A Fully Automated Content-Based Video Search Engine Supporting Spatiotemporal Queries

Shih-Fu Chang, *Member, IEEE,* William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong

(Invited Paper)

***Abstract*—The rapidity with which digital information, particularly video, is being generated has necessitated the development of tools for efficient search of these media. Content-based visual queries have been primarily focused on still image retrieval. In this paper, we propose a novel, interactive system on the Web, based on the visual paradigm, with spatiotemporal attributes playing a key role in video retrieval. We have developed innovative algorithms for automated video object segmentation and tracking, and use real-time video editing techniques while responding to user queries. The resulting system, called VideoQ (demo available at http://www.ctr.columbia.edu/VideoQ/), is the first on-line video search engine supporting automatic object-based indexing and spatiotemporal queries. The system performs well, with the user being able to retrieve complex video clips such as those of skiers and baseball players with ease.**

***Index Terms*—Content based, information retreival, object oriented, spatiotemporal, video query.**

I. Introduction

THE ease of capture and encoding of digital images has caused a massive amount of visual information to be produced and disseminated rapidly. Hence, efficient tools and systems for searching and retrieving visual information are needed. While there are efficient search engines for text documents today, there are no satisfactory systems for retrieving visual information.

Content-based visual queries (CBVQ) has emerged as a challenging research area in the past few years [7], [13]. While there has been substantial progress with the presence of systems such as QBIC [11], PhotoBook [27], Virage [14], and VisualSEEk [33], most systems only support retrieval of still images. CBVQ research on video databases has not been fully explored yet. Our system, VideoQ, is an advanced content-based video search system, with the following unique features:

- automatic video object segmentation and tracking;
- a rich visual feature library including color, texture, shape, and motion;
- query with multiple objects;
- spatiotemporal constraints on the query;
- interactive querying and browsing over the World-Wide Web;
- compressed-domain video manipulation.

Specifically, we present a novel video search system which allows users to search video based on a rich set of visual features and spatiotemporal relationships. Our objective is to investigate the full potential of visual cues in object-oriented content-based video search. We also support a keyword-based search, where the keywords have been manually generated. While the search on video databases ought to necessarily incorporate the diversity of the media (video, audio, text captions), our present work will integrate well into any such effort.

We will present the visual search paradigm in Section II, elaborate on the system overview in Section III, describe video objects and our automatic video analysis techniques in Sections IV and V, discuss the matching criteria and query resolution in Sections VII and VIII, and finally present some preliminary evaluation results in Section IX.

II. The Visual Paradigm

The fundamental paradigm under which VideoQ operates is the visual one. This implies that the query is principally formulated in terms of elements having visual attributes. VideoQ also supports search using keywords; these keywords have been generated manually, and can be used to "filter" the returned results. The visual features that are stored in the database are generated from an automatic analysis of the video stream. Many retrieval systems, such as PhotoBook, VisualSEEk, and Virage, share this paradigm, but only support still image retrieval.

Video retrieval systems should evolve toward a systematic integration of all available media such as audio, video, and captions. While video engines such as [15], [14], [31], and [24] attempt such an integration, much research on the representation and analysis of each of these different media remains to be done. Those that concentrate on the visual media alone fall into two distinct categories:

- query by example (QBE);
- visual sketches.

In the context of image retrieval, examples of QBE systems include QBIC, PhotoBook, VisualSEEk, Virage, and FourEyes [23]. Examples of sketch-based image retrieval systems include QBIC, VisualSEEk, [17], [16], and [4]. These two

Manuscript received October 31, 1997; revised May 30, 1998. This work was supported by Intel Research Council, the National Science Foundation under a CAREER award (IRI-9501266), IBM under a 1995 Research Partnership (Faculty Development) Award, and by sponsors of the ADVENT project at Columbia University. This paper was presented in part at the ACM Conference on MultiMedia, Seattle, WA, November 1997. This paper was recommended by Associate Editor S. Panchanathan.

The authors are with the Department of Electrical Engineering and the New Media Technology Center, Columbia University, New York, NY 10027 USA.

Publisher Item Identifier S 1051-8215(98)06329-0.

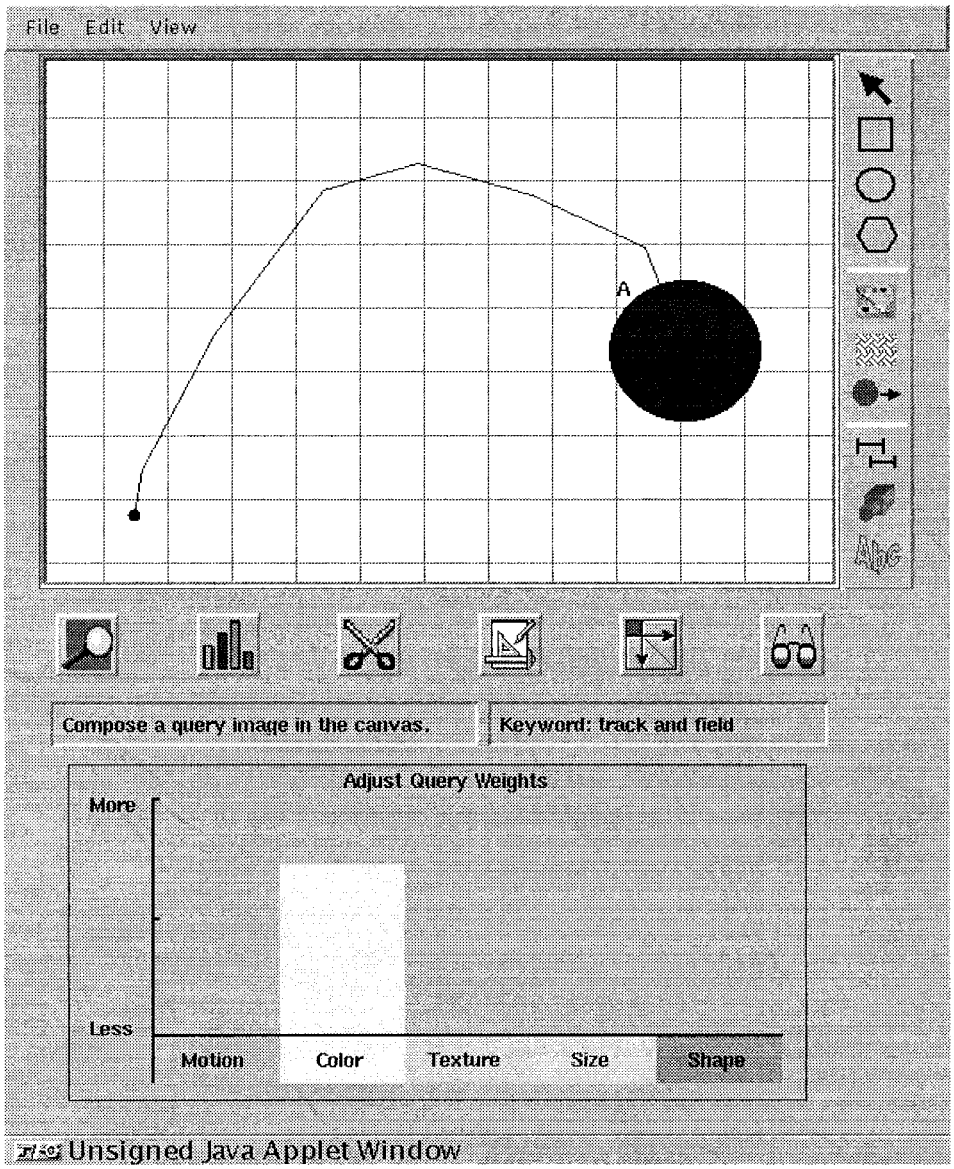

Fig. 1. Visual interface of VideoQ. The figure shows an example query to retrieve video shots of all high-jump sequences in the database. Note that the keywords "track and field" are used to filter the returned results. The results are shown in Fig. 2.

different ways of visually searching image databases may also be accompanied by learning and user feedback [23].

Query by example systems work under the realization that since the "correct" match must lie within the database, one can begin the search with an element of the database itself with the hope that one can guide the user toward the image that he likes over a succession of query examples. In QBE, one can use space-partitioning schemes to precompute hierarchical groupings, which can speed up the database search [23]. While the search speeds up, the groupings are static and need recomputation every time a new video is inserted into the database. QBE, in principle, is easily extensible to video databases as well, but there are some drawbacks. Video shots generally contain a large number of objects, each of which is described by a complex multidimensional feature vector. The complexity arises partly due to the problem of describing shape and motion characteristics.

Sketch-based query systems such as [16] compute the correlation between the sketch and the edge map of each of the images in the database, while in [4], the authors minimize an energy functional to achieve a match. In [17], the authors compute a distance between the wavelet signatures of the sketch and each of the images in the database.

What makes VideoQ powerful is the novel idea of an animated sketch to formulate the query. In an animated sketch, motion and temporal duration are the key attributes assigned to each object in the sketch, in addition to the usual attributes such as shape, color, and texture. Using the visual pallette, we sketch out a scene by drawing a collection of video objects. It is the spatiotemporal relationships between these objects that fully define a scene. An example query is illustrated in Fig. 1.

While we will extensively employ this paradigm, some important observations are to be kept in mind. The visual paradigm works best when there are only a few dominant

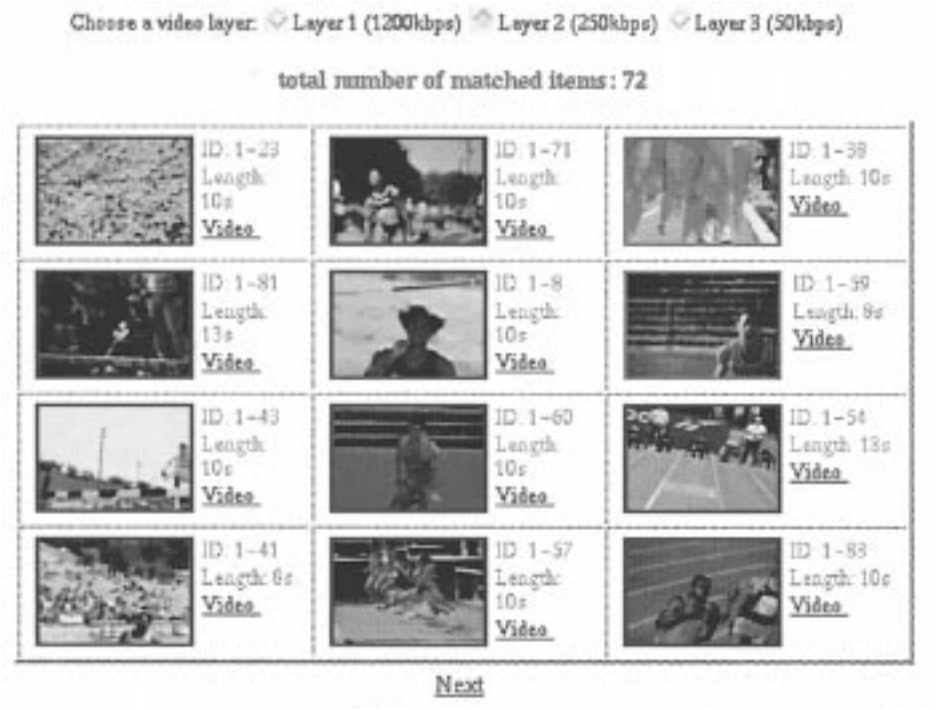

Fig. 2. Results: the displayed retrieved shots which include four excellent matches (the first, third, fourth, and fifth) indicate the importance of the motion attribute in video shot retrieval.

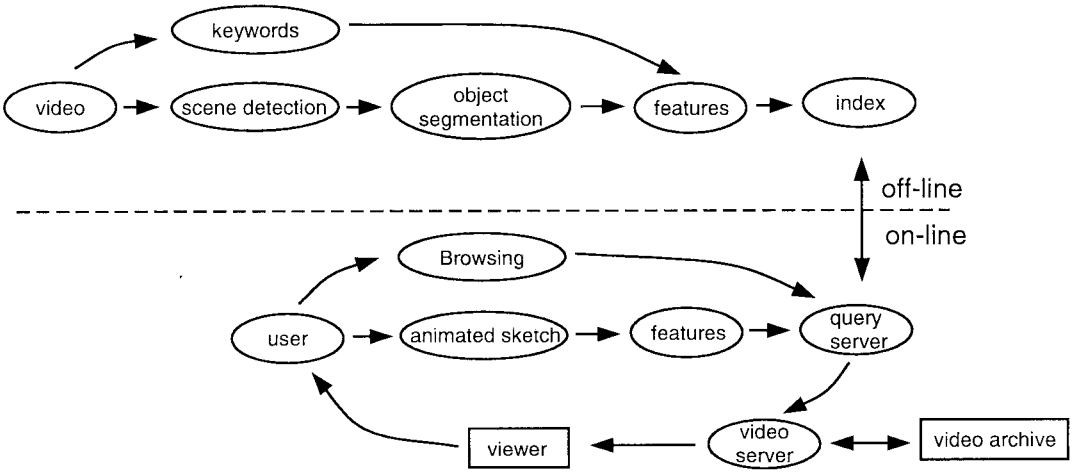

Fig. 3. VideoQ system where the queries are in the form of animated sketches. Both the animated sketch and the browsing modes support search in conjunction with keywords.

objects in the video with simply segmented backgrounds.[1] It will not work well if the user is interested in video sequences that are simple to describe, but are hard to sketch out, for example, a video shot of a group of soldiers marching, shots of a crowd on the beach, etc. It will also not work well when the user is interested in a particular semantic class of shots: he might be interested in retrieving that news segment containing the anchor person, when the news anchor is talking about Bosnia. For these examples, it is far more useful to search in conjunction with keywords.

[1] Note that, even if the background shows a crowd, due to aggressive region merging, they may be merged into one single region.

III. The VideoQ System Overview

VideoQ is a Web-based video search system (Fig. 3), where the user queries the system using animated sketches. The system, which resides on the Web, incorporates a client–server architecture. The client (a java applet) is loaded up into a Web browser where the user formulates (sketches) a query scene as a collection of objects with different attributes. Attributes include motion, spatiotemporal ordering, shape, and the more familiar attributes of color and texture.

The query server contains several feature databases, one for each of the individual features that the system indexes on. Since we index on motion, shape, as well as color and texture,

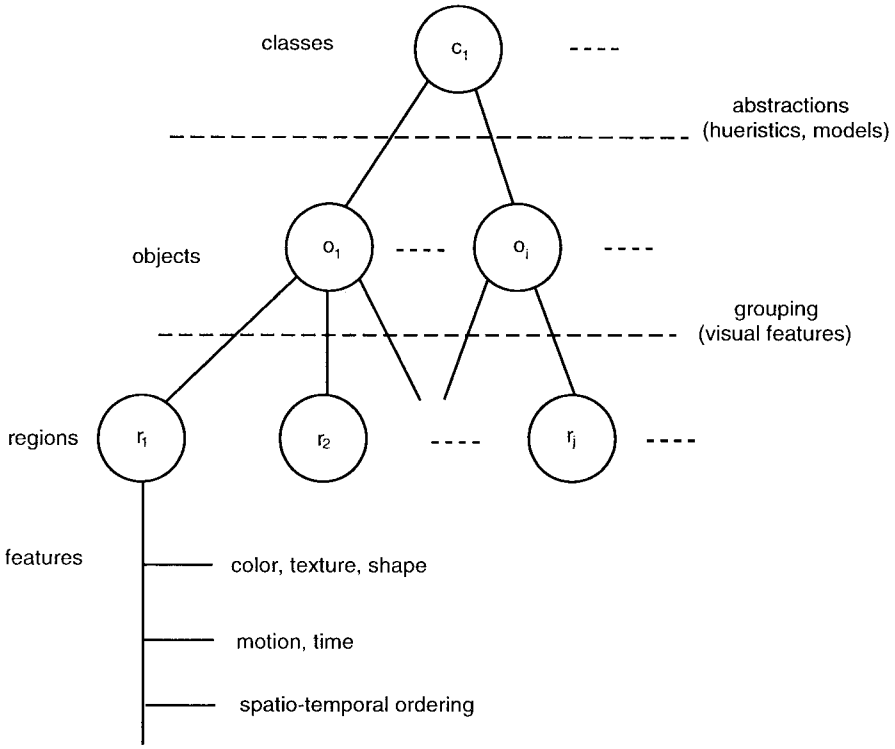

Fig. 4. Feature classification tree.

we have databases for each of these features. The source video shot database is stored as a compressed MPEG stream.

Once the user is done formulating the query, the client sends it over the network to the query server. There, the features of each object specified in the query are matched against the features of the objects in the database. Then, lists of candidate video shots are generated for each object specified in the query. The candidate lists for each object are then merged to form a single video shot list. Now, for each of these video shots in the merged list, key frames are dynamically extracted from the video shot database and returned to the client over the network. The matched objects are highlighted in the returned key frame.

The user can interactively view these matched video shots over the network by simply clicking on the key frame. Then, in the back end, the video shot corresponding to that key frame is extracted in real time from the video database by "cutting" out that video shot from the database. The video shots are extracted from the video database using basic video editing schemes [22] in the compressed domain. The user needs an MPEG player in order to view the returned video stream.

Since the query as formulated by the user in the VideoQ system comprises a collection of objects having spatiotemporal attributes, we need to formalize the definition of a video object.

IV. What Is a Video Object?

We define a region to be a contiguous set of pixels that is homogeneous in the features that we are interested in (i.e., texture, color, motion, and shape). A video object is defined as a collection of video regions which have been grouped together under some criteria across several frames. Namely, a video object is a collection of regions exhibiting consistency[2] across several frames in at least one feature. For example, a shot of a person (the person is the "object" here) walking would be segmented into a collection of adjoining regions differing in criteria such as shape, color, and texture, but all the regions may exhibit consistency in their motion attribute. As shown in Fig. 4, the objects themselves may be grouped into higher semantic classes.

The grouping problem of regions is an area of ongoing research, and for the purposes of this paper, we restrict our attention to regions only. Regions may be assigned several attributes, such as color, texture, shape, and motion.

A. Color, Texture, Shape

In the query interface of VideoQ, the set of allowable colors is obtained by uniformly quantizing the HSV color space. The Brodatz texture set is used for assigning the textural attributes to the various objects. The shape of the video object can be an arbitrary polygon along with ovals of arbitrary shape and size. The visual palette allows the user to sketch out an arbitrary polygon with the help of the cursor, other well-known shapes such as circles, ellipses, and rectangles are predefined, and are easily inserted and manipulated.

B. Motion, Time

Motion is the *key* object attribute in VideoQ. The interface allows the user to specify an arbitrary polygonal trajectory for the query object. The temporal attribute defines the overall

[2] If two regions exhibit consistency in all features, then they will be merged into one region. Regions which exhibit *no* consistency at all in any feature would probably not belong to the same object.

duration of the object, which can either be intuitive (long, medium, or short) or absolute (in seconds).

Since VideoQ allows users to frame multiple object queries, the user has the flexibility of specifying the overall scene temporal order by specifying the "arrival" order of the various objects in the scene. The death order (or the order in which they disappear from the video) depends on the duration of each object.

Another attribute related to time is the scaling[3] factor, or the rate at which the size of the object changes over the duration of the objects existence. Additional global scene attributes include the specification of the (perceived) camera motion like panning or zooming.

C. Weighting the Attributes

Prior to the actual query, the various features need to be weighted in order to reflect their relative importance in the query (refer to Fig. 1). The feature weighting is global to the entire animated sketch; for example, the attribute color, will have the same weight across all objects. The final ranking of the video shots that are returned by the system is affected by the weights that the user has assigned to various attributes.

V. Automatic Video Shot Analysis

The entire video database is processed off line. The individual videos are decomposed into separate shots, and then within each shot, video objects are tracked across frames.

A. Scene Cut Detection

Prior to any video object analysis, the video must be split up into "chunks" or video shots. Video shot separation is achieved by scene change detection. Scene changes are either abrupt scene changes or transitional (e.g., dissolve, fade in/out, wipe). Meng [21] describes an efficient scene change detection algorithm that operates on compressed MPEG streams.

It uses the motion vectors and discrete cosine transform coefficients from the MPEG stream to compute statistical measures. These measurements are then used to verify the heuristic models of abrupt or transitional scene changes. For example, when a scene change occurs before a B frame in the MPEG stream, most of the motion vectors in that frame will point to future reference frame. The real-time algorithm operates directly on the compressed MPEG stream, without complete decoding.

B. Global Video Shot Attributes

The global motion (i.e., background motion) of the dominant background scene is automatically estimated using the six-parameter affine model [30]. A hierarchical pixel-domain motion estimation method [3] is used to extract the optical flow. The affine model of the global motion is used to compensate the global motion component of all objects in the scene.[4] The six-parameter model

$$\Delta x = a_o + a_1 x + a_2 y \tag{1}$$

$$\Delta y = a_3 + a_4 x + a_5 y \tag{2}$$

where a_i are the affine parameters, x, y are the pixel coordinates, and Δx, Δy are the pixel displacements at each pixel.

Classification of global camera motion into modes such as zooming or panning is based on the global affine estimation. In order to detect panning, a global motion velocity histogram is computed along eight directions. If there is dominant motion along a particular direction, then the shot is labeled as a panning shot along that direction.

In order to detect zooming, we need to first check if the average magnitude of the global motion velocity field and two affine model scaling parameters (a_1 and a_5) satisfy certain threshold criteria. When there is sufficient motion, and a_1 and a_5 are both positive, then the shot is labeled as a "zoom-in" shot and if they are both negative, then the shot is labeled as a "zoom out."

C. A Brief Review of Video Object Segmentation

Common video object segmentation and tracking are techniques based on selective features such as motion, color, and edges as well as the consistency of their properties over space and time. In [12], morphological segmentation algorithms are used for intraframe and interframe segmentation of coherent motion regions from successive motion vector fields. To obtain accurate motion boundary, color-based spatial segmentations are used to refine the motion segmentation results. In [36], moving images are decomposed into sets of overlapping layers using block-based affine motion analysis and k-means clustering algorithm. Each layer corresponds to the motion, shape, and intensity of a moving region. Due to the complexity of object motion in general videos (e.g., a moving object may stop), these pure motion-based algorithms cannot be used to automatically segment and track regions through image sequences.

In the field of image segmentation, color (or gray level) and edges are two major features being widely used. As both of them have limitations, fusion of color and edge information is proposed in [29] to obtain more "meaningful regions." Here, the color segmentation results are further split and merged in order to ensure consistency with the edge map.

D. Tracking Objects: Motion, Color, and Edges

Our algorithm for segmentation and tracking of image regions based on the fusion of color, edge, and motion information in the video shot. The basic region segmentation and tracking procedure is shown in Fig. 5. The projection and segmentation module is the module where different features are fused for region segmentation and tracking.

Color is chosen as the major segmentation feature because of its consistency under varying conditions. As boundaries of

[3] This is the factor by which an object changes its size over its duration on the shot. This change can either be induced by camera motion or by the objects intrinsic motion.

[4] Global motion compensation is not needed if users prefer to search videos based on perceived motion.

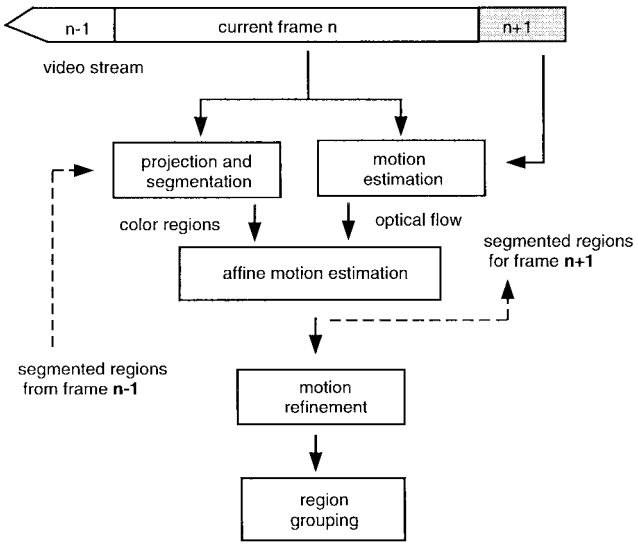

Fig. 5. Region segmentation and tracking of frame n.

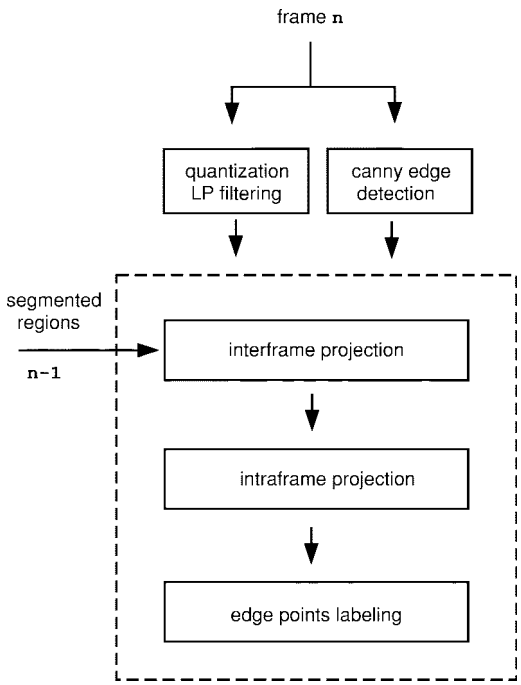

Fig. 6. Region projection and segmentation of frame n.

color regions may not be accurate due to noise, each frame of the video shot is filtered before color region merging is done. Edge information is also incorporated into the segmentation process to improve the accuracy. Optical flow is utilized to project and track color regions through a video sequence.

The optical flow of current frame n is derived from frame n and $n+1$ in the motion estimation module using a hierarchical block matching method [3]. Given color regions and optical flow generated from above two processes, a linear regression algorithm is used to estimate the affine motion for each region. Now, color regions with affine motion parameters are generated for frame n, which will the be tracked in the segmentation process of frame $n+1$. The projection and segmentation module is discussed in greater detail in [38]. Fig. 6 presents a brief overview.

Fig. 7 shows segmentation results with two sequences. In both cases, the top row shows the original sequence and the second row shows a subset of automatically segmented regions being tracked. Tracked regions are shown with their representative (i.e., average) colors. Experiments show that our algorithm is robust for the tracking of salient color regions under different circumstances, such as multiple objects, fast or slow motion, and instances of regions being covered and uncovered.

VI. Building the Visual Feature Library

Once each object in the video shot has been segmented and tracked, we then compute the different features of the object and store them in our feature library. For each object, we store the following features.

Color: The representative color in the quantized CIE-LUV space. It is important to bear in mind that the quantization is not static, and the quantization palette changes with each video shot. The quantization is calculated anew for each sequence with the help of a self-organizing map.

Texture: Three Tamura [35] texture measures, coarseness, contrast, and orientation, are computed as a measure of the textural content of the object.

Motion: The motion of the video object is stored as a list of $N-1$ vectors (where the number of frames in the video is N). Each vector is the average translation of the centroid of the object between successive frames[5] after global motion compensation [30]. Along with this information, we also store the frame rate of the video shot sequence, hence establishing the "speed" of the object as well as its duration.

Shape: For each object, we first determine the principal components of the shape by doing a simple eigenvalue analysis [28]. At the same time, we generate first- and second-order moments of the region. Two other new features, the normalized area,[6] and the percentage area[7] are calculated. We then determine if the region can be well approximated by an ellipse, and label it so if that is indeed the case. We chose not to store the best fit polygon to the object because of reasons of computational complexity. The computational complexity of matching two arbitrary N vertex polygons is $O(N^2, \log N)$ [1].

The resulting library is a simple database having an $\{$attribute, value$\}$ pair for each object. Creating a relational database will obviously allow for more complex queries to be performed over the system as well as decrease the overall search time. The issue of the structure of the database is an important one, but was not a priority in the current implementation of VideoQ.

[5] We could have also stored the successive affine transformations, but that would have increased the complexity of the search. Also, it is worth keeping in mind that the users will not have an "exact" idea of the trajectory of the object that they wish to retrieve.

[6] The ratio of the area of the object to the area of the circumscribing circle. Note that this feature is invariant to scale.

[7] This is the percentage of the area of the video shot that is occupied by the object.

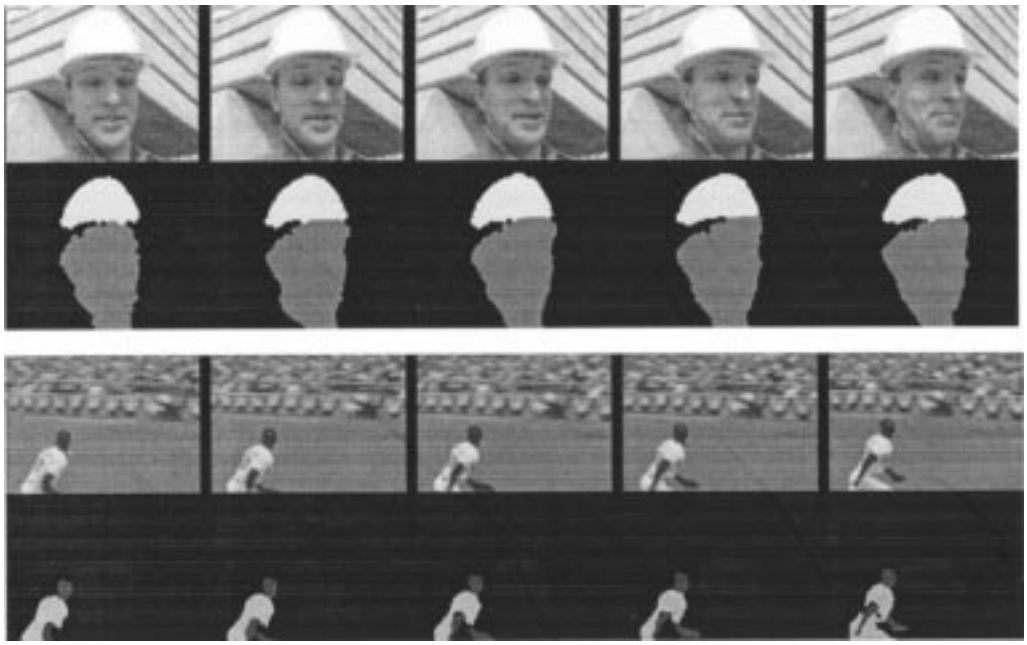

Fig. 7. Region segmentation on QCIF sequences, using feature fusion. The top rows show the original sequence, while the corresponding bottom rows show the segmented regions.

VII. Feature Space Metrics

The nature of the metric plays a key role in any image or video retrieval system. Designing good metrics is a challenging problem as it often involves a tradeoff between the computational complexity of the metric and the quality of the match. It is not enough to be able to locate images or videos that are close under a metric; they must be perceptually close to the query.

While we employ well-accepted metrics for color, texture, and shape, we have designed new metrics to exploit the spatiotemporal information in the video.

A. Matching Motion Trails

A motion trail is defined to be the three-dimensional (3-D) trajectory of a video object. It is represented by a sequence $\{x[i], y[i]\}$, $i \in \{1, \cdots, N\}$, the three dimensions comprising the two spatial dimensions x, y and the temporal dimension t (normalized to the frame number; the frame rate provides us with the true time information). Prior techniques to match motion [9], have used simple chain codes or a B spline to represent the trajectory, without completely capturing the spatiotemporal characteristic of the motion trail.

The user sketches out the trajectory as a sequence of vertices in the x–y plane. In order for him to specify the motion trail completely, he must specify the duration of the object in the video shot. The duration is quantized (in terms of the frame rate)[8] into three levels: long, medium, and short. We compute the entire trail by uniformly sampling the motion trajectory based on the frame rate.

We develop two major modes of matching trails.

Spatial: In the spatial mode, we simply project the motion trail onto the x–y plane. This projection results in an ordered contour. The metric then measures distances between the query contour and the corresponding contour for each object in the database. This kind of matching provides a "time-scale invariance." This is useful when the user is unsure of the time taken by an object to execute the trajectory.[9]

Spatiotemporal: In the spatiotemporal mode, we simply use the entire motion trail to compute the distance. We use the following distance metric:

$$\sum_i \left((x_q[i] - x_t[i])^2 + (y_q[i] - y_t[i])^2\right) \tag{3}$$

where the subscripts q and t refer to the query and the target trajectories, respectively, and the index i runs over the frame numbers.[10] Since, in general, the duration of the query object will differ from that of the objects in the database, there are some further refinements possible.

- When the durations differ, we could simply match the two trajectories up to the shorter of the two durations (i.e., the index i runs up to $\min(query\,duration, database\,object\,duration)$ and ignore the "tail").
- We could also normalize the two durations to a canonical duration, and then perform the match.

B. Matching Other Features

Let us briefly describe the distance metrics used in computing the distances in the other feature spaces.

[8] We quantify it in terms of (frame rate)/(unit distance) where the distance refers to the length of the motion trajectory in pixels. We assume a canonical frame rate of 30 frames/s.

[9] An immediate benefit of using this method is when one is matching against a database of sports shots, then "slow-motion" replays as well as "normal-speed" shots will be retrieved as they both execute the same xy contour.

[10] Alternately, the index could run over the set of subsampled points.

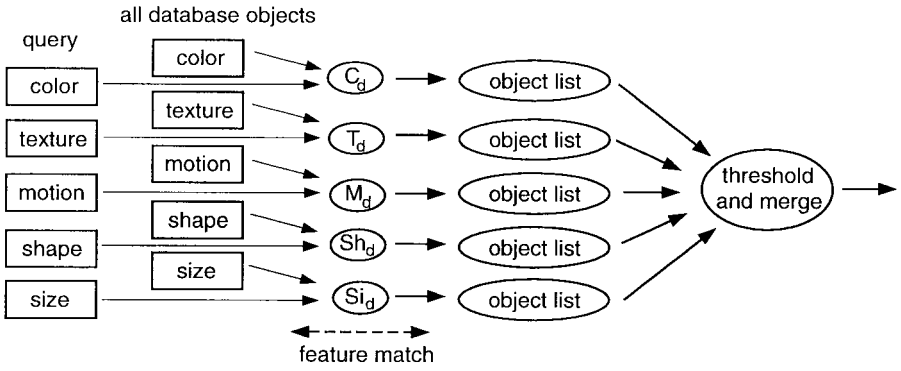

Fig. 8. Generating the candidate video shot list for a single-object query. The first column of features shows the features of the query, while the second column shows the features across all objects in the database.

Color: The color of the query object is matched with the mean color of a candidate tracked object in the database as follows:

$$C_d = \sqrt{(L_q - L_t)^2 + 4(U_q - U_t)^2 + 4(V_q - V_t)^2} \quad (4)$$

where C_d is the weighted Euclidean color distance in the CIE-LUV space, and the subscripts q and t refer to the query and the target, respectively.

Texture: In our system, we compute three Tamura [35] texture parameters (coarseness, contrast, and orientation) for each tracked object. The distance metric is simply the Euclidean distance weighted along each texture feature with the variances along each channel:

$$T_d = \sqrt{\frac{(\alpha_q - \alpha_t)^2}{\sigma_\alpha^2} + \frac{(\beta_q - \beta_t)^2}{\sigma_\beta^2} + \frac{(\phi_q - \phi_t)^2}{\sigma_\phi^2}} \quad (5)$$

where α, β, and ϕ refer to the coarseness, contrast, and orientation, respectively, and the various $\sigma_{\alpha,\beta,\phi}$ refer to the variances in the corresponding features.

Shape: In the current implementation, the metric only involves the principal components of the shape:

$$Sh_d = \left| \frac{\lambda_{2_q}}{\lambda_{1_q}} - \frac{\lambda_{2_t}}{\lambda_{1_t}} \right| \quad (6)$$

where λ_2 and λ_1 are the eigenvalues along the principal axes of the object (their ratio is the aspect ratio).

Size: This is simply implemented as a distance on the area ratio[11]

$$Si_d = 1 - \frac{\min(A_q, A_t)}{\max(A_q, A_t)} \quad (7)$$

where $A_{q,t}$ refer to the percentage areas of the query and target, respectively.

The total distance is simply the weighted sum of these distances, after the dynamic range of each metric has been normalized to lie in [0, 1], i.e.,

$$D_g = \sum_{i \in \{\text{features}\}} \omega_i D_i \quad (8)$$

where ω_i is the weight assigned to the particular feature and D_i is the distance in that feature space.

[11] This is the area of the object divided by the area of the entire shot.

VIII. Query Resolution

Using these feature space metrics and the composite distance function, we compute the composite distance of each object in the database with each object in the query. Let us now examine how we generate candidate video shots, given a single and multiple objects as queries. An example of single-object query along with the results (the candidate result) is shown in Fig. 1.

A. Single-Object Query

The search along each feature of the video object produces a candidate list of matched objects and the associated video shots. Each candidate list can be merged by a rank threshold or a feature distance threshold. Then, we merge the candidate lists, keeping only those that appear on the candidate list for each feature. Next, we compute the global weighted distance D_g, and then sort the merged list based on this distance. A global threshold is computed (based on the individual thresholds and additionally modified by the weights) which is then used to prune the object list. This is schematically shown is Fig. 8. Since there is a video shot associated with each of the objects in the list, we return the key frames of the corresponding video shots to the user.

B. Querying Multiple Objects

When the query contains multiple video objects, we need to merge the results of the individual video object queries. The final result is simply a logical intersection of all of the results of the individual query objects. When we perform a multiple-object query in the present implementation, we do not use the relative ordering of the video objects in space as well in time. These additional constraints could be imposed on the result by using the idea of 2-D strings [8], [32], [33] (discussed in Section X-C).

IX. How Does VideoQ Perform?

Evaluating the performance of video retrieval systems is still very much an open research issue [7]. There does not exist a standard video test set to measure retrieval performance or standard benchmarks to measure system performance. This is partly due to the emerging status of this field. To evaluate

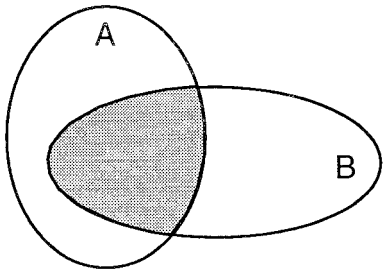

Fig. 9. Set A is the set that is obtained from the search, while set B refers to the ground truth. Precision and recall are defined using these two sets.

VideoQ, we use two different approaches. First, we extend the standard precision-recall metrics in information retrieval. Although we acknowledge several drawbacks of this classical metric, we include it here simply as a reference. Another type of metric measures the effort and cost required to locate a particular video clip that a user has in mind or one that the user may have previously browsed in the database.

A. Precision-Recall Type Metrics

In our experimental setup, we have a collection of 200 video shots, categorized into sports, science, nature, and history. By applying object segmentation and tracking algorithms to the video shots, we generated a database of more than 2000 salient video objects and their related visual features.

To evaluate our system, precision-recall metrics are computed. Formally, precision and recall are defined as follows:

$$\text{recall} = \frac{\text{retrieved and relevant}}{\text{all relevant in the database}} \tag{9}$$

$$\text{precision} = \frac{\text{retrieved and relevant}}{\text{number retrieved}} \tag{10}$$

where the relevant video shots are predefined by the ground truth database. Using Fig. 9

$$\text{recall} = \frac{|A \cap B|}{|B|} \tag{11}$$

$$\text{precision} = \frac{|A \cap B|}{|A|} \tag{12}$$

where the $|\ |$ operator returns the size of the set.

Before each sample query, the user establishes a ground truth by manually choosing a set of relevant or "desired" video shots from the database. For each sample query shown in Fig. 10, a ground truth is established by choosing all of the relevant video shots in the database that have corresponding features. The sample query returns a list of candidate video shots, and precision-recall values are calculated according to (9) and (10). The precision-recall curve is generated by increasing the size of the return list and computing the corresponding precision-recall values. Clearly, the precision-recall curve is a parametric function of the set size.

While the VideoQ system comprises many parts such as scene cut detection, object segmentation and tracking, feature selection, and matching, precision-recall metrics measure how well the system performs as a whole. Performance is based solely on the proximity of the returned results with the ground truth.

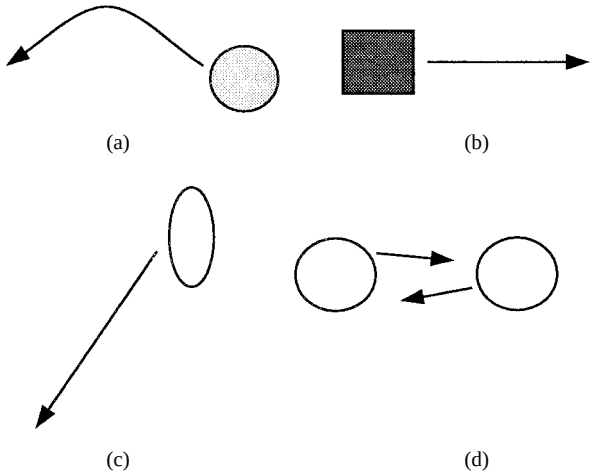

Fig. 10. Four sample queries used in the precision-recall experiments: (a) and (b) highlight motion, color, and size, (c) highlights motion and size, and (d) highlights multiple objects in addition to motion and size.

Four sample queries were performed as shown in Fig. 10. The first sample query specifies a skin-colored medium-sized object that follows a motion trajectory arcing to the left. The ground truth consisted of nine video shots of various high jumpers in action and brown horses in full gallop. The return size is increased from 1–20 video shots, and a precision-recall curve is plotted in Fig. 11(a).

An overlay of the four precision-recall curves is plotted in Fig. 11. For normal systems, the precision-recall curve remains relatively flat up to a certain recall value, after which the curve slopes downward. This can be readily visualized using Fig. 9. The precision-recall curves of Fig. 11 are averaged to yield the expected system performance (Fig. 12). The "knee" of this curve determines the optimal size of the returned set. A look at Fig. 12 indicates that the optimal size for these queries should lie between six–eight shots.

B. Time and Cost to Find a Particular Video Shot

Two benchmarks are used to evaluate how efficiently the system uses its resources to find the correct video shot: query frequency and bandwidth. While the former measures the number of separate queries needed to get a particular video shot in the return list, the latter measures the number of different false alarms that are returned before obtaining the correct video shot in the return list.

A randomly generated target video shot, shown in Fig. 13(b), is chosen from the database. In order to find this video shot, we query the system, selecting a combination of objects, features, and feature weights. The total number of queries to get this video shot is recorded. By varying the size of the return list, we generate the query frequency curve.

Each query returns a list of video shots from an HP 9000 server over the network to a client. The video shot is represented by a key frame, an 88×72 pixel image. In many cases, a series of queries was needed to reach a particular video shot. The number of key frames that were returned are totaled. Repeat frames are subtracted from this total since they are stored in the cache and not retransmitted over the network.

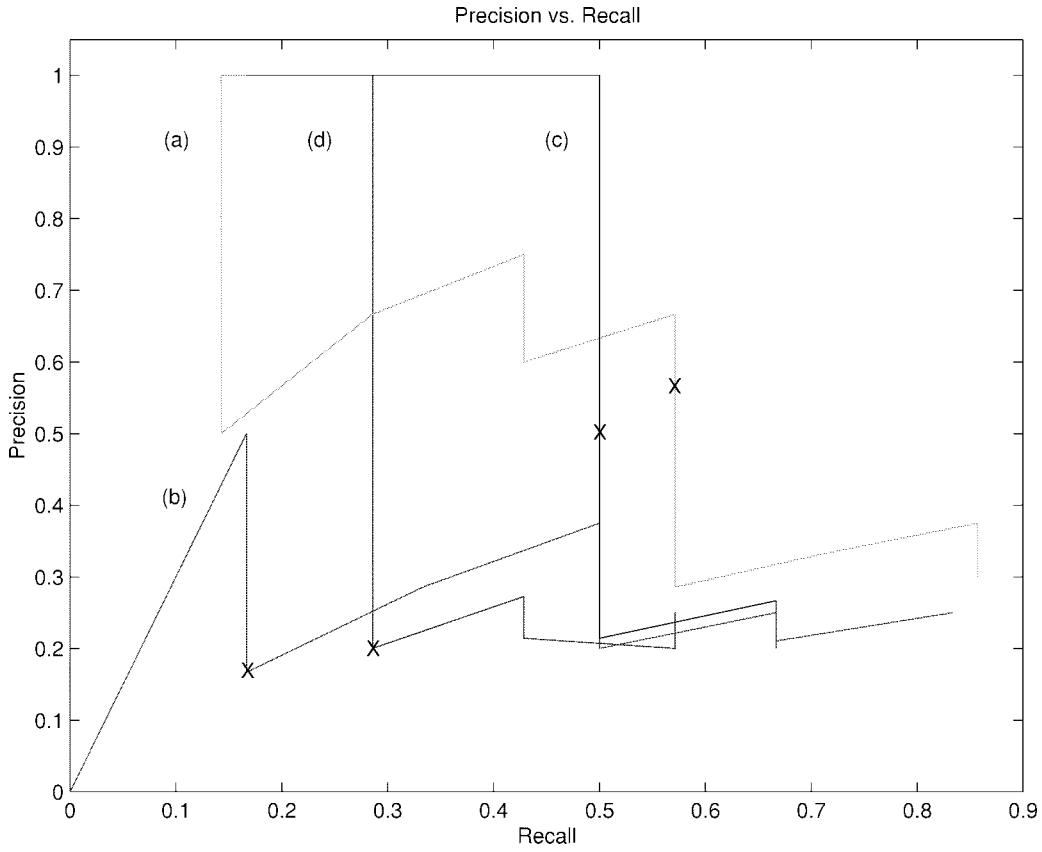

Fig. 11. Precision-recall curves corresponding to the sample queries of Fig. 10.

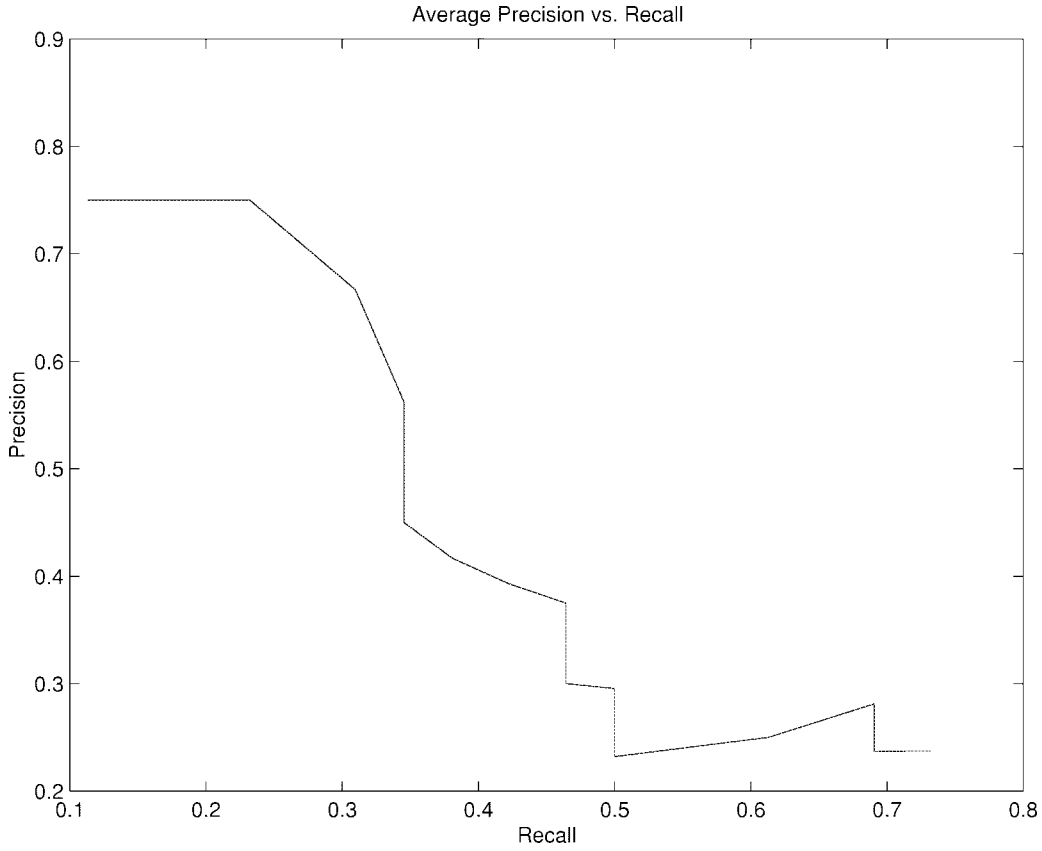

Fig. 12. Precision-recall curve averaged over the four precision-recall curves of Fig. 11.

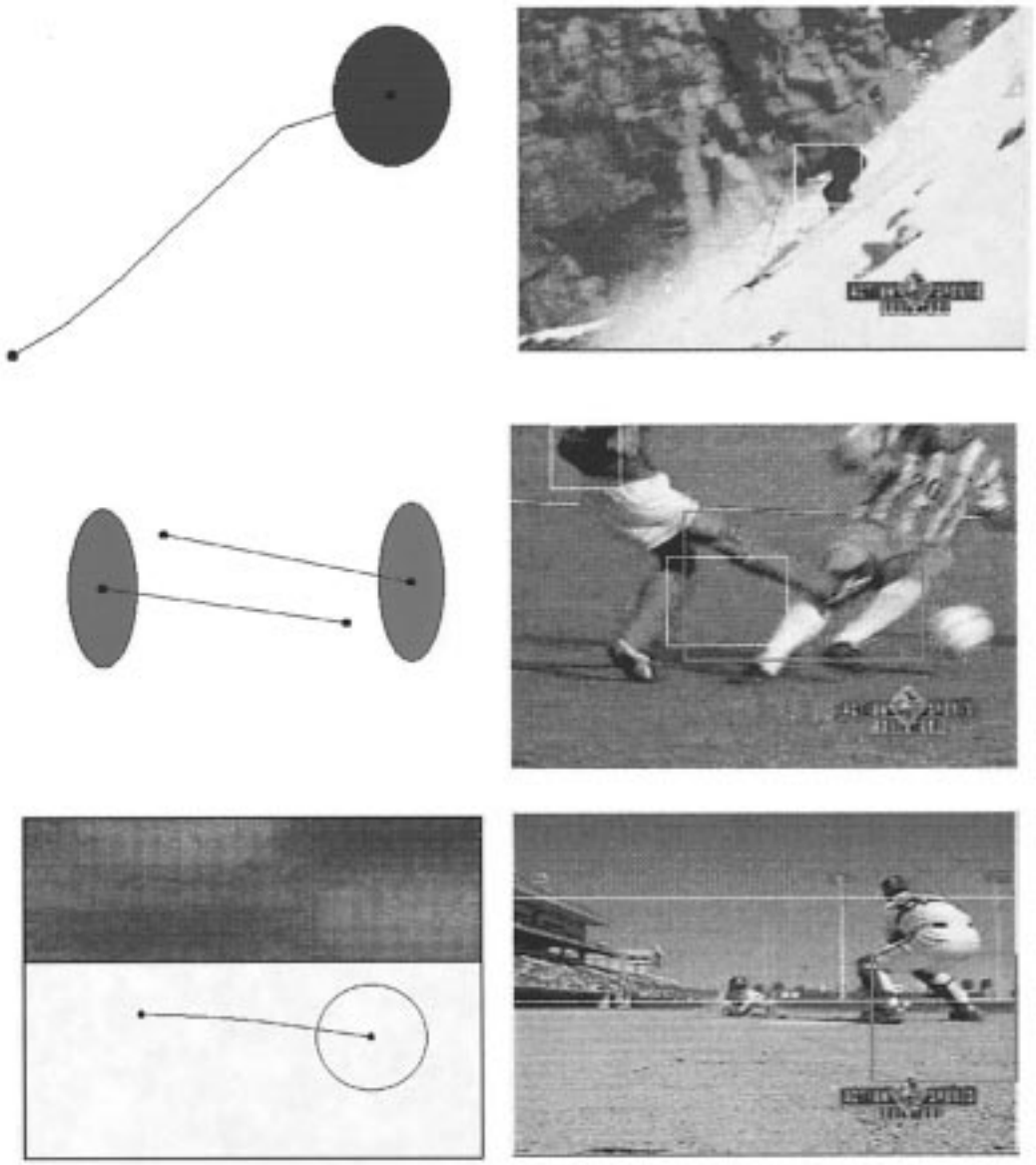

Fig. 13. Three sample queries used in system benchmarks. The left column shows the final sketch to successfully retrieve the video: (a) a skier, (b) two soccer players, and (c) a baseball query. In the baseball video clip, the catcher moves to the left. Also note the strong match of the sky-like texture to the sky. Video courtesy of Hot Shots Cool Cuts Inc. and Action Adventure Inc.

Conceptually, bandwidth is proportional to the total number of key frames transmitted. Therefore, the bandwidth is recorded, and by varying the size of the return list, the bandwidth curve is generated.

Twenty target video shots, similar to those in Fig. 13, are randomly selected. For each target video shot, sample queries are performed, and query frequency and bandwidth curves are generated by varying the return size from 3 to 18 video shots. The query frequency curve in Fig. 14 shows that a greater number of queries is needed for small return sizes. On average, for a return size of 14, only two queries are needed to reach the desired video shot.

Fig. 15 shows how the average bandwidth varies with increasing set size. The figure also shows an optimal "dip" in the curve, indicating the presence of an optimal return set size. This is observed to be around nine shots. For small return sizes, many times ten or more queries failed to place the video shot within the return list. In Fig. 15, we compensate for these failed queries by applying a heuristic to penalize the returned videos.

It was observed that the system performed better when it was provided with more information. Multiple-object queries proved more effective than single-object queries. Also, objects with a greater number of features, such as color, motion, size, and shape, performed better than those with just a few features. It is also important to emphasize that certain features proved more effective than others. For example, motion was the most effective, followed by color, size, shape, and texture.

X. Research Issues in VideoQ

While the results section (Section IX) demonstrates that VideoQ works well, there are other issues that need to be

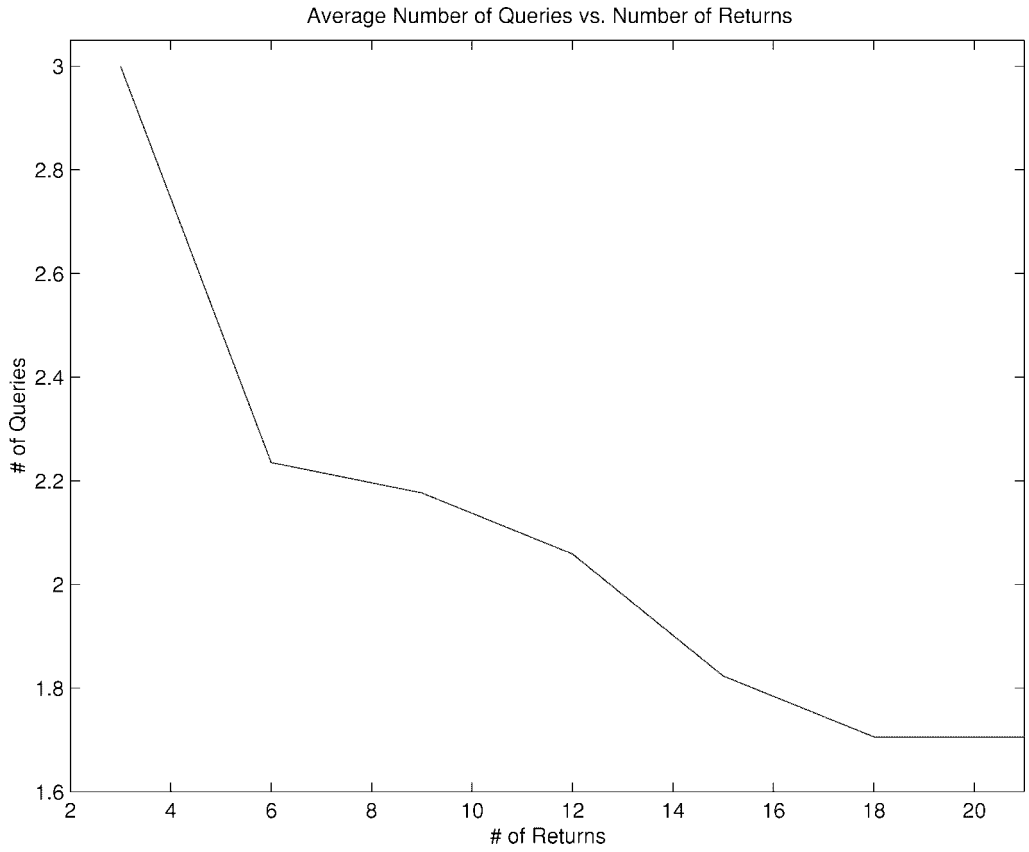

Fig. 14. Average number of queries needed to reach a video shot.

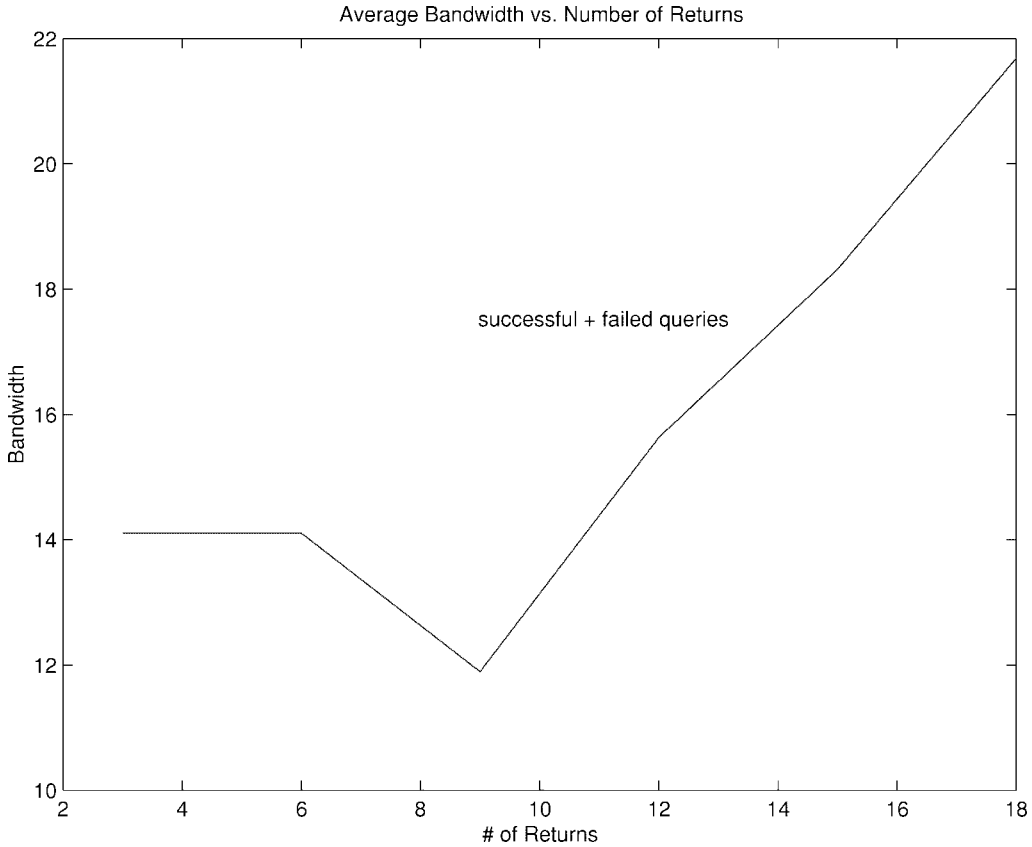

Fig. 15. Average bandwidth used.

addressed. This section contains a brief overview of the issues that we are currently working on.

A. Region Grouping

Automatic region grouping is an open problem in computer vision, and in spite of decades of research, we are still far from a completely automated technique that works well on unconstrained data. Nevertheless, the segmented results need to be further grouped in order for us to prune the search as well search at a higher semantic level. Also, good region grouping is needed to avoid oversegmentation of the video shot.

B. Shape

One of the biggest challenges with using shape as a feature is to be able to represent the object while retaining a computationally efficient metric to compare two shapes. The complexity of matching two arbitrary N point polygons is $O(N^2 \log N)$ [1].

One approach is to use geometric invariants to represent shape [26], [19], [20]. These are invariants on the coefficients of the implicit polynomial used to represent the shape of the object. However, these coefficients need to be very accurately calculated as the representation (that of implicit polynomials) is very sensitive to perturbations. Additionally, generating these coefficients is a computationally intensive task.

C. Spatiotemporal Search

We are currently extending the work done on VisualSEEk [33] on two-dimensional (2-D) strings [8] in order to effectively constrain the query results. There has been work using modified 2-D strings as a spatial index into videos [2], [32].

For video, 2-D strings can be extended to a sequence of 2-D strings or a 2-D string followed by a sequence of change edits [32]. Building on these observations, we propose two efficient methods for indexing spatiotemporal structures of segmented video objects.

- In the first method, only frames with significant changes of spatial structures need to be explicitly indexed (by 2-D strings of those image frames). Given such a representation, users will be able to search video objects or events of interest (e.g., two objects swap locations, birth, or death of objects) by specifying temporal instances or changes of spatial structures. A simplified representation is to include the 2-D strings at the beginning frame, the ending frame, and several sampled frames in between.
- The second method extends the 2-D string-based query to 3-D strings. Video objects may be projected to x, y and time dimensions to index their absolute centroid position, three-dimensional support, and relative relationships. More sophisticated variations of 3-D strings can be used to handle complex relationships such as adjacency, containment, and overlap.

XI. Conclusions

Video search in large archives is an emerging research area. Although integration of the diverse multimedia components is essential in achieving a fully functional system, we focus on exploiting visual cues in this paper. Using the visual paradigm, our experiments with VideoQ show considerable success in retrieving diverse video clips such as soccer players, high jumpers, and skiers. Indexing video objects with motion attributes and developing good spatiotemporal metrics have been the key issues in this paradigm.

The other interesting and unique contributions include developing a fully automated video analysis algorithm for object segmentation and feature extraction, a java-based interactive query interface for specifying multiobject queries, and the content-based visual matching of spatiotemporal attributes.

Extensive content analysis is used to obtain accurate video object information. Global motion of the background scene is estimated to classify the video shots as well as to obtain the local object motion. A comprehensive visual feature library is built to incorporate most useful visual features such as color, texture, shape, size, and motion. To support the on-line Web implementation, our prior results in compressed-domain video shot segmentation and editing are used. Matched video clips are dynamically "cut" out from the MPEG stream containing the clip without full decoding of the whole stream.

As described earlier, our current work includes region grouping, object classification, more accurate shape representation, and support of relative spatiotemporal relationships. An orthogonal direction addresses the integration of the video object library with the natural language features to fill the gap between low-level visual domain and the high-level semantic classes.

References

[1] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kadem, and J. S. B. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 209–216, Mar. 1991.

[2] T. Arndt and S. K. Chang, "Image sequence compression by iconic indexing," in *Proc. IEEE Workshop Visual Languages*, IEEE Computer Society, 1989, pp. 177–182.

[3] M. Bierling, "Displacement estimation by hierarchical block matching," *SPIE Visual Commun. Image Processing*, vol. 1001, 1988.

[4] A. Del Bimbo and P. Pala, "Visual image retrieval by elastic matching of user sketches," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 2, pp. 121–132, Feb. 1997.

[5] G. D. Borshukov, G. Bozdagi, Y. Altunbasak, and A. M. Tekalp, "Motion segmentation by multi-stage affine classification," *IEEE Trans. Image Processing*, to be published.

[6] M. G. Brown, J. T. Foote, G. J. F. Jones, K. S. Jones, and S. J. Young, "Open-vocabulary speech indexing for voice and video mail retrieval," presented at the ACM Multimedia Conf., Boston, Nov. 1996.

[7] S. F. Chang, J. R. Smith, H. J. Meng, H. Wang, and D. Zhong, "Finding images/video in large archives," *CNRI Digital Library Mag.*, Feb. 1997.

[8] S. K. Chang, Q. Y. Shi, and S. Y. Yan, "Iconic indexing by 2-D strings," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 413–428, May 1987.

[9] N. Dimitrova and F. Golshani, "$\mathcal{R}_\chi$ for semantic video database retrieval," in *ACM Multimedia Conf.*, San Francisco, CA, Oct. 1994, pp. 219–226.

[10] C. Faloutsos, M. Flickner, W. Niblack, D. Petkovic, W. Equitz, and R. Barber, "Efficient and effective querying by image content," Res. Rep. RJ 9203 (81511), IBM Almaden Res. Cen., San Jose, CA, Aug. 1993.

[11] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Comput. Mag.*, vol. 28, pp. 23–32, Sept. 1995.

[12] C. Gu, T. Ebrahimi, and M. Kunt, "Morphological moving object segmentation and tracking for content-based video coding," in *Multimedia Communication and Video Coding.* New York: Plenum, 1996.

[13] A. Gupta and R. Jain, "Visual information retrieval," *Commun. ACM*, vol. 40, pp. 70–79, May 1997.
[14] A. Hamrapur, A. Gupta, B. Horowitz, C. F. Shu, C. Fuller, J. Bach, M. Gorkani, and R. Jain, "Virage video engine," in *SPIE Proc. Storage and Retrieval for Image and Video Databases V*, San Jose, CA Feb. 1997, pp. 188–197.
[15] A. G. Hauptmann and M. Smith, "Text, speech and vision for video segmentation: The informedia project," presented at the AAAI Fall Symp., Computational Models for Integrating Language and Vision, Boston, MA, Nov. 1995.
[16] K. Hirata and T. Kato, "Query by visual example, content based image retrieval," in *Advances in Database Technology—EDBT'92*, A. Pirotte, C. Delobel, and G. Gottlob, Eds., Lecture Notes in Computer Science, vol. 580.
[17] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast miltiresolution image querying," in *Proc. SIGGRAPH*, Los Angeles, CA, Aug. 1995, pp. 277–286.
[18] K. S. Jones, *Information Retrieval Experiment.* Boston: Butterworth, 1981.
[19] D. Karen, D. Cooper, and J. Subrahmonia, "Describing complicated objects by implicit polynomials," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 38–53, Jan. 1994.
[20] Z. Lei, D. Karen, and D. Cooper, "Computationally fast Bayesian recognition of complex objects based on mutual algebraic invariants," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, Oct. 1995, pp. 635–638.
[21] J. Meng, Y. Juan, and S. F. Chang, "Scene change detection in a MPEG compressed video sequence," in *SPIE Symp. Electron. Imaging: Science and Technol.—Digital Video Compression: Algorithms and Technol.*, vol. 2419, San Jose, CA, Feb. 1995.
[22] J. Meng and S. F. Chang, "CVEPS: A compressed video editing and parsing system," presented at the ACM Multimedia Conference, Boston, MA, Nov. 1996, [Online]. Available WWW: http://www.ctr.columbia.edu/webclip.
[23] T. Minka, "An image database browser that learns from user interaction," M.I.T. Media Lab. Perceptual Computing Section, TR 365, 1996.
[24] R. Mohan, "Text based search of TV news stories," presented at SPIE Photonics East, Int. Conf. Digital Image Storage and Archiving Syst., Boston, MA, Nov. 1996.
[25] "Description of MPEG-4," ISO/IEC JTC1/SC29/WG11 N1410, MPEG Doc. N1410, Oct. 1996.
[26] J. L. Mundy and A. Zisserman, Eds., *Geometric Invariance In Computer Vision.* Cambridge, MA: M.I.T. Press, 1992.
[27] A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Content-based manipulation of image databases," *Int. J. Comput. Vision*, vol. 18, no. 3, pp. 233–254, 1996.
[28] E. Saber and A. M. Tekalp, "Region-based affine shape matching for automatic image annotation and query-by-example," *Visual Comm. and Image Representation*, to be published.
[29] E. Saber, A. M. Tekalp, and G. Bozdagi, "Fusion of color and edge information for improved segmentation and linking," *Image and Vision Computing*, to be published.
[30] H. S. Sawhney, S. Ayer, and M. Gorkani, "Model-based 2D & 3D dominant motion estimation for mosaicing and video representation," in *Int. Conf. Comput. Vision*, Boston, MA, June 1995, pp. 583–590.
[31] B. Shahraray and D. C. Gibbon, "Automatic generation of pictorial transcript of video programs," *SPIE*, vol. 2417, pp. 512–518, 1995.
[32] K. Shearer, S. Venkatesh, and D. Kieronska, "Spatial indexing for video databases," *J. Visual Commun. Image Representation*, vol. 8, Sept. 1997.
[33] J. R. Smith and S. F. Chang, "VisualSEEk: A fully automated content-based image query system," in *ACM Multimedia Conf.*, Boston, MA, Nov. 1996, pp. 87–98. [Online]. Available WWW: http://www.ctr.columbia.edu/VisualSEEk.
[34] S. W. Smoliar and H. J. Zhang, "Content-based video indexing and retrieval," *IEEE Multimedia Mag.*, Summer 1994.
[35] H. Tamura and S. Mori, "Textural features corresponding to visual perception," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, June 1978.
[36] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," M.I.T. Media Lab. Perceptual Computing Sect., TR 279.
[37] M. M. Yeung and B. L. Yeo, "Video content characterization and compaction for digital library applications," *SPIE, Storage and Retrieval for Still Image and Video Databases V*, vol. 3022, San Jose, CA, Feb. 1997, pp. 45–58.
[38] D. Zhong and S. F. Chang, "Video object model and segmentation for content-based video indexing," presented at the *IEEE Int. Conf. Circuits Syst.*, Hong Kong, June 1997 (Special Session on Networked Multimedia Technol. Appl.).

Shih-Fu Chang (S'89–M'90) received the Ph.D. degree in EECS from the University of California, Berkeley, in 1993.

He is currently an Associate Professor in the Department of Electrical Engineering and New Media Technology Center at Columbia University. At Columbia, he also actively participated in Columbia's Digital Library Project. His current research interests include content-based visual query, networked video manipulation, video coding, and communication. He is particularly interested in the application of content-based video processing to image/video retrieval, network resource management, image watermarking, and authentication. His group has developed several large-scale Web-based prototypes of visual information systems, including an MPEG video editing engine, WebClip, and content-based visual search engines, VideoQ, and WebSEEK. He holds two U.S. patents (with six more pending) in the area of visual search and compressed video processing. He has taught several short courses on visual information retrieval.

Dr. Chang was the recipient of two Best Paper Awards, an ONR Young Investigator Award 1998–2001, an NSF CAREER Award 1995–1998, and an IBM UPP Program Faculty Development Award 1995–1998. He actively participates in technical activities for international conferences and publications.

William Chen received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1993, and the M.S. degree in EECS from the University of California, Berkeley, in 1997.

He is currently a Ph.D. candidate in the Department of Electrical Engineering at Columbia University.

Horace J. Meng received the B.S. degree from the University of North Texas in 1993, and the M.S.E.E. degree from Columbia University in 1994.

He is currently a Ph.D. candidate and a Graduate Research Assistant at the Department of Electrical Engineering and New Media Technology Center, Columbia University. His current research interests include compressed domain video parsing, searching, browsing, editing, and watermarking. He has developed the first Web-based compressed video editing system: WebClip. He has two U.S. patents pending in compressed video editing and watermarking.

Mr. Meng has served on a number of conference and journal publications of IEEE, ACM, and SPIE.

Hari Sundaram received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1993, and the M.S. degree in electrical engineering from SUNY, Stony Brook, in 1995.

He is currently a Ph.D. candidate in the Department of Electrical Engineering, Columbia University. His current research interests include content-based multimedia retrieval, different forms of media representation including signal and higher level forms of representation, and issues in visual information retrieval.

Di Zhong received the B.S. and M.S. degrees in computer science from ZheJiang University, China, in 1990 and 1993, respectively, and the M.S. degree from the Institute of Systems Science of National University of Singapore, Singapore, in 1995. He is at present a Ph.D. candidate in the Department of Electrical Engineering, Columbia University.