VideoQ: An Automated Content Based Video Search System Using Visual Cues

Shih-Fu Chang William Chen Horace J. Meng Hari Sundaram Di Zhong

Dept. of Electrical Engineering
Columbia University
New York New York 10027.

E-mail: {sfchang,bchen,jmeng,sundaram,dzhong}@ctr.columbia.edu

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 focussed on still image retrieval. In this paper, we propose a novel, real-time, interactive system on the Web, based on the visual paradigm, with spatio-temporal attributes playing a key role in video retrieval. We have developed algorithms for automated video object segmentation and tracking and use real-time video editing techniques while responding to user queries. The resulting system performs well, with the user being able to retrieve complex video clips such as those of skiers, baseball players, with ease.

1. 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 [Chang 97], [Gupta 97]. While there has been substantial progress with the presence of systems such as QBIC [Flickner 95], PhotoBook [Pentland 96], Virage [Hamrapur 97] and VisualSEEk [Smith 96] most systems only support retrieval of still images. CBVQ research on video databases has not been fully explored yet. We propose an advanced content-based video search system with the following unique features:

ACM Multimedia *97 Seattle Washington USA*

- Automatic video object segmentation and tracking.
- A rich visual feature library including color, texture, shape, motion.
- Query with multiple objects.
- Spatio-temporal constraints on the query.
- Interactive querying and browsing over the World-Wide Web.
- Compressed-domain video manipulation.

Specifically, we propose to develop a novel video search system which allows users to search video based on a rich set of visual features and spatio-temporal relationships. Our objective is to investigate the full potential of visual cues in object-oriented content-based video search. While the search on video databases ought to necessarily incorporate the diversity of the media (video, audio, text captions) our research will complement any such integration.

We will present the the visual search paradigm in section 2, elaborate on the system overview in section 3, describe video objects and our automatic video analysis techniques in sections 4-5, discuss the matching criteria and query resolution in sections 7-8 and finally present some preliminary evaluation results in section 9.

2. The Visual Paradigm

The fundamental paradigm under which VideoQ operates is the visual one. This implies that the query is formulated exclusively in terms of elements having visual attributes alone. The features that are stored in the database are generated from an automatic analysis of the video stream. There is no information present in the query loop that emanates from the captions, textual annotations or the audio stream. Many retrieval systems such as PhotoBook [Pentland 96], VisualSEEk [Smith 96] and Virage [Hamrapur 97] share this paradigm, but only support still image retrieval. While QBIC [Flickner 95] is visual, it is not exclusively so as the images have been manually annotated allowing for keyword searches on the database.

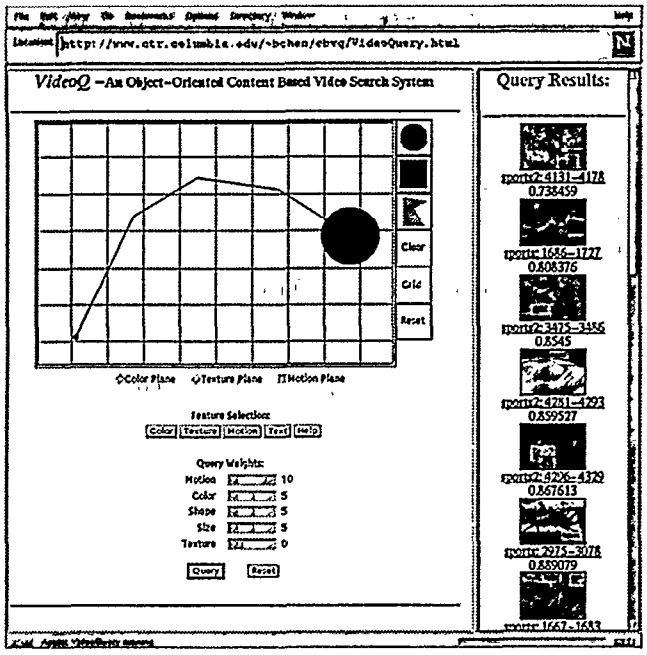

Figure 1. The visual interface of VideoQ. The figure shows an example query to retrieve video shots of all high jump sequences in the database. The retrieved shots which include three (the second, third and the seventh key-frame) successful matches, bear out the importance the motion attribute in video shot retrieval.

Video retrieval systems should evolve towards a systematic integration of all available media such as audio, video and captions. While video engines such as [Hauptmann 95], [Hamrapur 97], [Shahraray 95], [Mohan 96] attempt at 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 [Minka 96]. Examples of sketch based image retrieval systems include QBIC, VisualSEEk, [Jacobs 95], [Hirata 92] and [Del Bimbo 97]. These two different ways of visually searching image databases may also be accompanied by learning and user feedback [Minka 96].

Query by example systems work under the realization that since the "correct" match must lie within the database, one can begin the search with a member of the database itself. With the hope that one can guide the user towards 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 [Minka 96]. 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 whom are described by a complex multi-dimensional feature vector. The complexity arises partly due to the problem of describing shape and motion characteristics.

Sketch based query systems such as [Hirata 92] compute the correlation between the sketch and the the edge map of each of the images in the database, while in [Del Bimbo 97], the authors minimize an energy functional to achieve a match. In [Jacobs 95], 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 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 spatio-temporal ordering (and relationships) of these objects that fully define a scene. This is illustrated in Figure 1.

While we shall 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 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.

3. The VideoQ System Overview

VideoQ is a Web based video search system, where the user queries the system using animated sketches. An animated sketch is defined as a sketch where the user can assign motion to any part of the scene.

VideoQ 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, spatio-temporal ordering, shape and the 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, we have databases for each of these features. The video shot database is stored as compressed MPEG streams.

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

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

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.

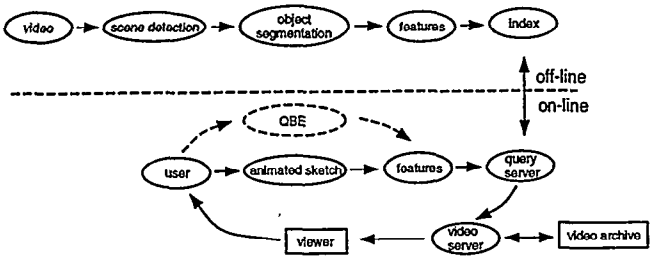

Figure 2. The VideoQ system where the queries are in the form of animated sketches. The dashed path shows a QBE (query by example) loop, which is absent in the current system, but will be incorporated into the implementation.

The user can interactively view these matched video shots over the network by simply clicking on the the key-frame. Then, in the backend, 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 [Meng 96] 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 of a collection of objects having spatio-temporal attributes, we need to formalize the definition of a video object.

4. What is a Video Object?

We define a region to be a contiguous set of pixels that is homogeneous in the 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 Figure 3, 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

[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

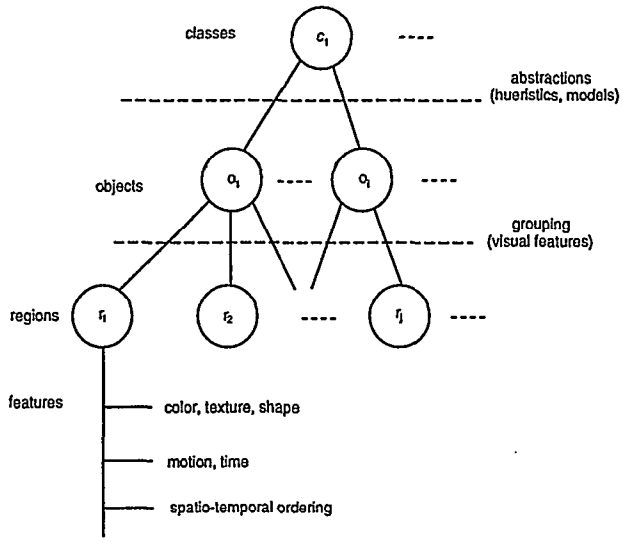

Figure 3. The feature classification tree.

attention to regions only. Regions may be assigned several attributes, such as color, texture, shape and motion.

4.1. 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 pre-defined and are easily inserted and manipulated.

4.2. Motion, Time

Motion is the *key* object attribute in VideoQ. The motion trajectory interface (Figure 4) allows the user to specify an arbitrary polygonal trajectory for the query object. The temporal attribute which defines the overall 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. The VideoQ implementation of the temporal attributes is shown in figure 4.

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

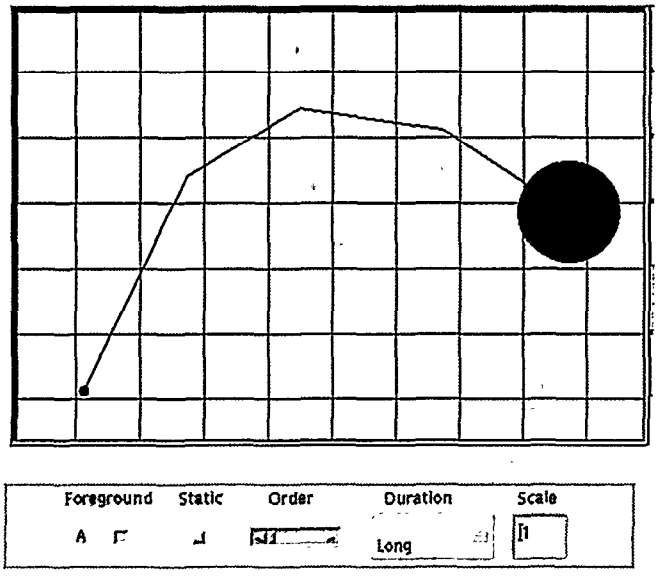

Figure 4. The spatio-temporal description of an video object. The user can assign the duration, and an arbitrary trajectory to the video object. In addition, the user also specifies the temporal arrival order of the object in the scene.

4.3. 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 Figure 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.

5. 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.

5.1. 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 change are either abrupt scene changes or transitional (e.g. dissolve, fade in/out, wipe). [Meng 95] 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.

5.2. 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 [Sawhney 95]. A hierarchical pixel-domain motion estimation method [Bierling 88] 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:

$$\Delta x = a_o + a_1 x + a_2 y, \quad (1)$$

$$\Delta y = a_3 + a_4 x + a_5 y, \quad (2)$$

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".

5.3. 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 Figure 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 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 [Bierling 88]. Given color regions and optical flow generated from above two processes, a linear regression algorithm is used to estimate the affine

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

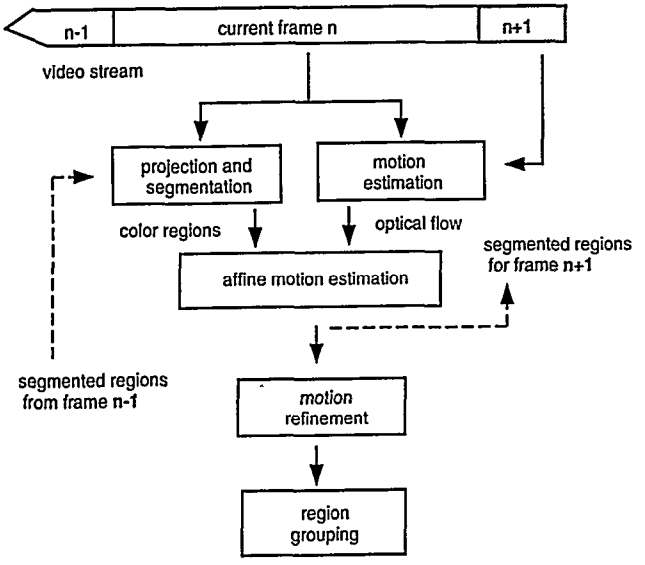

Figure 5. Region segmentation and tracking of frame n.

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 $n+1$.

5.3.1. Projection and Segmentation Module

Now we discuss the projection and segmentation module [Zhong 97] (see Figure 6). In the first step, the current frame (i.e. frame n) is quantized in a perceptually uniform color space (e.g., CIE LUV space). Quantization palettes can be obtained by a uniform quantizer or clustering algorithms (e.g., self-organization map). After quantization, non-linear median filtering is used to eliminate insignificant details and outliers in the image while preserving edge information. In the meanwhile, edge map of frame n is extracted using edge detectors (e.g. Canny edge detector).

For the first frame in the sequence, the system will go directly to intra-frame segmentation. For intermediate frames, as region information is available from frame $n-1$, an interframe projection algorithm is used to track these regions. All regions in frame $n-1$ are projected into frame n using their affine motion estimates. For every pixel in frame n that is covered by regions projected from the previous frame, we label it as belonging to the region to which it is closest in the CIE-LUV space. If a pixel is not covered by any projected region, then it remains unlabeled.

The tracked regions together with un-labeled pixels are further processed by an intra-frame segmentation algorithm. An iterative clustering algorithm is adopted: two adjoining regions with the smallest color distance are continuously merged until the difference is larger than a given threshold. Finally, small regions are merged to their neighbors by a morphological open-close algorithm. Thus, the whole procedure generates homogeneous color regions in frame n while tracking existing regions from frame $n-1$.

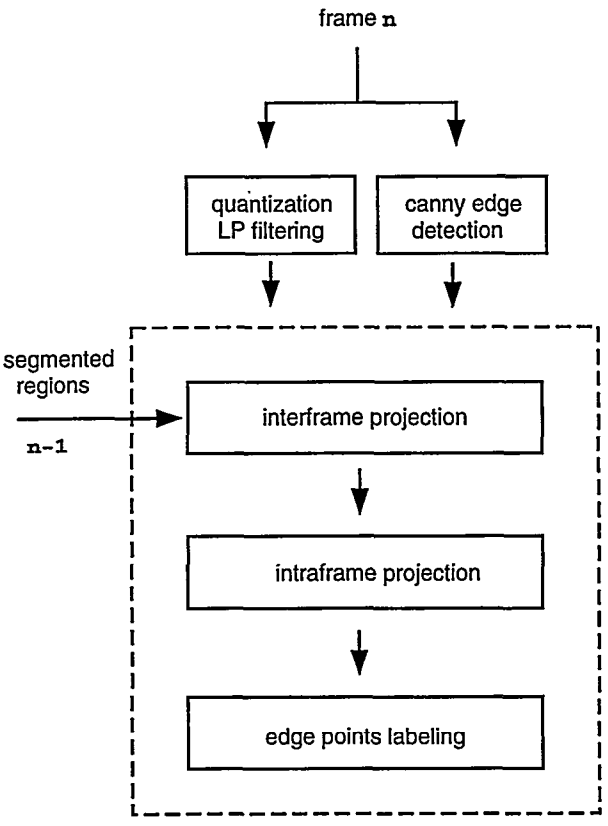

Figure 6. Region projection and segmentation of frame n

The edge map is used to enhance color segmentation accuracy [Zhong 97]. For example, regions clearly separated by long edge lines will not be merged with each other. Short edge lines which are usually inside one color region will not affect the region merging process.

Figure 7 shows segmentation results with two sequences. In both cases, the top row shows 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

6. 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.

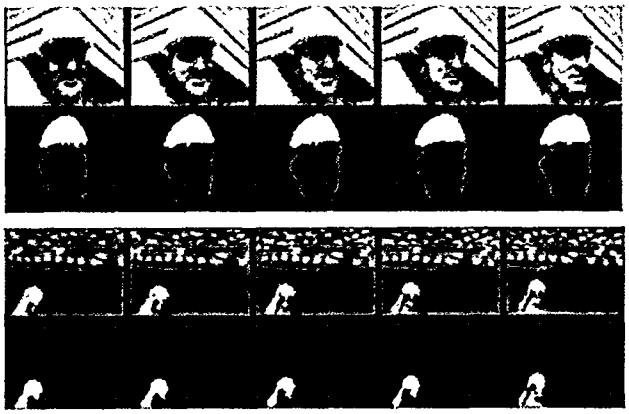

Figure 7. Region segmentation on QCIF sequences, using feature fusion. The top rows show the original sequence while the corresponding bottom rows show the segmented regions.

Texture Three Tamura [Tamura 78] 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 [Sawhney 95]. 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 [Saber 97a]. At the same time we generate first and second order moments of the region. Two other new features, the normalized area[6], and the 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 $O(N^2 \log N)$ [Arkin 91].

The resulting library is a simple database having a {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.

7. 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 computational complexity of the metric and the quality of the match. For 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 spatio-temporal information in the video.

7.1. Matching Motion Trails

A motion trail is defined to be the three dimensional trajectory of a video object. It is represented by a sequence $\{x[i], y[i]\}, i \in \{1, ..N\}$, the three dimensions comprising of 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 [Dimitrova 94], have used simple chain codes or a B-spline to represent the trajectory, without completely capturing the spatio-temporal 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 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 is 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].

Spatio-Temporal In the spatio-temporal mode, we simply use the entire motion trail to compute the distance. We use the following distance metric:

$$\sum_i ((x_q[i] - x_t[i])^2 + (y_q[i] - y_t[i])^2), \qquad (3)$$

[5] We could have also stored the 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 "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

[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/sec.

[9] A 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.

where, the subscripts q and t refer to the query and the target trajectories respectively and the index i runs over the 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 till the shorter of the two durations (i.e the index i runs up til min (*query duration*, *database object duration*) and ignore the "tail").
- We could also normalize the the two durations to a canonical duration and then perform the match.

7.2. Matching Other Features

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

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 [Tamura 78] 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 the 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.

8. Query Resolution

Using these feature space metrics and the composite distance function, we compute the composite distance of each object in the database with the 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 an single object query along with the results (the candidate result) is shown in Figure 1.

8.1. 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 Figure 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.

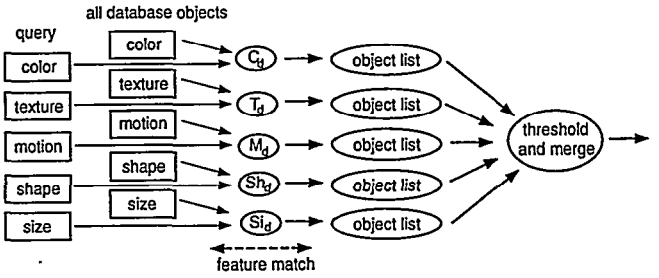

Figure 8. Generating the candidate video shot list for a single object query. The first column of features shows the features of the query, while the second column shows the features across all objects in the database.

9. How does VideoQ perform?

Evaluating the performance of video retrieval systems is still very much an open research issue [Chang 97]. There

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

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

does not exist a standard video test set to measure retrieval performance nor standard benchmarks to measure system performance. This is partly due to the emerging status of this field. To evaluate 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.

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

Before each sample query, the user establishes a ground truth by choosing a set of relevant or "desired" video shots from the database. For each sample query shown in Figure 9, a ground truth is established by choosing all 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 equations 9, 10. A precision-recall curve is generated by increasing the size of the return list and computing precision-recall values for each size.

$$\text{Recall} = \frac{\text{Retrieved and relevant}}{\text{All relevant in the database}} \quad (9)$$

$$\text{Precision} = \frac{\text{Retrieved and relevant}}{\text{Number retrieved}} \quad (10)$$

where, the relevant video shots are predefined by the ground truth database.

Four sample queries were performed as shown Figure 9. 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 to 20 video shots, and a precision-recall curve is plotted in Figure 10 (a).

An overlay of the four precision-recall curves is plotted in Figure 10. For an ideal system, the precision-recall curve is a horizontal line with precision value of 1.0 as recall ranges from 0.0 to 1.0. For normal systems, the precision-recall curve remains relatively flat up to a certain recall value, after which the curve slopes downward. We note that a breakpoint as the point where the number of retrieved video shots equals the number of relavant video shots. The breakpoints of the four precision-recall curves are marked with a cross. The distance of the breakpoint from 1.0 (the optimal recall point) indicates how effective that particular query was; the farther the distance of breakpoint, the less optimal the performance.

Both sample queries (a) and (c) performed well in Figure 10. In both cases, the motion trajectories were well-defined, not easily confused with camera motion. The sample query (b), however, did not perform as well. One reason is that, in some video shots, the background objects were not properly compensated by the global motion compensation algorithm and were treated as foreground objects. Since the video database contains many shots where the camera is panning from right to left, those same background objects were indexed with left/right motion trajectories. The average precision-recall curve is also calculated and plotted in Figure 11.

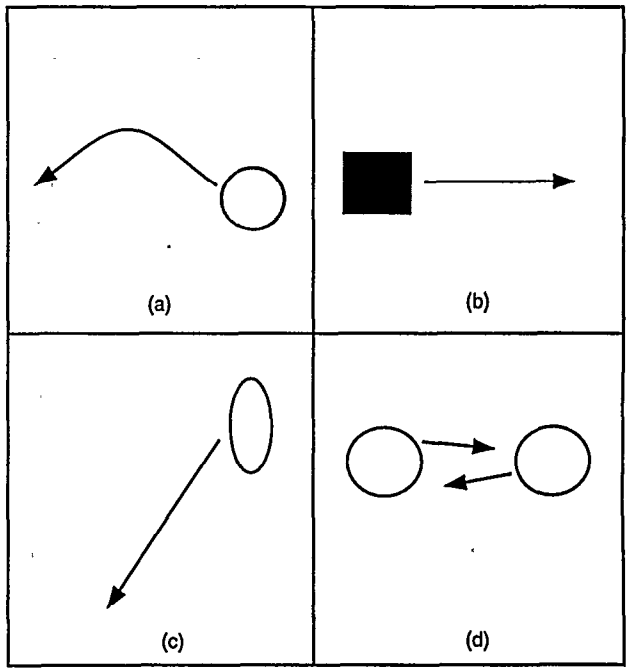

Figure 9. Four sample queries used in the precision-recall experiments. (a-b) Highlights motion, color and size. (c) Highlights motion and size. (d) Highlights multiple objects in addition to motion and size.

9.2. 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 measures how many separate queries are needed to get a particular video shot in the return list. Bandwidth measures how many different false alarms are returned

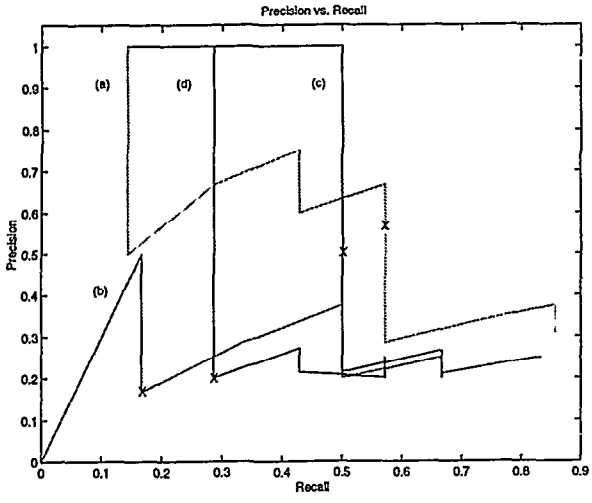

Figure 10. The precision-recall curves corresponding to the sample queries of Figure 9

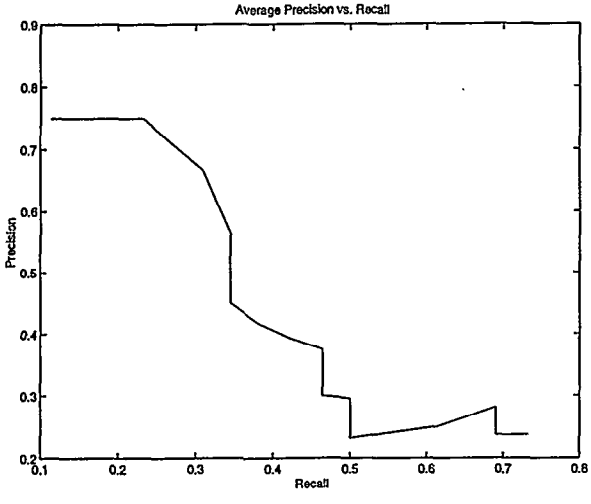

Figure 11. The precision-recall curve averaged over the four precision-recall curves of Figure 10

before obtaining the correct video shot in the return list.

A randomly generated target video shot, shown in Figure 12 (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, the query frequency curve is generated.

Each query returns a list of video shots from an HP 9000 server over the network to a client. The video shot is actually represented by a 88x72 key frame. In many cases, a series of queries were 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. 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 Figure 12, 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 Figure 13 shows that a greater number of queries are needed for small return sizes. On average for a return size of 14, only two queries are needed to reach the desired video shot.

In Figure 14, the bandwidth curve linearly decreases as return size decreases. For small return sizes, many times ten or more queries failed to place the video shot within the return list. These "failed" videos, shown in Figure 15, were simply discarded in this case. In Figure 16, we compensate for these failed queries by applying a heuristic to penalize the returned videos. Once this was done, the average bandwidth was recalculated in Figure 16, which shows that a medium return size requires the least amount of bandwidth.

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.

9.3. 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 an logical intersection of all the results of the individual query objects. When we perform a multiple object query in the 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 2D strings [Chang 87], [Shearer 97], [Smith 96] (discussed in 10.3).

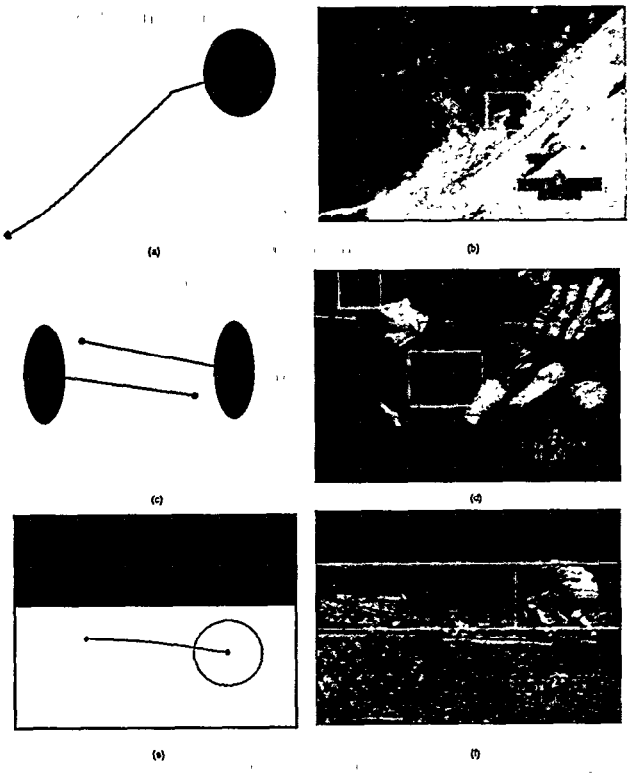

Figure 12. Three sample queries used in system benchmarks. The left column shows the final sketch to successfully retrieve the video. (a) A Skier (b) Two soccer players (c) A baseball query. In the baseball video clip, the catcher moves to the left. Also note the strong match of the sky-like texture to the sky.

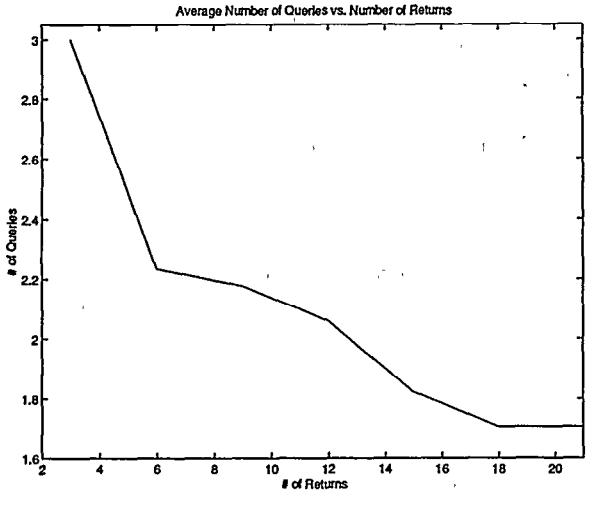

Figure 13. Average number of queries needed to reach a video shot (accounting for successful cases only).

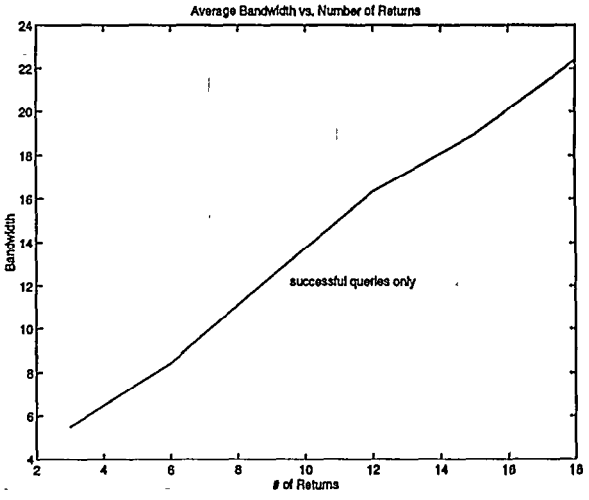

Figure 14. Average bandwidth used in reaching a video shot (accounting for successful cases only).

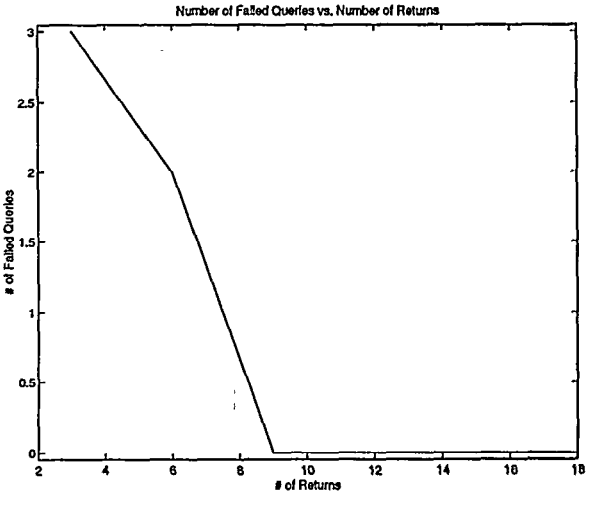

Figure 15. Number of failed videos for a particular return size.

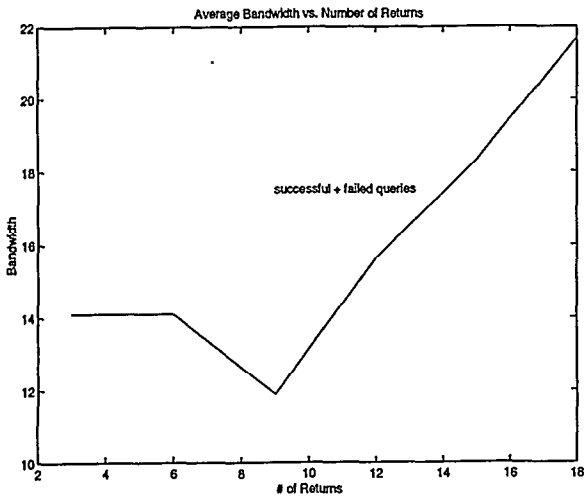

Figure 16. Average bandwidth used including the failed videos.

10. Research Issues in VideoQ

While the results section (Section 9) demonstrates that VideoQ works well, there are other issues that need to be addressed. This section contains a brief overview of the issues that we are currently working on.

10.1. Region Grouping

Automatic region grouping, is an open problem in computer vision, and in spite of decades of research, we are still far from completely a 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 avoid over-segmentation of the video shot.

10.2. 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 $O(N^2 \log N)$ [Arkin 91].

One approach is to use geometric invariants to represent shape [Mundy 92], [Karen 94], [Lei 95]. 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.

10.3. Spatio-Temporal Search

We are currently extending the work done on VisualSEEk [Smith 96] on 2-D strings [Chang 87] in order to effectively constrain the query results. There has been work using modified 2-D strings as a spatial index into videos [Arndt 89], [Shearer 97].

For video, 2-D strings can be extended to a sequence of 2D-strings or a 2D-string followed by a sequence of change edits [Shearer 97]. Building on these observations we propose two efficient methods for indexing spatio-temporal structures of segmented video objects.

- In the first method, only frames with significant changes of spatial structures need to be explicitly indexed (by 2D 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 2D strings at the beginning frame, the ending frame, and several sampled frames in between.

- The second method extends the 2D-string based query to 3D-strings. Video objects may be projected to x, y and time dimensions to index their absolute centroid position, 3-dimensional support, and relative relationships. More sophisticated variations of 3D strings can be used to handle complex relationships such as adjacency, containment, overlap.

11. 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. Annotating video objects with motion attributes and good spatio-temporal 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 multi-object queries, and the content-based visual matching of spatio-temporal 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 spatio-temporal 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

[Arkin 91] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kadem, J.S.B. Mitchell *An Efficiently Computable Metric for Comparing Polygonal Shapes* IEEE Trans. on PAMI, Vol. 13, No. 3, 209-216, Mar. 91.

[Arndt 89] T. Arndt, S.K. Chang, *Image Sequence Compression by Iconic Indexing*, IEEE Workshop on Visual Languages, 177-182, IEEE Computer Society, 1989.

[Bierling 88] M. Bierling, *Displacement Estimation by Hierarchical Block Matching*, SPIE Visual Communications and Image Processing, Vol. 1001, 1988.

[Del Bimbo 97] A. Del Bimbo, P. Pala *Visual Image Retrieval by Elastic Matching of User Sketches*, IEEE Trans. on PAMI, Vol. 19, No. 2, 121-132, Feb. 1997.

[Borshukov 97] G.D. Borshukov, G. Bozdagi, Y. Altunbasak, A.M. Tekalp, *Motion segmentation by multi-stage affine classification*, to appear in IEEE Trans. Image Processing.

[Brown 96] M.G. Brown, J.T. Foote, G.J.F. Jones, K.S. Jones, S.J. Young, *Open-Vocabulary Speech Indexing for Voice and Video Mail Retrieval*, ACM Multimedia Conference, Boston, Nov. 1996.

[Chang 97] S.F. Chang, J.R. Smith, H.J. Meng, H. Wang, D. Zhong, *Finding Images/Video in Large Archives*, CNRI Digital Library Magazine, Feb. 1997.

[Chang 87] S.K. Chang, Q.Y. Shi, S.Y. Yan, *Iconic Indexing by 2-D Strings*, IEEE Trans. on PAMI, Vol. 9, No. 3, 413-428, May 1987.

[Dimitrova 94] N. Dimitrova, F. Golshani, $\mathcal{R}_X$ *for Semantic Video Database Retrieval*, ACM Multimedia Conference, 219-226, San Francisco, Oct. 1994.

[Faloutsos 93] C. Faloutsos, M. Flickner, W. Niblack, D. Petkovic, W. Equitz, R. Barber, *Efficient and Effective Querying by Image Content*, Research Report #RJ 9203 (81511), IBM Almaden Research Center, San Jose, Aug. 1993.

[Flickner 95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, P. Yanker, *Query by Image and Video Content: The QBIC System*, IEEE Computer Magazine, Vol. 28, No. 9, pp. 23-32, Sep. 1995.

[Gupta 97] A. Gupta, R. Jain, *Visual Information Retrieval*, Communications of ACM, Vol. 40, No. 5, 70-79, May 1997.

[Hamrapur 97] A. Hamrapur, A. Gupta, B. Horowitz, C.F. Shu, C. Fuller, J. Bach, M. Gorkani, R. Jain, *Virage Video Engine* SPIE Proceedings on Storage and Retrieval for Image and Video Databases V, 188-97, San Jose, Feb. 97.

[Hauptmann 95] A.G. Hauptmann, M. Smith, *Text, Speech and Vision for Video Segmentation: The Informedia Project*, AAAI Fall Symposium, Computational Models for Integrating Language and Vision, Boston, Nov. 1995.

[Hirata 92] K. Hirata, T. Kato, *Query by Visual Example, Content Based Image Retrieval*, Advances in Database Technology - EDBT 92, A. Pirotte, C. Delobel, G. Gottlob, eds., Lecture Notes on Computer Science, Vol. 580.

[Jacobs 95] C.E. Jacobs, A. Finkelstein, D.H. Salesin, *Fast Miltiresolution Image Querying*, Proceedings of SIGGRAPH, 277-286, Los Angeles, Aug. 1995.

[Jones 81] K.S. Jones. *Information Retrieval Experiment*, Butterworth and Co., Boston, 1981.

[Karen 94] D. Karen, D. Cooper, J. Subrahmonia, *Describing Complicated Objects by Implicit Polynomials*, IEEE Trans. on PAMI, Vol. 16, No. 1, 38-53, Jan. 1994.

[Lei 95] Z. Lei, D. Karen, D. Cooper, *Computationally Fast Bayesian Recognition of Complex Objects Based on Mutual Algebraic Invariants*, Proc. IEEE International Conference on Image Processing, Vol. 2, 635-638, Oct. 1995.

[Meng 95] J. Meng, Y. Juan, S.F. Chang, *Scene Change Detection in a MPEG Compressed Video Sequence*, SPIE Symposium on Electronic Imaging: Science and Technology - Digital Video Compression: Algorithms and Technologies, SPIE Vol. 2419, San Jose, Feb. 1995.

[Meng 96] J. Meng, S.F. Chang, *CVEPS: A Compressed Video Editing and Parsing System*, ACM Multimedia Conference, Boston, Nov. 1996. `http://www.ctr.columbia.edu/webclip`

[Minka 96] T. Minka, *An Image Database Browser that Learns from User Interaction*, MIT Media Laboratory Perceptual Computing Section, TR #365, 1996.

[Mohan 96] R. Mohan, *Text Based Search of TV News Stories*, SPIE Photonics East, International Conference on Digital Image Storage and Archiving Systems, Boston, Nov. 1996.

[MPEG4 96] *Description of MPEG-4*, ISO/IEC JTC1/SC29/WG11 N1410, MPEG document N1410 Oct. 1996.

[Mundy 92] J.L. Mundy, A. Zisserman, eds. *Geometric Invariance in Computer Vision*, MIT Press, 1992.

[Pentland 96] A. Pentland, R.W. Picard, and S. Sclaroff, *Photobook: Content-Based Manipulation of Image Databases*, International Journal of Computer Vision, Vol. 18, No. 3, pp. 233-254, 1996.

[Saber 97a] E. Saber, A.M. Tekalp, *Region-based affine shape matching for automatic image annotation and query-by-example*, to appear in Visual Comm. and Image Representation.

[Saber 97b] E. Saber, A.M. Tekalp, G. Bozdagi, *Fusion of Color and Edge Information for Improved Segmentation and Linking*. to appear in Image and Vision Computing.

[Sawhney 95] H.S. Sawhney, S. Ayer, M. Gorkani, *Model-based 2D & 3D Dominant Motion Eastimation for Mosaicing and Video Representation*, International Conference on Computer Vision, 583-590, Boston, Jun. 1995.

[Shahraray 95] B. Shahraray and D.C. Gibbon, *Automatic Generation of Pictorial Transcript of Video Programs*, SPIE Vol. 2417, 512-518, 1995.

[Shearer 97] K. Shearer, S. Venkatesh, D. Kieronska, *Spatial Indexing for Video Databases*, To appear in Journal of Visual Communication and Image Representation, Vol. 8, No. 3, Sep. 1997.

[Smith 96] J.R. Smith, S.F. Chang, *VisualSEEk: A Fully Automated Content-Based Image Query System*, ACM Multimedia Conference, Boston, 87-98, Nov. 1996. `http://www.ctr.columbia.edu/VisualSEEk`

[Smoliar 94] S.W. Smoliar, H.J. Zhang, *Content-Based Video Indexing and Retrieval*, IEEE Multimedia Magazine, Summer 1994.

[Tamura 78] H. Tamura, S. Mori, *Textural Features Corresponding to Visual Perception*, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 8, No. 6, Jun. 1978.

[Yeung 97] M.M. Yeung , B.L. Yeo, *Video Content Characterization and Compaction for Digital Library Applications*, SPIE, Storage and Retrieval for Still Image and Video Databases V, SPIE Vol. No. 3022, 45-58, San Jose, Feb. 1997.

[Zhong 97] D. Zhong and S.F. Chang, *Video Object Model and Segmentation for Content-Based Video Indexing*, IEEE International Conference on Circuits and Systems, Jun. 1997, Hong Kong. (special session on Networked Multimedia Technology and Applications)