functions are intended to mimic human perception to approximate a perceptual ordering of the database. Figure 2 shows the match engine, the collection of all distance functions. The match engine interacts with a filtering/indexing module (see "Fast searching and indexing" sidebar, next page) to support fast searching methodologies such as indexing. Users interact with the query interface to generate a query specification, resulting in the features that define the query.

DATABASE POPULATION

In still image database population, the images are reduced to a standard-sized icon called a thumbnail and annotated with any available text information. Object identification is an optional but key part of this step. It lets users manually, semiautomatically, or fully automatically identify interesting regions—which we call objects—in the images. Internally, each object is represented as a binary mask. There may be an arbitrary number of objects per image. Objects can overlap and can consist of multiple disconnected components like the set of dots on a polka-dot dress. Text, like "baby on beach," can be associated with an outlined object or with the scene as a whole.

Object-outlining tools

Ideally, object identification would be automatic, but this is generally difficult. The alternative—manual identification—is tedious and can inhibit query-by-content

** The scene image database used in the figures consists of about 7,450 images from the Mediasource Series of images and audio from Applied Optical Media Corp., 4,100 images from the PhotoDisc sampler CD, 950 images from the Corel Professional Photo CD collection, and 450 images from an IBM collection.*

Fast searching and indexing

Indexing tabular data for exact matching or range searches in traditional databases is a well-understood problem, and structures like *B*-trees provide efficient access mechanisms. In this scenario, indexing assures sublinear search while maintaining completeness; that is, all records satisfying the query are returned without the need for examining each record in the database. However, in the context of similarity matching for visual content, traditional indexing methods may not be appropriate. For queries in which similarity is defined as a distance metric in high-dimensional feature spaces (for example, color histogram queries), indexing involves clustering and indexable representations of the clusters. In the case of queries that combine similarity matching with spatial constraints on objects, the problem is more involved. Data structures for fast access of high-dimensional features for spatial relationships must be invented.

In a query, features from the database are compared to corresponding features from the query specification to determine which images are a good match. For a small database, sequential scanning of the features followed by straightforward similarity computations is adequate. But as the database grows, this combination can be too slow. To speed up the queries, we have investigated a variety of techniques. Two of the most promising follow.

Filtering

A computationally fast filter is applied to all data, and only items that pass through the filter are operated on by the second stage, which computes the true similarity metric. For example, in QBIC we have shown that color histogram matching, which is based on a 256-dimensional color histogram and requires a 256 matrix-vector multiply, can be made efficient by filtering. The filtering step employs a much faster computation in a 3D space with no loss in accuracy. Thus, for a query on a database of 10,000 elements, the fast filter is applied to produce the best 1,000 color histogram matches. These filtered histograms are subsequently passed to the slower complete matching operation to obtain, say, the best 200 matches to display to a user, with the guarantee that the global best 200 in the database have been found.

Indexing

For low-dimensional features such as average color and texture (each 3D), multidimensional indexing methods such as *R**-trees can be used. For high-dimensional features—for example, our 20-dimensional moment-based shape feature vector—the dimensionality is reduced using the K-L, or principal component, transform. This produces a low-dimensional space, as low as two or three dimensions, which could be indexed by using *R**-trees.

applications. As a result, we have devoted considerable effort to developing tools to aid in this step. In recent work, we have successfully used fully automatic unsupervised segmentation methods along with a foreground/background model to identify objects in a restricted class of images. The images, typical of museums and retail catalogs, have a small number of foreground objects on a generally separable background. Figure 4 shows example results. Even in this domain, robust algorithms are required because of the textured and variegated backgrounds.

We also provide semiautomatic tools for identifying objects. One is an enhanced flood-fill technique. Flood-fill methods, found in most photo-editing programs, start from a single object pixel and repeatedly add adjacent pixels whose values are within some given threshold of the original pixel. Selecting the threshold, which must change from image to image and object to object, is tedious. We automatically calculate a dynamic threshold by having the user click on background as well as object points. For reasonably uniform objects that are distinct from the background, this operation allows fast object identification

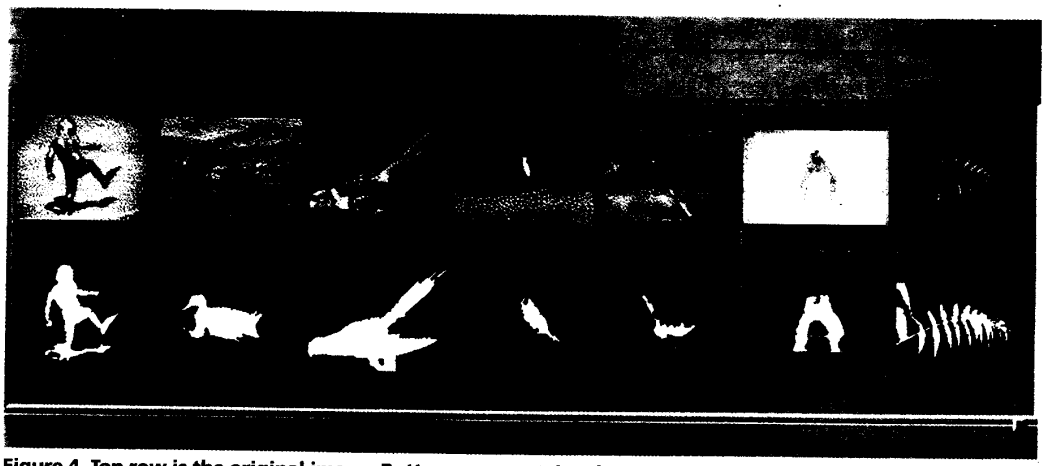


Figure 4. Top row is the original image. Bottom row contains the automatically extracted objects using a foreground/background model. Heuristics encode the knowledge that objects tend to be in the center of the picture.

without manually adjusting a threshold. The example in Figure 3 shows an object, a fox, identified by using only a few clicks.

We designed another outlining tool to help users track object edges. This tool takes a user-drawn curve and automatically aligns it with nearby image edges. Based on the "snakes" concept developed in recent computer vision research, the tool finds the curve that maximizes the image gradient magnitude along the curve.

The spline snake formulation we use allows for smooth solutions to the resulting nonlinear minimization problem. The computation is done at interactive speeds so that, as the user draws a curve, it is "rubber-banded" to lie along object boundaries.

Figure 5. Scene cuts automatically extracted from a 1,148-frame sales demo from Energy Productions.

Video data

For video data, database population has three major components:

- shot detection,
- representative frame creation for each shot, and
- derivation of a layered representation of coherently moving structures/objects.

Shots are short sequences of contiguous frames that we use for annotation and querying. For instance, a video clip may consist of a shot smoothly panning over the skyline of San Francisco, switching to a panning shot of the Bay meeting the ocean, and then to one that zooms to the Golden Gate Bridge. In general, a set of contiguous frames may be grouped into a shot because they

- depict the same scene,
- signify a single camera operation,
- contain a distinct event or an action like a significant presence and persistence of an object, or
- are chosen as a single indexable entity by the user.

Our effort is to detect many shots automatically in a preprocessing step and provide an easy-to-use interface for the rest.

SHOT DETECTION. Gross scene changes or scene cuts are the first indicators of shot boundaries. Methods for detecting scene cuts proposed in the literature essentially fall into two classes: (1) those based on global representations like color/intensity histograms without any spatial information, and (2) those based on measuring differences between spatially registered features like intensity differences. The former are relatively insensitive to motion but can miss cuts when scenes look quite different but have similar distributions. The latter are sensitive to moving objects and camera. We have developed a method that combines the strengths of the two classes of detection. We use a robust normalized correlation measure that allows for small motions and combines this with a histogram distance measure.[5] Results on a few videos containing from 2,000 to 5,000 frames show no misses and only a few false cuts. Algorithms for signaling edit effects like fades and dissolves are under development. The results of cut detection on a video containing commercial advertisement clips are shown in Figure 5.

Shots may also be detected by finding changes in camera operation. Common camera transformations like zoom, pan, and illumination changes can be modeled as unknown affine 2×2 matrix transformations of the 2D image coordinate system and of the image intensities themselves. We have developed an algorithm[6] that computes the dominant global view transformation while it remains insensitive to nonglobal changes resulting from independently moving

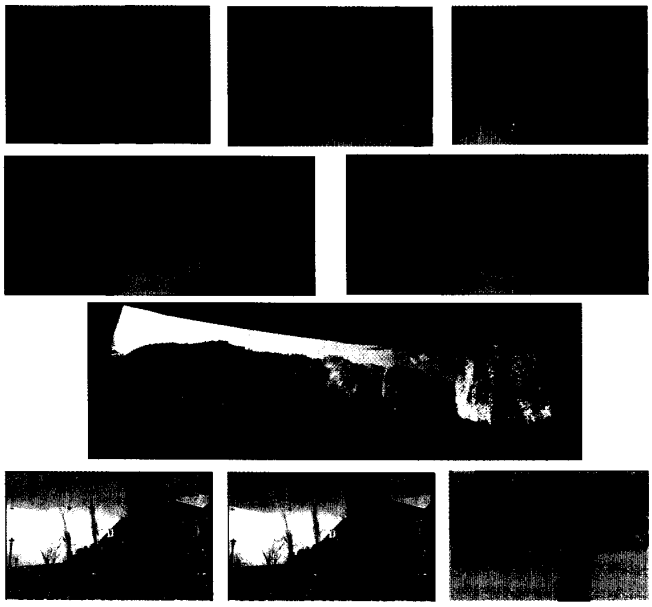


Figure 6. Top: Three frames from the *charlie* sequence and the resulting dynamic mosaics of the entire sequence. Below that is a mosaic from a video sequence of Yosemite National Park. Bottom: Original images and segmented motion layers for the flower garden sequence in which only the camera is moving. The flower bed, tree, and background have been separated into three layers shown in different shades of gray.

objects and local brightness changes. The affine transformations that result from this computation can be used for camera operation detection, shot boundary detection based on the camera operation, and creating a synthetic *r*-frame wherever appropriate.

Shot boundaries can also be defined on the basis of events: appearance/disappearance of an object, distinct change in the motion of an object, or similar events. For instance, segmenting an object of interest based on its appearance and/or motion, and tracking it throughout its significant presence may be used for defining shots.

REPRESENTATIVE FRAME GENERATION. Once the shot boundaries have been detected, each shot is represented using an *r*-frame. *R*-frames are used for several purposes. First, during database population, *r*-frames are treated as still images in which objects can be identified by using the previously described methods. Secondly, during query, they are the basic units initially returned in a video query. For example, in a query for shots that are dominantly red, a set of *r*-frames will be displayed. To see the actual video shot, the user clicks on the displayed *r*-frame icon.

The choice of an *r*-frame could be as simple as a particular frame in the shot: the

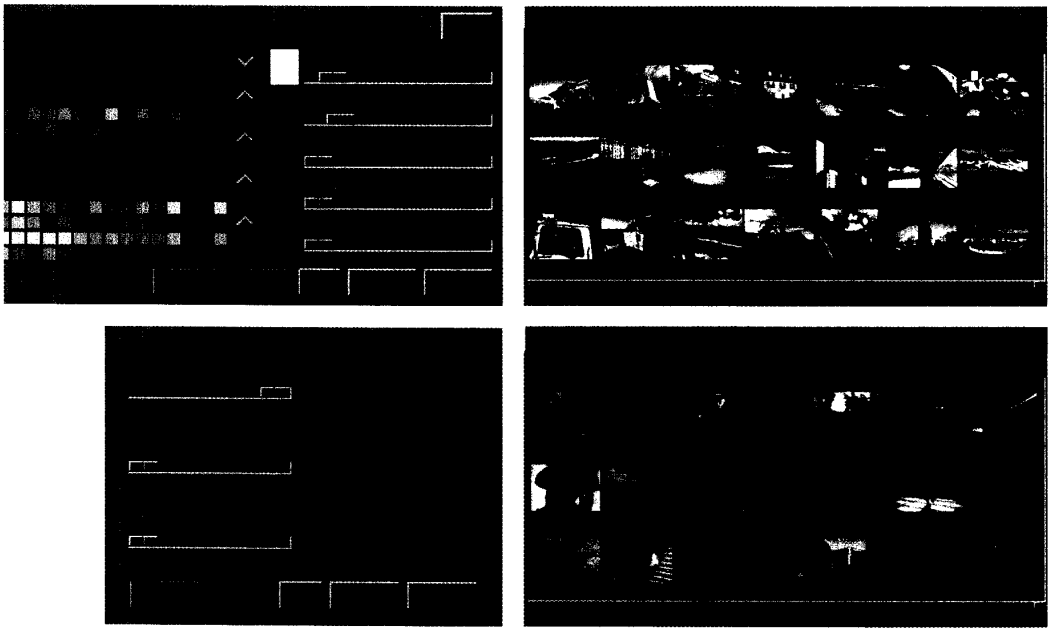
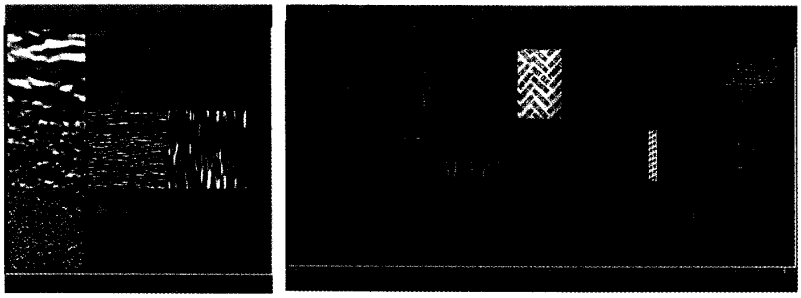


Figure 7. Top: Query by histogram color. Histogram color query specification on left; best 21 results from a 12,966-picture database on right. Bottom: A query for a red video *r*-frame. The color picker is on the left; the resulting *r*-frame thumbnails of the best matches are shown on the right. Each thumbnail is an active button that allows the user to play the shot.



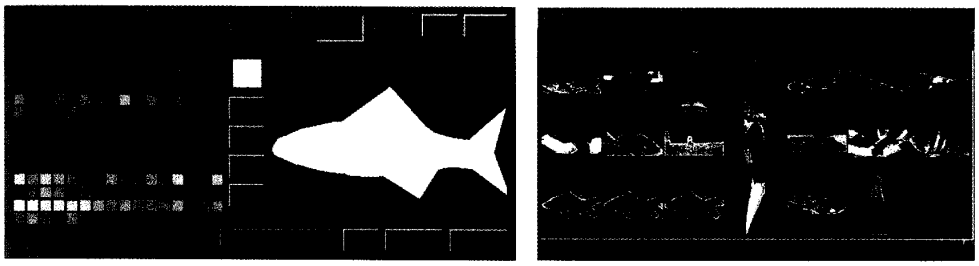


Figure 8. Top: Query by texture. Texture sampler on left (query specification is right middle texture); best 21 results from a 12,966-picture database on right. Bottom: Query by shape. User input shape on left and query results on right.

first, the last, or the middle. However, in situations such as a long panning shot, no single frame may be representative of the entire shot. We use a synthesized *r*-frame[7,8] created by seamlessly mosaicking all the frames in a given shot using the computed motion transformation of the dominant background. This frame is an authentic depiction of all background captured in the whole shot. Any foreground object can be superimposed on the background to create a single, static visual representation of the shot. The *r*-frame mosaicking is done by using warping transforms that result from automatic dominant motion computation. Given a video sequence with dominant motion and moving object(s), the 2D motion estimation algorithm is applied between consecutive pairs of frames. Then, a reference frame is chosen, and all the frames are warped into the coordinate system of the reference frame to create the mosaicked *r*-frame.

Figure 6 illustrates mosaic-based *r*-frame creation on a video sequence of an IBM commercial. Three frames of this sequence plus the final mosaic are shown. Two dominant-component-only mosaics of the *charlie* sequence are shown in Figure 6. In one case, the moving object has been removed from the mosaic by using temporal median filtering on the frames in the shot. In the other case, the moving object remains from the first frame in the sequence. We are also developing methods to visually represent the object motion in the *r*-frame.

LAYERED REPRESENTATION. To facilitate automatic segmentation of independently moving objects and significant structures, we take further advantage of the time-varying nature of video data to derive what is called a layered representation[9] of video. The different layers are used to identify significant objects in the scene for feature computations and querying. Our algorithm divides a shot into a number of layers, each with its own 2D affine motion parameters and region of support in each frame.[10]

The algorithm is first illustrated on a shot where the scene is static but the camera motion induces parallax motion onto the image plane due to the different depths in the scene. Therefore, surfaces and objects that may correspond to semantically useful entities can be segmented based on the coherence of their motion. Figure 6 (bottom row) shows the results for the layers from the flower garden sequence.

SAMPLE QUERIES

For each full-scene image, identified image object, *r*-frame, and identified video object resulting from the above processing, a set of features is computed to allow content-based queries. The features are computed and stored during database population. We present a brief description of the features and the associated queries. Mathematical details on the features and matching methods can be found in Ashley et al.[11] and Niblack et al.[12]

Average color queries let users find images or objects that are similar to a selected color, say from a color wheel, or to the color of an object. The feature used in the query is a 3D vector of Munsell color coordinates. Histogram color queries return items with matching color distributions—say, a fabric pattern with approximately 40 percent red and 20 percent blue. For this case, the underlying feature is a 256-element histogram computed over a quantized version of the color space.

Figure 7 shows a histogram query on still images and a color query on video *r*-frames. Note that in the query specification for the histogram query of Figure 7, the user has selected percentages of two colors (blue and white) by adjusting sliders. Using such a query, an advertising agent could, for example, search for a picture of a beach scene, one predominantly blue (for sky and water) and white (for sand and clouds); or find images with similar color spreads for a uniform ad campaign. The average color query demonstrates a query against a video shot database where the user is searching for red *r*-frames. Again, the query specification is on the left and the best hits are on the right.

Figure 8 shows an example texture query. In this case, the query is specified by selecting from a sampler—a set of

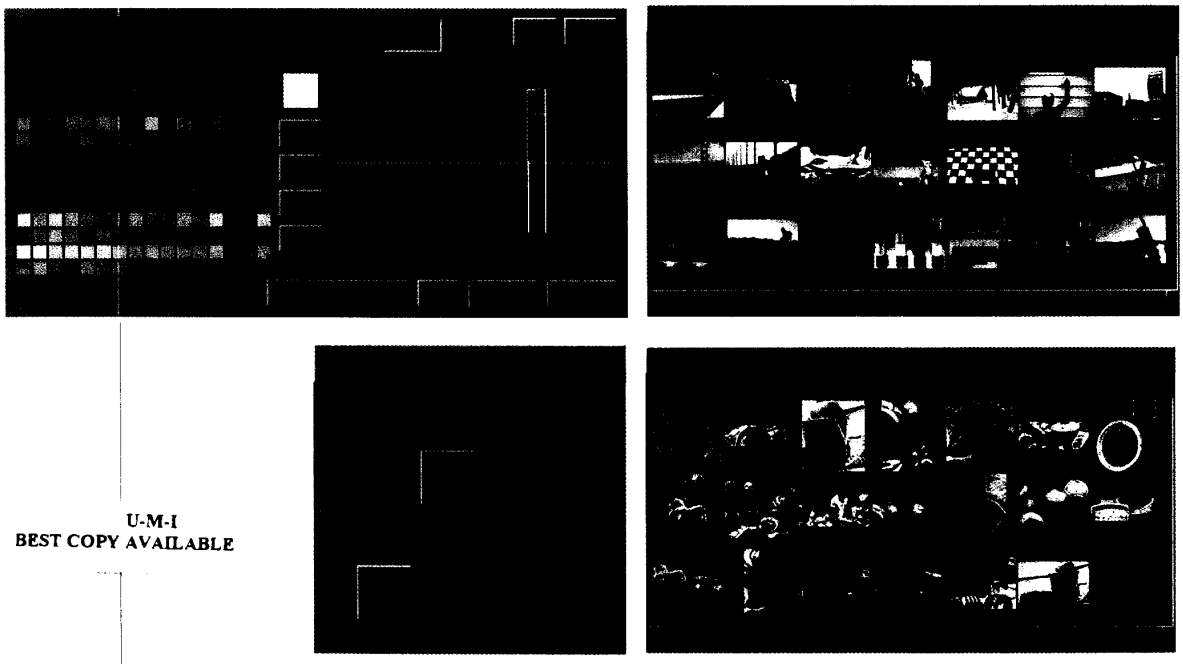


Figure 9. Top: Query by sketch. Sketched query specification on left; best 21 hits from a 12,965-image database on right. Bottom: A Multi-* object query. The query specification on the left describes a query for images with a red round object and a green textured object. Best 20 matches shown on right.

prestored example images. The underlying texture features are mathematical representations of coarseness, contrast, and directionality features. Coarseness measures the scale of a texture (pebbles versus boulders), contrast describes its vividness, and directionality describes whether it has a favored direction (like grass) or not (like a smooth object).

An object shape query is shown in Figure 8. In this case, the query specification is the drawn shape on the left. Area, circularity, eccentricity, major-axis direction, features derived from the object moments, and a set of tangent angles around the object perimeter are the features used to characterize and match shapes.

Figure 9 illustrates query by sketch. In this case, the query specification is a freehand drawing of the dominant lines and edges in the image. The sketch feature is an automatically extracted reduced-resolution "edge map." Matching is done by using a template-matching technique.

A multiobject query asking for images that contain both a red round object and a green textured object is shown in the bottom of Figure 9. The features are standard color and texture. The matching is done by combining the color and texture distances. Combining distances is applied to arbitrary sets of objects and features to implement logical And semantics.

WE HAVE DESCRIBED A PROTOTYPE SYSTEM that uses image and video content as the basis for retrievals. Technology from this prototype has already moved into a commercial stand-alone product, IBM's Ultimedia Manager, and is part of IBM's Digital Library and DB2 series of products. Other companies are beginning to offer products with similar capabilities. Key challenges remain in making this technology pervasive and useful.

ANNOTATION AND DATABASE POPULATION TOOLS. Automatic methods (such as our Positional Color query) that don't rely on object identification, methods that identify objects automatically as in the museum image example, fast and easy-to-use semiautomatic outlining tools, and motion-based segmentation algorithms will enable additional application areas.

FEATURE EXTRACTION AND MATCHING METHODS. New mathematical representations of video, image, and object attributes that capture "interesting" features for retrieval are needed. Features that describe new image properties such as alternate texture measures or that are based on fractals or wavelet representations, for example, may offer advantages of representation, indexability, and ease of similarity matching.

INTEGRATION WITH TEXT AND PARAMETRIC ANNOTATION. Query by visual content complements and extends existing query methods. Systems must be able to integrate queries combining date, subject matter, price, and availability with content properties such as color, texture, and shape.

EXTENSIBILITY AND FLEXIBILITY. System architectures must support the addition of new features and new matching/similarity measures. Real applications often require

new features, say a face-matching module, to add to their existing content-based retrieval capabilities.

USER INTERFACE. The user interface must be designed to let users easily select content-based properties, allow these properties to be combined with each other and with text or parametric data, and let users reformulate queries and generally navigate the database.

INDEXING AND PERFORMANCE. As image and video collections grow, system performance must not slow down proportionately. Indexing, clustering, and filtering methods must be designed into the matching methods to maintain performance.

With these technologies, the QBIC paradigm of visual content querying, combined with traditional keyword and text querying, will lead to powerful search engines for multimedia archives. Applications will occur in areas such as decision support for retail marketing, on-line stock photo and video management, cataloging for library and museum collections, and multimedia-enabled applications in art, fashion, advertising, medicine, and science. ▮

References

1. IFIP, *Visual Database Systems I and II*, Elsevier Science Publishers, North-Holland, 1989 and 1992.
2. *Proc. Storage and Retrieval for Image and Video Databases I, II, and III*, Vols. 1,908; 2,185; and 2,420; W. Niblack and R. Jain, eds., SPIE, Bellingham, Wash., 1993, 1994, and 1995.
3. A. Pentland, R.W. Picard, and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *Proc. Storage and Retrieval for Image and Video Databases II*, Vol. 2,185, SPIE, Bellingham, Wash., 1994, pp. 34-47.
4. T. Kato, T. Kurita, and H. Shimogaki, "Intelligent Visual Interaction with Image Database Systems—Toward the Multimedia Personal Interface," *J. Information Processing* (Japan), Vol. 14, No. 2, 1991, pp. 134-143.
5. A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances," *Visual Database Systems, II, IFIP Trans. A-7*, Elsevier Science Publishers, North-Holland, 1992, pp. 113-127.
6. H.S. Sawhney, S. Ayer, and M. Gorkani, "Model-Based 2D & 3D Dominant Motion Estimation for Mosaicking and Video Representation," *Proc. Fifth Int'l Conf. Computer Vision*, Order No. PRO7042, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 583-590; http://www.almaden.ibm.com/pub/cs/reports/vision/dominant_motion.ps.Z.
7. Y.T.A. Akutsu, K. Otsuji, and T. Sadakata, "VideoMAP and VideoSpaceIcon: Tools for Anatomizing Video Content," *ACM INTERCHI*, 1993, pp. 131-136.
8. L.A. Teodosio and W. Bender, "Salient Video Stills: Content and Context Preserved," *ACM Int'l Conf. Multimedia*, ACM, New York, 1993.
9. J.Y.A. Wang and E.H. Adelson, "Layered Representation for Motion Analysis," *Proc. Computer Vision and Pattern Recognition Conf.*, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 361-366.
10. S. Ayer and H.S. Sawhney, "Layered Representation of Motion Video Using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding," *Proc. Fifth Int'l Conf. Computer Vision*, Order No. PRO7042, IEEE CS Press, Los

Alamitos, Calif., 1995, pp. 777-784, http://www.almaden.ibm.com/pub/cs/reports/vision/layered_motion.ps.Z.
11. J. Ashley et al., "Automatic and Semiautomatic Methods for Image Annotation and Retrieval in QBIC," *Proc. Storage and Retrieval for Image and Video Databases III*, Vol. 2,420, SPIE, Bellingham, Wash., 1995, pp. 24-25.
12. W. Niblack et al., "The QBIC Project: Querying Images by Content Using Color, Texture, and Shape," *Proc. Storage and Retrieval for Image and Video Databases*, Vol. 1,908, SPIE, Bellingham, Wash., 1993, pp. 173-187.

Myron "Flick" Flickner *architected parts of QBIC and implemented several shape-matching methods and the database population GUI. His current research interests include image and shape representations, content-based image retrieval, and vision-based man-machine interaction.*

Harpreet "Video" Sawhney *has done much of the exploratory and implementation work to extend the QBIC system to video. The beautiful Yosemite mosaic is a direct result of his work. Sawhney's current research interests are in image and video analysis, representation, and indexing; 3D modeling from images; and multimedia information systems.*

Wayne "Dad" Niblack *is the manager of the Machine Vision group at IBM's Almaden Research Center and the project leader of Query by Image Content (QBIC). He implemented early versions of the color-matching code. Niblack's current interests are in image recognition and retrieval.*

Jonathan "Jaybird" Ashley *has been working on implementing positional color queries for QBIC. His current interests are in bird-watching, and in image processing and synthesis for holographic data storage.*

Qian "Cat" Huang *was a postdoctoral scholar at IBM during 1994-1995. She implemented the automatic object-identification program, whose results are shown in Figure 4. Her interests include computer vision, pattern recognition, medical imaging, artificial intelligence, multimedia, parallel computing, and cats.*

Byron "Bike" Dom *was manager of IBM's Machine Vision group before taking a sabbatical during 1995 to the K.U. Leuven in Belgium. Dom architected many parts of video QBIC as well as the automatic identification of objects by using image segmentation. His interests are segmentation, inspection, and information-theoretic approaches to vision.*

Monika "MPEG" Gorkani *was with IBM during 1994-1995 working on video QBIC. She enabled QBIC for MPEG video as well as implementing the warping code used for mosaicking. Her research interests include video understanding and texture analysis.*

Jim "Symbols-are-fun" Hafner *has solved many of the difficult mathematical problems related to QBIC. He has also implemented much of the positional and histogram color QBIC querying facility. His primary interests are in number theory, but he has worked in complexity theory, matrix theory, and image processing. Hafner loves to solve "impossible" integration problems.*

Denis "Software Wizard" Lee *designed and developed much of the QBIC system, including the graphical user interface, query engine, and World Wide Web server. His areas of interest include graphics, multimedia, computer-aided VLSI design, image processing, virtual reality, user interfaces, distributed systems, neural networks, and genetic algorithms.*

Dragutin "Human-in-the-loop" Petkovic *manages the Advanced Algorithms, Architectures, and Applications Department. Despite all the managerial responsibilities, he is still involved in marketing, applications, and testing much of the software the group creates. His research interests include image analysis applied to industrial, commercial, and biomedical problems, content-based search, large-image and multimedia databases, and advanced user interfaces.*

David "Chess" Steele*, a Canadian exiled to Silicon Valley since 1983, is involved in finding shots in video sequences as well as in the QBIC WWW server. The strongest chess player in the group, Steele is interested in machine vision, artificial intelligence, and decision analysis.*

Peter "Ski" Yanker *was involved in developing IBM Ultimedia Manager, a product that allows queries of images by color, texture, and shape. His current interests are video annotation, skiing, and hiking.*

Please direct e-mail correspondence to qbicwww@ almaden.ibm.com.