UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MASSACHUSETTS

CANON, INC., CANON U.S.A., INC. AND AXIS COMM. AB, *Plaintiffs,*	
v.	Civil Action No. 1:19-MC-91401-NMG-JCB
AVIGILON FORTRESS CORPORATION, *Defendant.*	

DECLARATION OF KATHERINE ZIMMERMAN

I, Katherine Zimmerman, state and declare as follows:

1. I am a Scholarly Communications and Licensing Librarian at the Massachusetts Institute of Technology ("MIT") Libraries, 105 Broadway, Building NE36, Suite 6101, Cambridge, Massachusetts 02142.
2. I am over 18 years of age and am competent to make this Declaration. I make this Declaration based on my own personal knowledge, based on my knowledge and review of the business records and practices of the MIT Libraries, based on conversations with other library staff, and based on the notes and records of Marilyn McSweeney who prepared Declarations until her retirement in 2016.
3. I have been employed at MIT since 2016.
4. Through the actions described in paragraph 2, I have become knowledgeable about the MIT Libraries' normal business practices with respect to how MIT receives, catalogs, indexes, shelves, and makes available to the public journals and publications.

5. Attached as Exhibit A to this Declaration is a true and accurate copy of the catalog record from the MIT Libraries' online catalog system (known as the Barton Catalog) for the publication series entitled ACM Transactions on Information Systems: a publication of the Association for Computing Machinery vols. 7 (1989) - 26 (2008) ("ACM Transactions on Information Systems"). This is a record that MIT maintains in the ordinary course of its regular activities.

6. Attached as Exhibit B to this Declaration is a true and accurate copy of the issue cover, first page, back cover, and full article text, for the article titled "Motion Recovery for Video Content Classification" by Nevenka Dimitrova and Forouzan Golshani published on pages 408-439 of Volume 13, No. 4 of the ACM Transactions on Information Systems, which was published in October 1995 (the "October 1995 Issue."). The ACM Transactions on Information Systems is available in print format in vols. 7 (1989) - 26 (2008) from the MIT Libraries, and is a record that MIT maintains in the ordinary course of its regular activities.

7. The October 1995 Issue has an MIT Libraries date stamp of "NOV 13 1995," indicating that the MIT Libraries received the issue on November 13, 1995.

8. After a serials issue receives a date stamp, it undergoes a process of being labeled and moved to a shelf of the MIT Libraries. Based on current MIT Libraries practice, this process typically takes one to two weeks. According to the MIT Libraries' current normal business practice, the October 1995 Issue would have been displayed on a shelf of the MIT Libraries no later than November 27, 1995.

9. Once a publication is on a shelf of the MIT Libraries it is available to be viewed within the MIT Libraries by any member of the public or requested via Interlibrary Loan.
10. To the best of my knowledge and that of current MIT employees, unless stated otherwise, the above statements are descriptions of normal business practices at the MIT Libraries from at least the beginning of 1995 and through the present.

I declare under penalty of perjury that the foregoing is true and correct. Executed on October 23, 2019, at Cambridge, Massachusetts.

KATHERINE ZIMMERMAN

EXHIBIT A

Full Record

Permalink for this record: http://library.mit.edu/item/000395517

Results List | **Add to Bookshelf** | **Save/Email**

Choose format: **Standard** | **Citation** | **MARC tags**

Record 2 out of 3 Prev record Next record

Title	**ACM transactions on information systems** : a publication of the Association for Computing Machinery.
Continues	**ACM transactions on** office **information systems**
Online Access	Get this MIT v.7:no.1 (1989:Jan.)-
Library Holdings	Library Storage Annex - Off Campus Collection | HF.A184 v.7(1989)-v.26(2008)
Shelf Access	Find it in the library/Request item
Shelf Location	Library Storage Annex - Off Campus Collection | HF.A184
Published	New York, NY : The Association, c1989-
Description	v. : ill. ; 26 cm.
Numbering	Vol. 7, no. 1 (Jan. 1989)-
Series	**ACM** series **on** computing methodologies.
Current Frequency	Quarterly
Format	Serial (e.g. journals, book series, etc.)
Note	Title from cover.
Other Format	Also available via the World Wide Web.
Subject	Electronic data processing -- Periodicals. **Information** storage and retrieval **systems** -- Periodicals. **Information** retrieval -- Periodicals.
Other Author	Association for Computing Machinery.
Title Abbreviation	**ACM** trans. inf. sys.
Other Title	**Transactions on information systems.** Association for Computing Machinery **transactions on information systems.**
ISSN	1046-8188
CODEN	ATISET
Local System Number	000395517

Prev record Next record

Basic Search of Full Catalog

Search type: Search for:

Keyword
Title begins with...
Title Keyword
Author (last name first)
Author Keyword
Call Number begins with...
----- Scroll down for more choices -----

Search

Barton Questions: Ask Us! | Contact Us
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139-4307 USA

-- Quick Links --

EXHIBIT B

October 1995
Volume 13 Number 4

acm Series on
Computing
Methodologies

acm Transactions on Information Systems

Special Issue on Video Information Retrieval

acm PRESS

acm Transactions on Information Systems

1515 Broadway
New York, NY 10036
Tel: (212) 869-7440

ACM European Service Center
Avenue Marcel Thiry 204
1200 Brussels, Belgium
Tel: 32 2 774 9602
Fax: 32 2 774 9690
Email: acm_europe@acm.org

ACM Transactions on Information Systems (ISSN 1046-8188) is published 4 times a year in January, April, July, and October by the Association for Computing Machinery, Inc., 1515 Broadway, New York, NY 10036. Second-class postage paid at New York, NY 10001, and at additional mailing offices. Postmaster: Send address changes to Transactions on Information Systems, ACM, 1515 Broadway, New York, NY 10036.

For subscription and submissions information, see inside back cover.

Motion Recovery for Video Content Classification

NEVENKA DIMITROVA and FOROUZAN GOLSHANI
Arizona State University, Tempe

Like other types of digital information, video sequences must be classified based on the semantics of their contents. A more-precise and completer extraction of semantic information will result in a more-effective classification. The most-discernible difference between still images and moving pictures stems from movements and variations. Thus, to go from the realm of still-image repositories to video databases, we must be able to deal with motion. Particularly, we need the ability to classify objects appearing in a video sequence based on their characteristics and features such as shape or color, as well as their movements. By describing the movements that we derive from the process of motion analysis, we introduce a dual hierarchy consisting of spatial and temporal parts for video sequence representation. This gives us the flexibility to examine arbitrary sequences of frames at various levels of abstraction and to retrieve the associated temporal information (say, object trajectories) in addition to the spatial representation. Our algorithm for motion detection uses the motion compensation component of the MPEG video-encoding scheme and then computes trajectories for objects of interest. The specification of a language for retrieval of video based on the spatial as well as motion characteristics is presented.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*motion*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Content-based retrieval of video, motion recovery, MPEG compressed video analysis, video databases, video retrieval

1. INTRODUCTION

Applications such as video on demand, automated surveillance systems, video databases, industrial monitoring, video editing, road traffic monitoring, etc. involve storage and processing of video data. Many of these applications can benefit from retrieval of the video data based on their content. The problem is that, generally, any content retrieval model must have the capability of

This article is a revised version with major extensions of an earlier paper which was presented at the ACM Multimedia '94 Conference.
Authors' addresses: N. Dimitrova, Philips Laboratories, 345 Scarborough Road, Briarcliff Manor, NY 10562; email: nvd@philabs.philips.com; F. Golshani, Department of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287-5406; email: golshani@asu.edu.

dealing with massive amounts of data. As such, classification is an essential step for ensuring the effectiveness of these systems.

Motion is an essential feature of video sequences. By analyzing motion of objects we can extract information that is unique to the video sequences. In human and computer vision research there are theories about extracting motion information independently of recognizing objects. This gives us support for the idea of classifying sequences based on the motion information extracted from video sequences regardless of the level of recognition of the objects. For example, using the motion information we can not only submit queries like "retrieve all the video sequences in which there is a moving pedestrian and a car" but also queries that involve the exact position and trajectories of the car and the pedestrian.

Previous work in dynamic computer vision can be classified into two major categories based on the type of information recovered from an image sequence: recognition through recovering structure from motion and recognition through motion directly. The first approach may be characterized as attempting to recover either low-level structures or high-level structures. The low-level structure category is primarily concerned with recovering the structure of rigid objects, whereas the high-level structure category is concerned primarily with recovering nonrigid objects from motion. Recovering objects from motion is divided into two subcategories: low-level motion recognition and high-level motion recognition. Low-level motion recognition is concerned with making the changes between consecutive video frames explicit (this is called optical flow [Horn and Schunck 1981]). High-level motion recognition is concerned with recovering coordinated sequences of events from the lower-level motion descriptions.

Compression is an inevitable process when dealing with large multimedia objects. Digital video is compressed by exploiting the inherent redundancies that are common in motion pictures. Compared to encoding of still images, video compression can result in huge reductions in size. In the compression of still images, we take advantage of spatial redundancies caused by the similarity of adjacent pixels. To reduce this type of redundancy, some form of transform-based coding (e.g., Discrete Cosine Transform, known as DCT) is used. The objective is to transform the signal from one domain (in this case, spatial) to the frequency domain. DCT operates on 8×8 blocks of pixels and produces another block of 8×8 in the frequency domain whose coefficients are subsequently quantized and coded. The important point is that most of the coefficients are near zero and after quantization will be rounded off to zero. Run-length coding, which is an algorithm for recording the number of consecutive symbols with the same value, can efficiently compress such an object. The next step is coding. By using variable-length codes (an example is Huffman tables), smaller code words are assigned to objects occurring more frequently, thus further minimizing the size.

Our aim in the coding of video signals is to reduce the temporal redundancies. This is based on the fact that, within a sequence of related frames, except for the moving objects, the background remains unchanged. Thus to reduce temporal redundancy a process known as motion compensation is

used. Motion compensation is based on both predictive and interpolative coding.

MPEG (Moving Pictures Expert Group) is the most general of the numerous techniques for video compression [Furht 1994; LeGall 1991; Mattison 1994]. In fact, the phrase "video in a rainbow" is used for MPEG, implying that by adjusting the parameters, one can get a close approximation of any other proposal for video encoding. Motion compensation in MPEG consists of predicting the position of each 16 × 16 block of pixels (called a macroblock) through a sequence of predicted and interpolated frames. Thus we work with three types of frames—namely, those that are fully coded independently of others (called reference frames or I-frames), those that are constructed by prediction (called predicted frames or P-frames), and those that are constructed by bidirectional interpolation (known as B-frames). It begins by selecting a frame pattern which dictates the frequency of I-frames and the intermixing of other frames. For example, the frame pattern IBBPBBI indicates (1) that every seventh frame is an I-frame, (2) that there is one predicted frame in the sequence, and (3) that there are two B-frames between each pair of reference and/or predicted frames. Figure 1 illustrates this pattern.

Our approach to extracting object motion is based on the idea that during video encoding by the MPEG method, a great deal of information is extracted from the motion vectors. Part of the low-level motion analysis is already performed by the video encoder. The encoder extracts the motion vectors for the encoding of the blocks in the predicted and bidirectional frames. A macroblock can be viewed as a coarse-grained representation of the optical flow. The difference is that the optical flow represents the displacement of individual pixels while the macroblock flow represents the displacement of macroblocks between two frames. At the next, intermediate level, we extract macroblock trajectories which are spatiotemporal representations of macroblock motion. These macroblock trajectories are further used for object motion recovery. At the highest level, we associate the event descriptions to object/motion representations.

Macroblock displacement in each individual frame is described by the motion vectors which form a coarse optical-flow field. We assume that our tracing algorithm is fixed on a moving set of macroblocks and that the correspondence problem is elevated to the level of macroblocks instead of individual points. The advantage of this elevation is that even if we lose individual points (due to turning, occlusion, etc.) we are still able to trace the object through the displacement of a macroblock. In other words, the correspondence problem is much easier to solve and less ambiguous. Occlusion and tracing of objects which are continuously changing are the subject of our current investigations.

In Section 2 of this article we survey some of the research projects related to our work. In Section 3 we present the object motion analysis starting from the low-level analysis through the high-level analysis. We discuss the importance of motion analysis and its relevance to our model which is presented in Section 3.4. Section 4 introduces the basic OMV structures (object, motion,

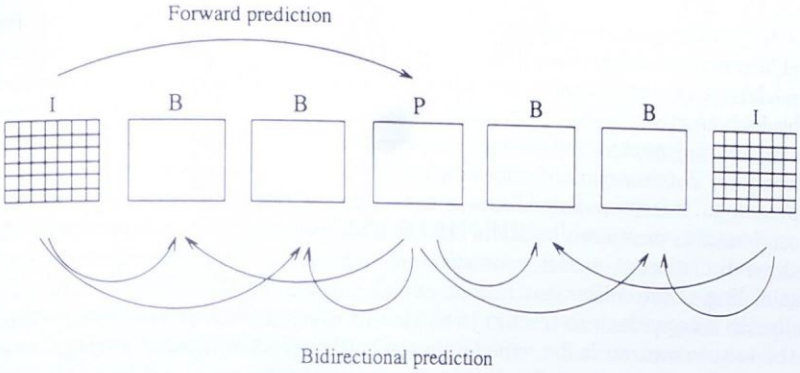


Fig. 1. Forward and bidirectional prediction in MPEG.

video-sequence), as the basis for the video information model. The basic retrieval operators, the OMV-language specification, and some examples are given. Empirical results are outlined in Section 5, and Section 6 presents some concluding remarks.

2. RELATED WORK

The research presented in this article builds on the existing results in two areas: dynamic computer vision and digital video modeling.

A current trend in computational vision is influenced by the idea that motion analysis does not depend on complex-object descriptions. Our work follows this trend and is based on the recent publications that are in agreement with this idea in computational vision. The idea of object/event recognition regardless of the existence of object representations can be traced back to the early 70's when Johansson [1976] introduced his experiments with *moving-light displays*. The idea was to attach lights to the joints of a human subject dressed in dark-colored clothing and observe the motion of lights against a dark background. The audience not only could recognize the object (human being) but could also describe the motion and the events taking place. Goddard [1992] investigated the high-level representations and computational processes required for the recognition of human motion based on moving-light displays. The idea is that recognition of any motion involves indexing into stored models of the movement. These stored models, called scenarios, are represented based on coordinated sequences of discrete motion events. The structures and the algorithms are articulated in the language of structured connectionist models. Allmen [1991] introduced a computational framework for intermediate-level and high-level motion analysis based on spatiotemporal surface flow and spatiotemporal flow curves. Spatiotemporal surfaces are projections of contours over time. Thus, these surfaces are direct representations of object motion.

In the dynamic computer vision literature there are general models for object motion estimation and representation, as well as domain-restricted models. A general architecture for the analysis of moving objects is proposed by Kubota et al. [1993]. The process of motion analysis is divided into three stages: moving-object candidate detection, object tracking, and final motion analysis. The experiments are conducted using human motion. Another approach to interpretation of the movements of articulated bodies in image sequences is presented by Rohr [1994]. The human body is represented by a three-dimensional model consisting of cylinders. This approach uses the modeling of the movement from medical motion studies. Koller et al. [1993] discuss an approach to tracking vehicles in road traffic scenes. The motion of the vehicle contour is described using an affine motion model with a translation and a change in scale. A vehicle contour is represented by closed cubic splines. We make use of the research results in all these domain-specific motion analysis projects. Our model combines the general area of motion analysis with individual frame (image) analysis.

In case of video modeling, the video footage usually is first segmented into shots. Segmentation is an important step for detection of cut points which can be used for further analysis. Each video shot can be represented by one or more key frames. Features such as color, shape, and texture could be extracted from the key frames. An approach for automatic video indexing and full video search is introduced by Nagasaka and Tanaka [1992]. This video-indexing method relies on automatic cut detection and selection of first frames within a shot for content representation. Otsuji and Tonomura [1993] propose a video cut detection method. Their projection detection filter is based on finding the biggest difference in consecutive-frame histogram differences over a period of time. A model-driven approach to digital video segmentation is proposed by Hampapur et al. [1994]. The paper deals with extracting features that correspond to cuts, spatial edits, and chromatic edits. The authors present an extensive formal treatment of shot boundary identification based on models of video edit effects. In our work, we rely on these methods for the initial stages of video processing, since we need to identify shot boundaries to be able to extract meaningful information within a shot.

One representation scheme of segmented video footage uses key frames [Arman et al. 1994]. The video segments can also be processed for extraction of synthetic images, or layered representational images, to represent closely the meaning of the segments. A methodology for extracting a representative image, *salient video stills*, from a sequence of images is introduced by Teodosio and Bender [1993]. The method involves determining the optical flow between successive frames, applying affine transformations calculated from the flow-warping transforms, such as rotation, translation, etc., and applying a weighted median filter to the high-resolution image data resulting in the final image. A similar method for synthesizing panoramic overviews from a sequence of frames is implemented by Teodosio and Mills [1993].

Swanberg et al. [1993] introduced a method for identifying desired objects, shots, and episodes prior to insertion in video databases. During the insertion process, the data are first analyzed with image-processing routines to identify

the key features of the data. In this model, episodes are represented using finite automata. Only video clips with inherently well defined structure can be represented. The model exploits the spatial structure of the video data without analyzing object motion. Zhang et al. [1994] presented an evaluation and a study of knowledge-guided parsing algorithms. The method has been implemented for parsing of television news, since video content parsing is possible when one has an a priori model of a video's structure.

Another system, implemented by Little et al. [1993], supports content-based retrieval and playback. They define a specific schema composed of movie, scene, and actor relations with a fixed set of attributes. Their system requires manual feature extraction. It then fits these features into the schema. Querying involves the attributes of movie, scene, and actor. Once a movie is selected, a user can browse from scene to scene beginning with the initial selection. Weiss [1994] presented an algebraic approach to content-based access to video. Video presentations are composed of video segments using a *video algebra*. The algebra contains methods for temporally and spatially combining video segments, as well as methods for navigation and querying. Media Streams is a visual language that enables users to create multilayered iconic annotations of video content [Davis 1993]. The objects denoted by icons are organized into hierarchies. The icons are used to annotate the video streams in a *Media Time Line*. The Media Time Line is the core browser and viewer of Media Streams. It enables users to visualize video at multiple time scales simultaneously, in order to read and write multilayered, iconic annotations, and it provides one consistent interface for annotation, browsing, query, and editing of video and audio data.

The work presented here follows from a number of efforts listed above. Specifically, we use low- and intermediate-level motion analysis methods similar to those offered by Allmen [1991] and others. Our object recognition ideas have been influenced by the work of Jain and his students [Gupta et al. 1991a; 1991b], Grosky [Grosky and Mehrotra 1989], and the research in image databases. Several lines of research such as those in Little et al. [1993], Swanberg et al. [1993], Zhang et al. [1994], and Weiss [1994] provided many useful ideas for the modeling aspects of our investigations. An early report of our work was presented in Dimitrova and Golshani [1994].

3. MOTION RECOVERY IN DIGITAL VIDEO

In this section we describe in detail each level of the motion analysis pipeline. At the low-level motion analysis we start with a domain of motion vectors. During intermediate-level motion analysis we extract motion trajectories that are made of motion vectors. Each trajectory can be thought of as an n-tuple of motion vectors. This trajectory representation is a basis for various other trajectory representations. At the high-level motion analysis we associate an activity to a set of trajectories of an object using domain knowledge rules.

3.1 Low-Level Motion Extraction: Single Macroblock Tracing

In MPEG, to encode a macroblock in a predicted or a bidirectional frame, we first need to find the best matching macroblock in the reference frames, then

find the amount of x and y translation (i.e., the motion vector), and finally calculate the error component [Patel et al. 1993]. The motion vector is obtained by minimizing a cost function that measures the mismatch between a block and each predictor candidate. Each bidirectional and predicted frame is an abundant source of motion information. In fact, each of these frames might be considered a crude interpolation of the optical flow. Thus, the extraction of the motion vectors of a single macroblock through a sequence of frames is similar to low-level motion analysis.

Tracing a macroblock can continue until the end of the video sequence if we do not impose a stopping criterion. We have a choice: to stop after a certain number of frames, stop after the object (macroblock) has come to rest, stop if the block comes to a certain position in the frame, stop if the macroblock gets out of the scene, or stop if the macroblock is occluded.

The algorithm for tracing the motion of a single macroblock through one frame pattern for MPEG encoding is given in Figure 2. In Dimitrova [1995], we describe object motion tracing for video databases in more detail. The algorithm takes the forward and backward motion vectors that belong to a particular macroblock and computes the macroblock's trajectory. The algorithm computes the macroblock's position in a B-frame by averaging the positions obtained from: (1) the previous block coordinates and forward motion vectors and (2) next (predicted) block coordinates and the backward motion vector. The position of a macroblock in a P-frame is computed using only block coordinates and forward motion vectors. If during the tracing procedure the initial macroblock moves completes out of its position, then we have to extract motion vectors for the new macroblock position, which implies that we are continuing by tracing the macroblock whose position coincides with the (x, y) coordinates of the initial macroblock. In the rest of this article, we will use τ to indicate the set of all possible motion vectors.

3.1.1 *Trajectory Description.* Various motion retrieval procedures have specific requirements for retrieving desired objects. These requirements depend on the characteristics of the retrieval which may be flexible to strict. The choice of trajectory representation may dictate the manner in which retrieval is conducted. Given a set of motion vectors for a macroblock, a number of mechanisms exist for trajectory representation. Below we present a sample list:

(1) *Point Representation:* A trajectory in this case is a set of points represented by the absolute or relative frame coordinates of the position of the object, say

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

where (x_i, y_i) is derived by projecting (x, y, i) onto the image plane. (x, y, i) denotes the position of an object, i.e., (x, y), at time instant i.

(2) *Curve Representation:* A parametric B-spline curve $P(u)$ can be computed that passes through each of the trajectory points (x_i, y_i) (see Farin [1990] for a detailed discussion). The first step involves generating a parameteri-

```
Given: frames F =  Fi i = 0,...,n;
motion vectors V = (fmx(i),fmy(i)),(bmx(i),bmy(i)) i= 1,n
initial block coordinates bx, by
 Initialize R = ∅,
  for i=1,..., n
    if F(i) ≠ I then
      if F(i) == P then
        if previousType == I
          cx = bx - fmx(i)/2;
          cy = by - fmy(i)/2;
          nextblockx = cx; nextblocky = cy;
        if previousType == P
          givenx = futurex;
          giveny = futurey;
          futurex = futurex - fmx(i)/2;
          futurey = futurey - fmy(i)/2;
      if F(i) == B then
        cx=((givenx-fmx(i)/2)+(futurex-bmx(i)/2))/2;
        cy=((giveny-fmy(i)/2)+(futurey-bmy(i)/2))/2;
      if block(bx,by) ∩ block(cx,cy) == ∅ then
        extract(mx(i),my(i)) for (cx,cy)
      R = R ∪ {(mx(i),my(i))}
        if F(i) is the last in a group of B frames before a P frame
          cx = futurex;
          cy = futurey;
      if block(bx,by) ∩ block(cx,cy) == ∅ then
        extract(mx(i),my(i)) for (cx,cy)
      R = R ∪ {(mx(i),my(i))}
    if F(i) == I then
      (bx,by) ← bestMatch(bx,by) in I
    if stopping criteria == true, then
        return R;
  endfor
```

Fig. 2. Algorithm for tracing the motion of a macroblock.

zation or *knot sequence* $u_1 \le u_2 \le \cdots \le u_n$. A commonly used approach employs cumulative chord lengths defined by the points (x_i, y_i). The next step involves setting up and solving a tridiagonal linear system of equations whose unknowns are the control points d_i of the B-splines $N_i(u)$. The linear system depends on the x_i, y_i, and u_i values. This linear system can be efficiently solved in $\mathcal{O}(n)$ time using standard techniques for tridiagonal matrices. The B-spline curve has the form:

$$P(u) = \sum d_i N_i(u),$$

and it satisfies the following:

(a) $P(u_i) = (x_i, y_i)$;

(b) $P(u)$ is a piecewise cubic polynomial, i.e., for $u_i \leq u \leq u_{i+1}$, $P(u)$ is a polynomial of degree less or equal to three; and
(c) the first and the second derivatives of $P(u)$ are continuous.

(3) *Chain Code Representation:* We develop a piecewise linear approximation to the trajectory using a set of orientation primitives. Given a set of discrete trajectory orientation primitives, we use a zig-zag line representation of the trajectory to generate the code. Another way of viewing this approach is derived from a neighborhood matrix with each neighbor coded to correspond to the primitives in the figure [Schalkoff 1989].

(4) *Differential Chain Code Representation:* Each segment is coded relative to the next line segment using the direction (left or right) and the length. For example, we can have a code for: right shorter–1, right equal–2, right longer–3, left shorter–4, left equal–5, left longer–6 [Schalkoff 1989]. This scheme is useful for approximate matching of object trajectories. It is a rotation-, scaling-, and translation-invariant scheme.

Figure 3 illustrates these methods used for the representation of an arbitrary movement. Figure 3(a) is an exact coordinate representation; 3(b) is a B-spline curve representation. Figure 3(c) represents the chain-coding process, and 3(d) shows the differential chain code representation of the trajectory.

Note that in the coordinate representation and B-spline and chain code representation schemes we have a way of representing zero motion, i.e., when the motion vector is a null vector. If the macroblock does not move over a certain number of frames, the point will be repeated. In the B-spline representation, the knot (i.e., the control point) will have a multiplicity greater than one. In the chain code representation, the zero motion is represented by the code "0." So, in all these representations the trajectory is not only a spatial representation of the object's motion (the path) but also a temporal characterization of the motion. By keeping track of the zero motion we are able to describe stationary objects as well.

The diversity of the trajectory representations makes the querying process more flexible. The actual method of representation does not have a significant impact on the querying process as long as modeling, representation, and querying are all done in the same fashion.

3.1.2 *Trajectory-Matching Functions.* Applications such as automated surveillance may require retrieval of either video sequences or objects contained in these sequences based on the object trajectories. For example, queries of the type "retrieve objects that have a motion trajectory whose point of origination is the main gallery door and terminate at the Juan Miro's picture on the opposite wall" may help in the identification of the person who damaged the picture.

Matching functions used for motion retrieval depend on the method employed for trajectory representation, as described below.

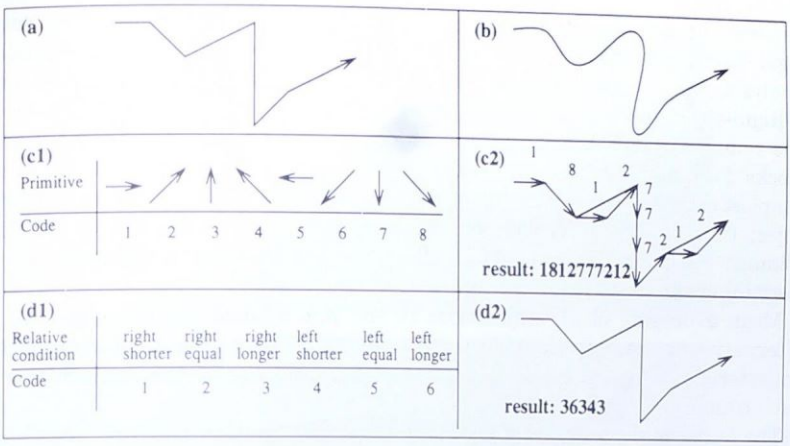


Fig. 3. Alternatives for object motion representation: (a) motion trajectory; (b) B-spline curve representation; (c1) chain-coding scheme; (c2) chain code representing the trajectory; (d1) differential chain-coding scheme; (d2) resulting differential chain code.

— Exact matching function that uses absolute frame coordinates (least-square minimization problem). This matching function has two variations:

(1) exact start position and exact trajectory match

(2) any start position and exact trajectory match.

— Exact matching function that uses relative coordinates. This function is used when the initial position of the object is not important.

— Curve comparison based on the curve-fitting approach used for interpolated trajectory representation.

— Approximate matching that uses chain code:

(1) exact start position and inexact trajectory match

(2) any start position and inexact trajectory match.

The chain code matching translates the problem of trajectory matching into a pattern-matching problem.

— Qualitative matching that uses differential chain code.

The result in each case is a similarity factor between the input trajectory and a target trajectory in the set of object trajectories.

3.2 Intermediate-Level Motion Analysis

A macroblock trajectory is the spatiotemporal representation of the macroblock's motion. These trajectories are further used for extracting object motion. This process is different for rigid and nonrigid bodies. A rigid object consists of one solid part to which motion trajectory is associated. If the object consists of several parts which themselves represent rigid objects with inde-

pendent movements, then, such a nonrigid object is represented as a set of rigid objects with their respective trajectories. At the highest level of motion analysis, we associate "activities" with the object trajectory representations.

Rigid-object motion is represented by a single trajectory. The trajectory is one common representation of the trajectories of all the component macroblocks. Finding the most-representative trajectory is not a simple task. In the simplest case we can take the trajectory of the object centroid as the reference object trajectory. A more-complicated case occurs if we decide to create a common trajectory by processing all of the macroblock trajectories or by examining only a subset of all macroblock trajectories.

Mean averaging of all trajectories of the macroblocks of the object is an alternative to choosing the object centroid's trajectory. The averaging of the trajectories in the exact form is pointwise averaging of the trajectories at each frame.

The following two assumptions make the object motion recovery feasible:

(1) *Integrity of Objects:* We assume objects are rigid or consist of rigid parts connected to each other. We do not consider situations in which objects disintegrate. This assumption is important because we only use object trajectory representation.

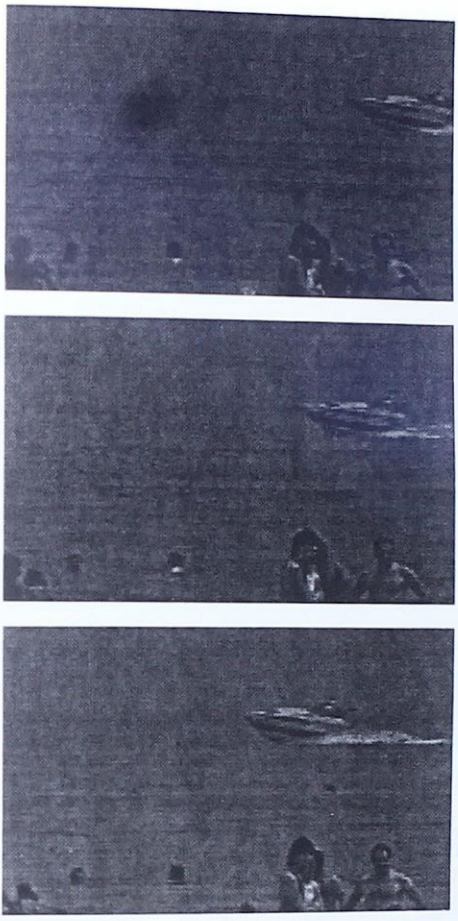
(2) *Motion Continuity:* Each macroblock under consideration has continuous motion. This assumption is important for the trajectory representation, since every trajectory segment represents continuation of the previous trajectory segment.

Averaging trajectories is used for determining a representation of a nonrigid body motion. For nonrigid objects, we must determine the number of trajectory clusters and their locations. Each cluster corresponds to a single coherent motion that represents a moving part (i.e., a rigid object). We use a hierarchical clustering algorithm (due to Duda and Hart [1973]) for determining the number of rigid object parts. Initially, the algorithm begins with clusters that contain only one trajectory each. At each subsequent step, we attempt to merge those neighboring clusters that have a similar trajectory. Individual trajectories, in this case, will be averaged to compute a trajectory for the extended cluster.

An example of a traced object through 20 encoded frames using the IBBPBBBP frame pattern is given in Figure 4. Figure 4(a) contains first, middle, and last frames of a video sequence capturing a water skiing scene. Figure 4(b) contains the motion trace for the moving yacht. The axes in Figure 4(b) correspond to the x and y axes of the video frames where the (0, 0) coordinate is at the top left corner.

Figure 5 shows six out of the 60 frames of the "Walfky" video sequence used for our next experiment. The object being traced is a small toy which performs very uneven motion. Figure 6(a) shows how the tracing of a macroblock progresses when every frame in the sequence is used. The frame pattern IBBBPBBB is used for video encoding when macroblock trajectories are extracted in Figure 6. This experiment shows that the macroblock tracing

(a)



(b)

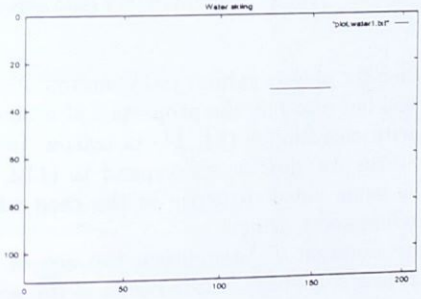


Fig. 4. Tracing a moving yacht. (a) first, middle, and last frames in video sequences; (b) motion trace of the yacht.

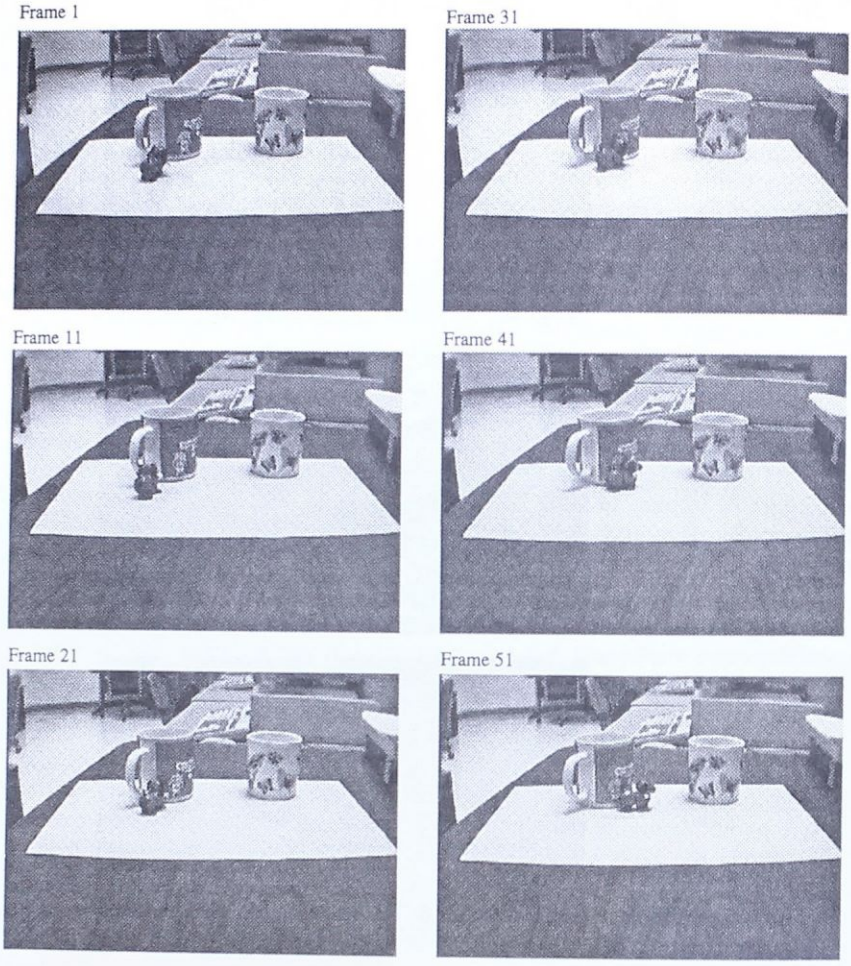


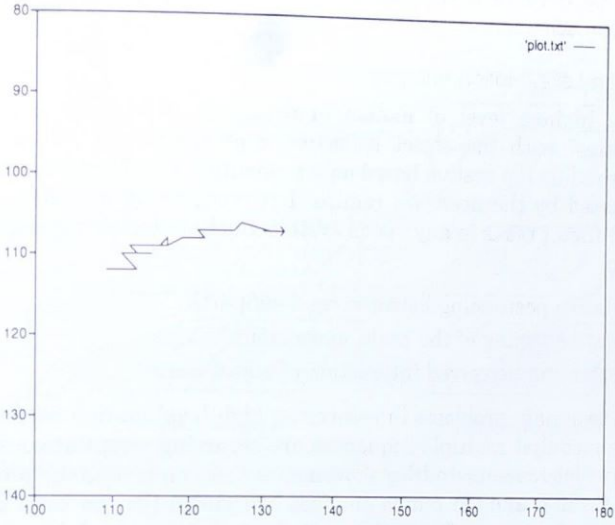
Fig. 5. Snapshots from a video sequence.

is possible when the objects exhibit jerky motion. As expected the trajectory is not only curved but also has the properties of a zig-zag line. In this case the macroblock with coordinates (14, 14) is traced. In terms of absolute frame coordinates, these coordinates correspond to (112, 112). Figure 6(b) shows tracing of the same video sequence in the case when every other frame is used for encoding and tracing.

We use the notation T to indicate the set of object trajectories. Each member of T is a sequence[1] whose range is the set of all motion vectors τ, i.e., $\forall t \in T(t : \mathcal{N} \rightarrow \tau)$, where $\mathcal{N}$ is the set of natural numbers. In other words

[1] A sequence is simply a function whose domain is the natural numbers.

(a) Trace of the macroblock (14,14) using all frames



(b) Trace of the macroblock (14,14) using every other frame

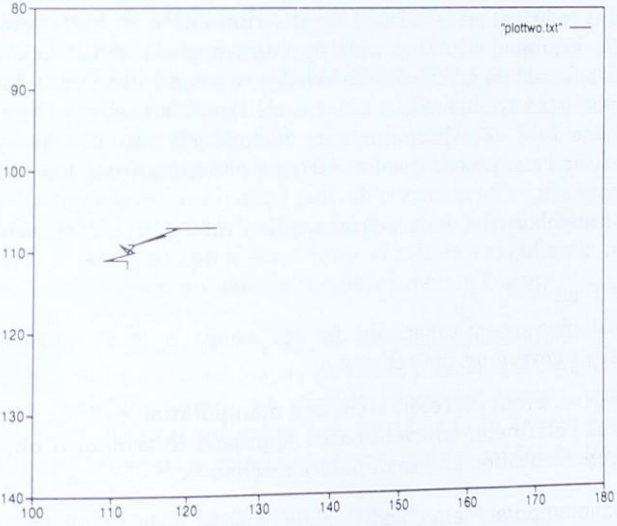


Fig. 6. Traced trajectories in the Walfky video sequence. (a) all frames; (b) only every other frame is used.

each object trajectory is a sequence of motion vectors identifying macroblock displacement for the components of the object. As discussed previously, the actual appearance of the members of T depends on the choice of the representation scheme.

3.3 High-Level Motion Analysis

At the highest level of motion analysis, we associate domain-dependent "activities" with the object trajectory representations. An activity can be recognized by the system based on a predefined set of procedures, or it can be designated by the user. We realize that recognizing activities is one of the most-difficult tasks in any vision system. Such undertaking requires information on:

(1) Relative positioning between rigid subparts
(2) Relative timing of the parts movements
(3) Actual and perceived interaction of object parts.

The two main problems in recovering high-level motion representation are (1) the fact that multiple sequences are occurring simultaneously (for example, arm movements and leg movements in human motion) and in a coordinated fashion and (2) tempo changes are global (in the case of the human body, the changes apply to all four limbs and occur slowly).

An activity involves both spatial and temporal representations of the objects of interest. We must identify the object components (shape and other features) and their respective trajectories (as we did in the previous section) at the intermediate-level motion analysis and then assemble activities. The temporal information is needed for discrimination of activities of the same type, for example, strolling, walking, hurrying, etc. After assembling object activities, based on additional knowledge, we can infer event information.

We use $\mathscr{A}$ to symbolize the set of activities. We assume the existence of a knowledge base $\mathscr{K}$ whose contents include all the necessary rules, constraints, and the procedures for deriving activities from lower-level descriptions.

Each member a of $\mathscr{A}$ is a "composition" of $t_1, t_2, \ldots, t_n$, where for every $1 < i < n$ we have:

— $t_i \in T$ and
— t_i satisfies every constraint in $\mathscr{C}_a$, where $\mathscr{C}_a \in \mathscr{K}$ represents the constraints governing the activity a.

High-level event representation and manipulation call for the use of either temporal Petri nets, an event-based approach to temporal objects, or other event representation and manipulation schemes.

3.4 Spatiotemporal Hierarchical Representation

We use a semantic multiresolution hierarchy for spatiotemporal representation (Figure 7) because it helps video analysis at various resolution levels, with coarser resolutions used for high-level event/scenario descriptions. The

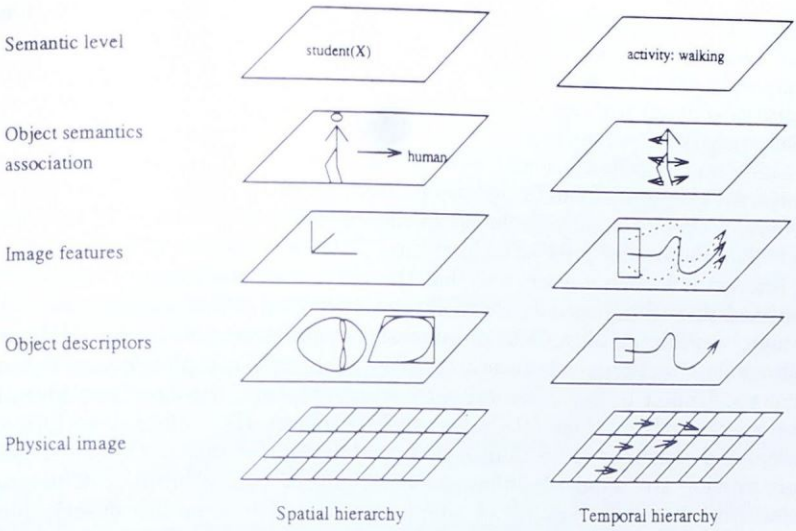


Fig. 7. Multiresolution hierarchy for spatial and temporal video representation.

advantage of multiresolution representation is that it offers a mechanism to make the trade-off between the competing demands of fine spatial/temporal resolution and low computational complexity. The idea of representational hierarchy for still images has been utilized by several image data models [Grosky and Mehrotra 1989; Gupta et al. 1991a; 1991b].

At successive time intervals, a frame is inserted at the base of the spatial hierarchy, and the features are computed for the next levels. The motion features are computed starting at the frame, and the temporal part of the hierarchy is filled with the appropriate motion descriptions. Motion analysis starts with the motion vector recovery (bottom of the temporal hierarchy, Figure 7). At the next level, individual macroblock trajectories are traced. At the intermediate level, rigid-body motion is recovered, followed by nonrigid-motion recovery. Finally, at the highest level of motion analysis, description of activities is derived from previously computed motion features (top of the temporal hierarchy, Figure 7).

The temporal part of the hierarchy can be used for various kinds of motion retrieval ranging from full-object trajectory-based matching to single-macroblock trajectory matching. Since we provide exact and inexact trajectory representation, our retrieval functions can take inputs in terms of precise spatial coordinates, orientation coordinates, and qualitative descriptions.

4. INFORMATION FILTERING AND DIGITAL VIDEO

The motion-tracing and representation scheme introduced in previous sections serves as a basis for the classification and retrieval of video sequences. Video sequences may be retrieved using the temporal (motion) part of the

hierarchy or the combination of spatial and the temporal representations. The idea of this representation is that we can compute the spatial and temporal features independently of each other. We emphasize that temporal features coupled with spatial features are important in discriminating and classifying video sequences.

Like other knowledge representation cases, we do not attempt to have a universal system that can recognize and distinguish all possible objects. Such general-purpose (i.e., domain independent) representations have been shown to be too complex for present technologies. Thus, we assume that the domain of interest is known a priori and that the video classification system will be confined to working on only those objects. Consider a domain D, called the "scope," containing all objects of interest. Formally, the elements of D are defined as object-oriented structures with potentially complex internal components. Similar to any object-oriented representation, the user can identify the objects of D by their attributes, such as object ID, image descriptions, name, and shape (or convex hull), or a combination of these. Thus, the user may provide any available information on any of the attributes of desired object (for example, object ID, or shape together with a partial description), and the system will attempt to identify the intended object. Although we do not make any assumptions on how the elements of D and their attributes are represented, we offer the following example as an indication of a typical structure.

Example 4.1. A walking human may be represented as a moving object $a_1 = (o_1, m_1, v_1)$ where

$$o_1 = (category : human, convexHull : o_6 : skeleton : o_7, \\ parts : \{head : o_2, torso : o_3\}),$$

$$m_1 = (trajectory : 2467332, activity : walking), \text{ and}$$

$$v_1 = (v\# : 234, firstFrame : 45, lastFrame : 485).$$

Similarly, the head and torso also have their spatial and motion descriptions.

In a database containing only "still" images, a correspondence table of the form (O, I)—where O stands for the object, and I stands for the image—will suffice. In a video database, we have the added parameter of temporal changes. Although the motion of each object can be modeled as an attribute of the object, say, "dog, big, brown, running," it is more appropriate to separate objects and their motions as two different parameters. Note that if motion is considered as just another attribute of the object, then in case the same object appears more than once, each time with a different motion, we would need multiple, different entries into the database. For example, there would be multiple entries for the big brown dog: running right, running left, running in circles, and jumping.

Video sequences are identified by objects present in the scene and their respective motion. The goal of the motion analysis is to extract activity and event representation. An index entry of an activity in a video sequence has

the following form:

$$(O, M, V) = (objectRep, motionRep, vid)$$

— *objectRep:* an object represented by its extracted features (convex hull, object skeleton, centroid, texture, set of macroblocks covering the object, etc.; see left side of Figure 7). An object representation might include a set of object representations of the constituting parts (e.g., objects that represent a human figure include head, torso, arms, and legs object representations).

— *motionRep:* an object trajectory specified by objectRep, velocity, trajectory curvature, torsion, and activity description (see right side of Figure 7).

— *vid:* identity of the video subsequence to which an object belongs to: *vid* consists of (*viSeqId*, *firstFrame*, *lastFrame*).

(1) *viSeqId* is a video sequence identity which is unique for a sequence across the whole video database.
(2) *firstFrame:* the first frame in which the specified object appears.
(3) *lastFrame:* the last frame in which the specified object appears.

4.1 Content-Filtering Operators

The OMV triplet is the basis for the query functions. There are many possibilities for the selection of filters (in this context, query functions.) A sample selection is presented below. These operators may be used in a relational form, mostly in a table lookup mode, or may be embedded into a more-elaborate query language, as presented in Section 4.2. Recall that $\mathcal{P}$(A) is used to denote the powerset of the set A, i.e., the set of all subsets of A.

$$V\_Seq : O \times M \rightarrow \mathcal{P}(V)$$

This function takes any description that can be provided at any level of the spatial hierarchy. The input might be a characterization of the object in terms of its bounding polygon, stick figure, a name, or concept. At this point we need to emphasize that we use the properties of the object-oriented nature of the representation of the objects. For example, the expression

$$V\_Seq(O\_category = pet, (Activity = walking, Trajectory = t_1))$$

translates into "retrieve all the video sequences in which a pet walks and makes a trajectory t_1." The answer will include all the objects (animals) that are classified as pets: cats, dogs, fish called Wanda, etc. It is important to note that here we discuss only the formal framework in an informal way and that these functions are implemented within an interactive window-based graphical query interface which we discuss in Section 5.1.

The function

$$Object\_motion : O \times V \rightarrow \mathcal{P}(M)$$

takes any object description and a particular video sequence and returns a set of motion descriptions related to that object. In order to detect which

objects performed a particular type of motion, i.e., the agent of the action, we use a function of the following family:

$$Agents : M \times V \rightarrow \mathcal{P}(O).$$

The next function is used to get a detailed description of all the objects and their respective motions in a video sequence:

$$Describe\_Video : V \rightarrow \mathcal{P}(O \times M).$$

If we just want information about the spatial characteristics of objects in a sequence, we use the function

$$Objects : V \rightarrow \mathcal{P}(O).$$

This is equivalent to the "agents" function where the first argument is unimportant. Thus, given a $v_i \in V$, $Objects(v_i) = Agents(any, v_i)$.

The above functions allow for inexactness, and by default, they return results that are approximately similar to the precise answer. To make the operator exact, we use a higher-order operator that converts the query function in the desired manner, in this case, makes it exact. There are several types of these operators, e.g., exact, partial, and similar.

For making the retrieval exact, the symbol ! is placed in front of the query function. For example, "!*Agents*" returns only objects that have exactly the same motion description as the one given in M. Similarly, "!*Object_motion*" returns only motion descriptions of objects whose spatial characteristics (for example, exact bounding polygon, texture) match the spatial characteristics of a given object.

The # symbol placed in front of the query function is used for partial retrieval. Partial retrieval means that any of the motion or temporal characteristics of the given object should match. For example, "*#Agents*" will return all objects that match at least one of the motion descriptors.

4.2 The Query Language

The retrieval functions introduced in the previous section are embedded into the framework of a multimedia functional query language called EVA, described in Golshani and Dimitrova [1994]. EVA is the interface to a multimedia database system capable of storage, retrieval, management, analysis, and delivery of objects of various media types, including text, audio, images, and moving pictures. The language deals with the temporal and spatial aspects of multimedia information retrieval and delivery, in addition to the usual capabilities provided by the ordinary database languages. EVA has five groups of operators, namely: operations for querying and updating (i.e., editing) the multimedia information, operations for screen management, temporal operators, operators for specifying rules and constraints, and aggregation (computational) operators. EVA is an extension of a functional query language whose notation is based on that of conventional set theory. Both the original language and its extensions are formally defined in an algebraic framework. EVA is object oriented and supports objects, object classes, at-

tributes and methods of objects, and relationships between objects. It has been ported onto several different platforms.

EVA provides a wide collection of operators that deal with text, graphics, scanned images, audio, and video. In addition, there are numerous type-independent operators such as set operators like union and set membership. One of the general operations is: the set construction operator. Generally, this has the form $\{f(x) \mid P(x)\}$, where $f(x)$ denotes the desired output objects, and $P(x)$ denotes the retrieval predicate which has to be true for those objects.

— *set operation symbols:* isin, isSubsetOf, isTrueSubsetOf, union, intersection, difference, Union, Intersection, noOf.

— *equality operators:* is, isnot.

— *temporal synchronization* (*for all media types*): sim, before, meets, equals, starts, at, finishes.

— *spatial composition* (*applied only to graphics, images, and video*): left, right, bottom, up, showIn, arrange.

— media-dependent operation symbols include

  —*text:* appendPar, cutPar, eqPar, keyword, isKeywordIn, parSim.

  —*graphics:* insPatch, pictureSum, fill, domain, colors, getPatch, getColor, restriction, scale, translate, dot, lineSeg, box, coincident, contains, disjoint, visible, bounded.

  —*images:* shift, zoom, superimpose, overlay, imageSim.

  —*audio:* intensity, extract, audioIns, audioLen, audioSim.

  —*video:* videoLen, pace, videoClip, videoIns.

—*integer operation symbols:* +, −, *, <, >, <=, >=, min, max, ave, sum, prod.

—*string operation symbols:* concat, strLen.

—*logical operation symbols:* and, or, implies, not.

To demonstrate the capabilities of EVA and how queries are constructed, we present a simplified example. The first part of the example will demonstrate the language without the OMV extensions. The video content retrieval extensions will be discussed once the appropriate distinctions are made.

We present the schema of a multimedia database system and then provide a few sample queries. The schema is represented as a graph whose nodes are object classes (in algebraic terms, sorts) and whose arcs are the relationships between object classes (represented as functions). Readers familiar with the algebraic framework would recognize this as a many-sorted algebra.

Illustrated in Figure 8, the schema models a college basketball multimedia database. The *basic types* in this system are String, Integer, Audio, Text, Video, while Player and School are *user-defined data types*. The highlighted portion appearing in dotted lines relates to the extension for video content retrieval described in Section 4.3.

The main difference between these basic and user-defined types is that the former constitute the application-independent constituents of any schema, whereas the user-defined types depend on each individual application. In

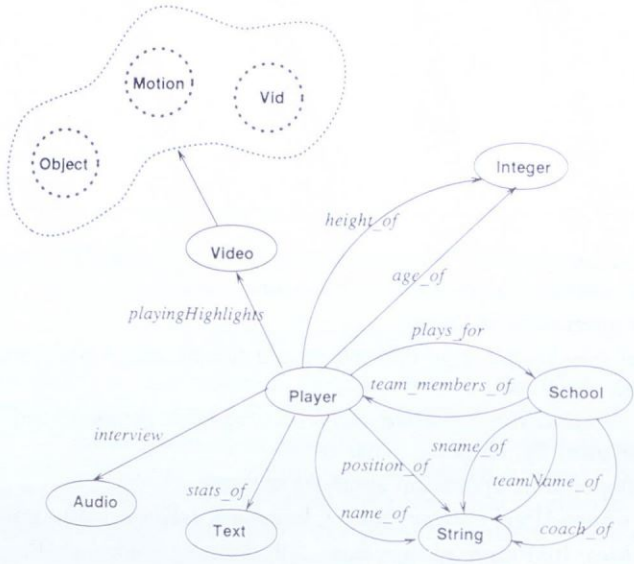


Fig. 8. Basketball schema.

Figure 8, all object types, both basic and user defined, appear in ovals. The attributes of objects and their relationships that are captured by arrows are the following functions:

$$
\begin{array}{ll}
name\_of & : Player \rightarrow String \\
position\_of & : Player \rightarrow String \\
state\_of & : Player \rightarrow Text \\
interview & : Player \rightarrow Audio \\
playingHighlights & : Player \rightarrow Video \\
height\_of & : Player \rightarrow Integer \\
age\_of & : Player \rightarrow Integer \\
plays\_for & : Player \rightarrow School \\
teammembers\_of & : School \rightarrow \mathscr{P}(School) \\
sname\_of & : School \rightarrow String \\
teamName\_of & : School \rightarrow String \\
coach\_of & : School \rightarrow String
\end{array}
$$

Here are some queries on this database.

(1) List all guards who are taller than 190cm.

$$\{name\_of(P) \mid position\_of(P) is \text{“}Guard\text{”} \ and \ height\_of(P) > 190\}$$

This is a simple query on the nonmultimedia portion of the database. P is a variable of type *Player*. The result is a list of names of players who satisfy the given conditions.

(2) Play video clips of all centers, and simultaneously display their statistics.

$$\{(name\_of(P), stats\_of(P))sim\ playingHighlights\_of(P)|\ position\_of(P)is\text{“}Center\text{”}\}$$

While variable P ranges over the elements of type *Player*, whenever the condition on position is satisfied, the name, statistics, and the corresponding video clip of the qualified player are displayed. "sim," standing for "simultaneously," is one of the synchronization operators that ensure proper semantics for presentations.

(3) Display the statistics of all Phoenix State University guards, and show their highlights before playing their interviews.

$$\{(name\_of(P), stats\_of(P))sim(\ playingHighlights\_of(P)\ beforeinterview(P))|\ plays\_for(P)is\text{“}PhoenixStateUniversity\text{”}and\ position\_of\ (P)is\text{“}Guard\text{”}\}$$

The result of this query is that, for every guard of the appropriate school, while their name and statistics are displayed, their video clip is presented first, and then their respective interview is played. The term "before" is another synchronization operator.

4.3 Querying Video Contents

Note that in the above queries, we treated *Video*, *Audio*, and *Text* as basic types in a similar manner to the type Integer, i.e., as objects whose contents can be displayed or presented, but no further specific characteristics are known about the contents. Our motion recovery algorithm and specifically the OMV functions enable us to treat *Video* in a different way, as described below.

Grosky's [1994] categorization makes a distinction between the physical basic data types and the conceptual data types. He adopts a generic model to represent *content-independent* and *content-based* properties of multimedia objects. Content-independent properties are related to the physical data object itself (uninterpreted data) as well as synchronization and storage information. Content-based properties refer to relationships between nonmultimedia real-world application entities and multimedia objects. The content-based properties associate semantics to the object at various levels.

A binary object containing the video stream that corresponds to the playing highlights of a particular player is an instance of physical data type. The extracted spatial and motion characteristics are stored in the conceptual data type. The queries on the content of the video data are directed to the conceptual video data type.

The *conceptual video data type* is molded from the spatiotemporal hierarchy presented in Figure 7 using the object-motion-video structures. The OMV retrieval functions augment EVA's retrieval capabilities since they turn the physical object Video into a conceptual one, i.e., an object with its own specific

set of properties that can be incorporated into queries for more-precise questions. The extension to the schema which enhances the video type to be a conceptual type appears in the dotted line in Figure 8. Below is a list of operators that augment the retrieval capabilities based on the OMV retrieval functions:

— *Function Composition:* given functions $f: X \rightarrow Y$ and $g: Y \rightarrow Z$, the composition is $f \circ g(x) = g(f(x))$. For *example*, Given an object (any characteristic) in a video sequence v_1, retrieve objects in another video sequence v_2 which have similar motion.

$$Agents(Object\_motion(o_1, v_1), v_2)$$

— *Temporal Combination Functions:* $f\theta g$ where $\theta \in \{before, meets, simultaneously, starts, finishes\}$. Although the same syntax is used, these should not be mistaken for the synchronization operators. In this case, no confusion is expected since the context would determine the designation of the operator. An example of usage for this type of operator is the query "retrieve all the sequences in which a tall person is waving while the president walks."

— *Spatial Combination Functions:* $f\xi g$ where $\xi \in \{next, behind, inFront, left, right\}$.

Using the above combinators and the OMV structure, many new types of queries that refer to the contents of video sequences can be specified. Specifically, we can express queries that refer to the contents of video sequences. Examples include the following:

(1) "Retrieve all the video sequences with the longest successful shots." This query translates into "retrieve all the video sequences for which the length of the trajectory of the ball is maximum."
(2) "Spell out all the details of movements of the players whose height is greater than 200cm." This query is good for analyzing the pattern in which certain players move and achieve the score.
(3) "Find the video sequences in which the player is wearing a blue shirt." The "blue shirt" is inferred using image analysis.

The target language is a visual one that allows for inclusion of spatial properties (sketches) and exact and inexact images. The notation presented in this article is the basis for the visual query interface [Dimitrova 1995; Michael 1994].

5. AN ARCHITECTURE FOR VIDEO CLASSIFICATION AND RETRIEVAL

In the previous sections we introduced a model for video classification which exploits motion recovery and representation. In this section, we discuss a general architecture for video database retrieval based on the model. The proposed architecture, as presented in Figure 9, consists of:

— Insertion module

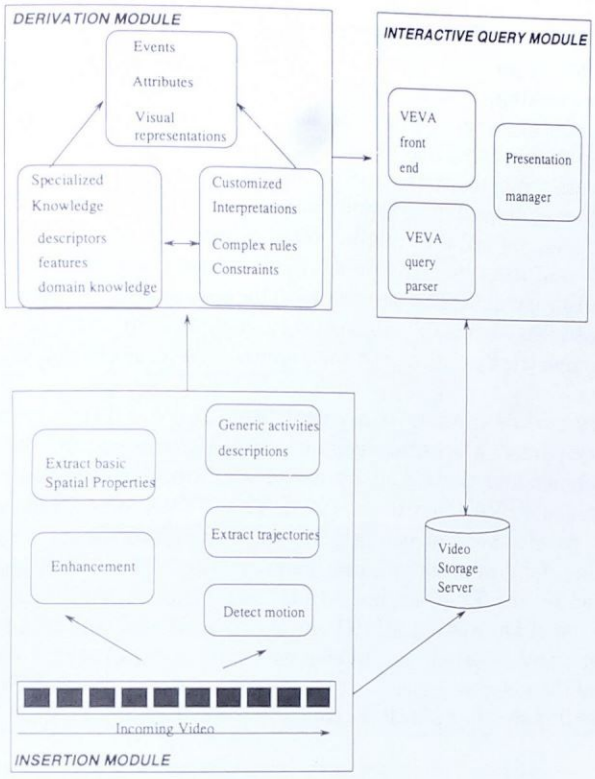


Fig. 9. An architecture for video classification and retrieval.

— Derivation module
— Interactive query module
— Video storage server.

The insertion module is responsible for initial analysis of the incoming video signal. It consists of a suite of operators for image enhancement, operators for the extraction of basic spatial properties, and operators for motion detection and the extraction of motion trajectories. With respect to the spatiotemporal hierarchy, this module is an implementation of the operators between the lowest level of the hierarchy (raw physical data) to the intermediate representation. Currently, the functionalities for spatial analysis are supplied by the Khoros computer vision environment [Rasure et al. 1990]. The extraction of image features, finding regions, and thinning operators are performed by calls to Khoros functions. Although features are automatically extracted, the process of feature selection is manual. For example, we can apply an operator for image segmentation and find the regions in a video frame. However, the selection of regions of importance is decided by the

application designer. The automation of this whole process is possible for strictly limited application domains such as industrial monitoring, domain-specific video editing, camera surveillance, and others. The motion detection and tracing operators are also part of the insertion module. The implementation of the motion-tracing algorithm is given in Section 5.2.

The derivation module consists of operators for translation of the extracted features into meaningful descriptions for retrieval. Each application typically defines its own set of meaningful entities and events and has its own interpretation of the same. In our video model and language, the extracted properties are represented by predicates. The derivation module provides the mapping between the visual properties extracted form the video sequences which are geometric by nature and the algebraic representation which is used for querying.

The query module consists of a visual front-end for query composition, a visual query parser, a schema designer, and a presentation manager. The schema designer and the visual front-end are incorporated into the visual query language VEVA [Dimitrova 1995]. The VEVA prototype serves as a testbed for development of new algorithms for video/image segmentation, video parsing, feature selection, and classification. The prototype has been implemented in Tcl/Tk [Osterhout 1994] with the added image and video widgets on top of an existing MPEG encoder [Rowe and Smith 1992].

The video storage server is envisioned to be a disk array serving as a repository of the video sequences. At this point we use a simple file system for storing a limited number of MPEG compressed video sequences.

5.1 The Visual Query Language VEVA

Spatial and motion characteristics of objects, derived from images and video sequences respectively, are inherently visual. In this section, we outline the design of a multimedia database language which has well-defined semantics in both character-based and icon-based paradigms.

Defined within the algebraic framework described above, VEVA is a visual query language that provides all the necessary constructs for retrieval and management of multimedia information. The basis for the language is a schema (algebraic signature) which contains entity types (both user-defined and application-independent types) and the associated operators [Golshani and Dimitrova 1994]. By using these operators, the user can visually specify a query for the desired objects in a simple way. VEVA has a formal grammar with which the set of acceptable expressions can be generated. The grammar for the visual language VEVA is given using visual rules in the style of a picture description language which was developed within the syntactic approach to pattern recognition [Schalkoff 1989]. The grammar rules contain nonterminal and terminal icons. The rules are given as graph-rewriting rules where the left-hand side is a nonterminal icon, and the right-hand side is a graph containing nonterminal and terminal icons connected with customized links.

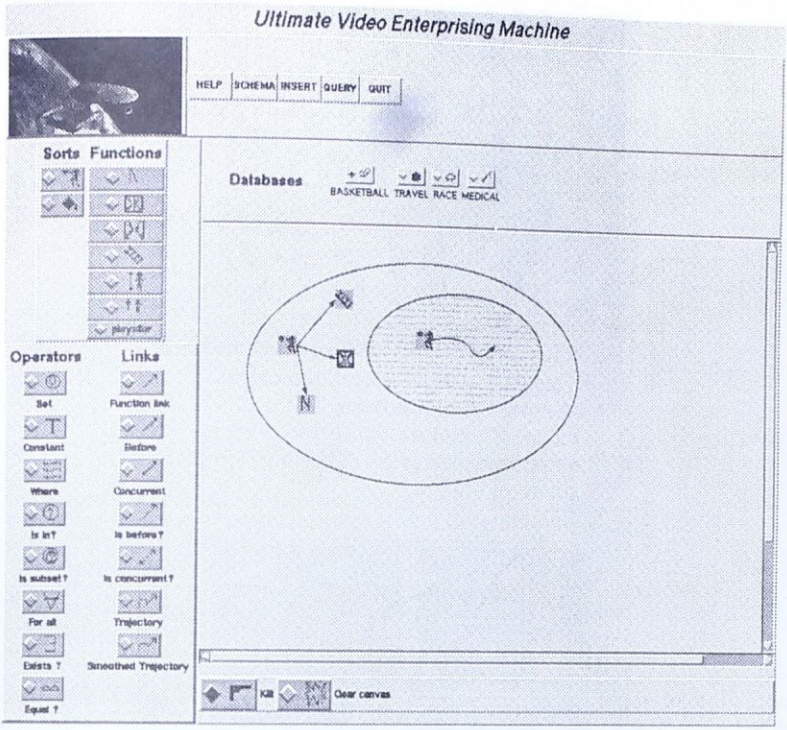


Fig. 10. Visual query involving a trajectory description.

Parsing of visual expressions in VEVA is a process of determining the structure of the workspace. Note that parsing is the first step of the VEVA language processing, because lexical analysis is not necessary. All available icon symbols can be drawn from the given pallette and connected by a set of permissible links. Thus, every expression that is drawn is lexically correct. The execution process begins by parsing the contents of the VEVA workspace. The algorithm finds the top-level set expressions which may contain other set expressions. Translated into visual terms, this algorithm finds the enclosed visual expressions or other iconic elements within a given oval. The algorithm calls the set evaluation procedure recursively for the sets that are contained in it, until there are single sets with simple function-predicate expressions left. The evaluated sets can be connected with temporal links which prescribe the order in which the resulting objects should be presented by the presentation manager. If the evaluated expression contains temporal links, then the parsed execution order is delivered to the presentation manager.

An example query is given in Figure 10. As we stated earlier, VEVA allows for visual queries in which we can specify the path of a moving object. In this example the input trajectory for the player is given as a smoothed trajectory. The visual query given in Figure 10 will select those video sequences from the

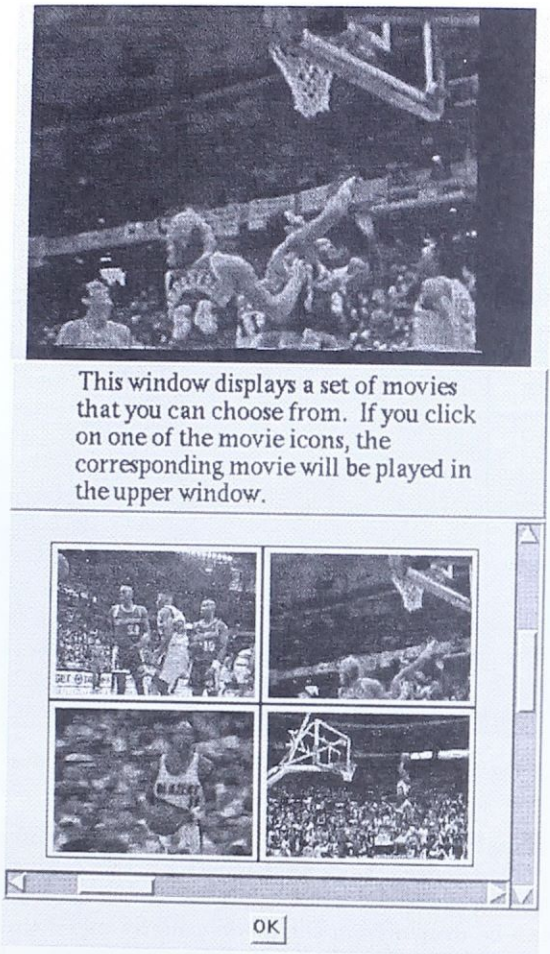


Fig. 11. Results from the visual query. Courtesy of NBA Entertainment.

repository in which the player's trajectory is similar to the one drawn by the user and display the name and the position of the player. The result of the query is shown in Figure 11. The user can browse and play the selected video segments.

Various models have been proposed for temporal synchronization, composition, and presentation in multimedia applications, for example, Buchanan and Zellweger [1993] and Little et al. [1991]. On the other hand, a number of models for content-based access of digital video has been proposed [Arman et al. 1994; Bobick 1993; Rowe et al. 1994; Swanberg et al. 1993; Zhang et al. 1994]. However, a general formal model and a language for content representation, composition, and querying of digital video based on the temporal and the spatial properties of objects found in the video sequences has not been

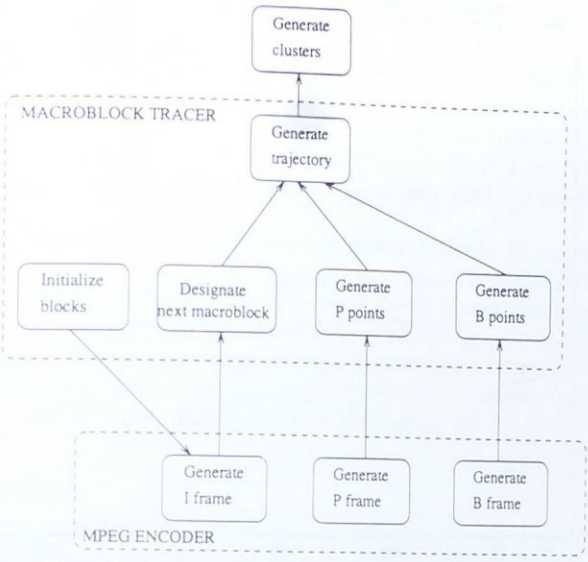


Fig. 12. Macroblock motion extraction.

offered yet. Our video model and language VEVA attempts to unify the presentation aspects as well as content representation aspects of multimedia objects.

5.2 Implementation of Macroblock Tracing

The motion-tracing algorithm is a part of the derivation module in our video classification architecture. We have tested our ideas by implementing the motion-tracing and extraction algorithm under Solaris 2.3 using the MPEG encoder produced by the Digital Video research team at the University of California, Berkeley. A functional view of the MPEG-based motion extraction is given in Figure 12. We have introduced functions for extraction of motion vectors during the generation of P- and B-frames. We use the motion-tracing algorithm to compute the macroblock trajectories.

The performance results are shown in Figure 13. We have tested our motion-tracing algorithm by ranging the number of macroblocks being traced from zero to all macroblocks. The input video sequence is the standard table tennis sequence, which consists of 10 frames, each of size 352 by 240 pixels. This sequence is a good performance test case, because it has background and foreground motion. The encoding frame pattern is IBBBPBBBBP. This means that all the input frames are used for video encoding. If only encoding is performed without any motion-tracing algorithm, the total elapsed time is 32.6 seconds (+/− 0.05 seconds). With the motion-tracing algorithm, the time increase is evident with the increase of the number of blocks. Starting

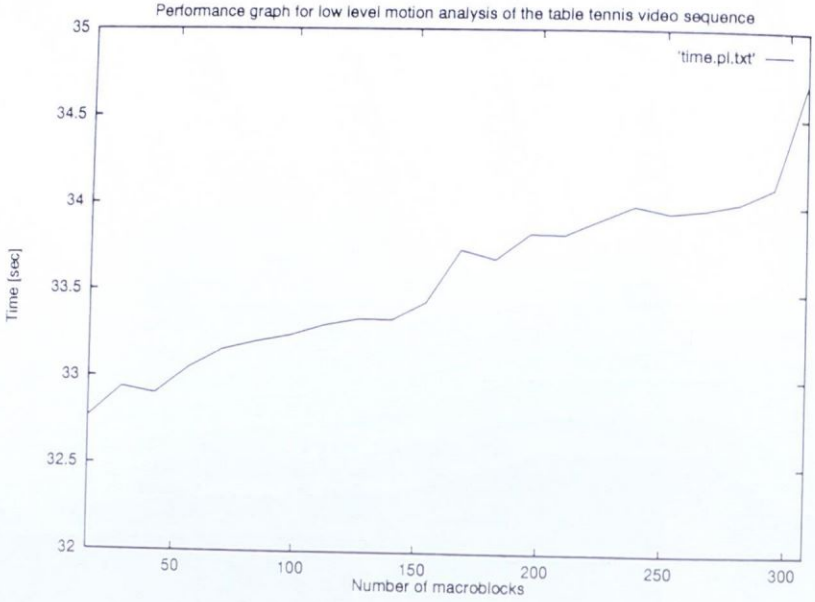


Fig. 13. Performance of object motion tracking.

with one macroblock, we get elapsed time of 32.76 seconds for encoding and tracing which is 0.16 seconds more than the previous case. As shown in Figure 13, when the number of traced macroblocks increases up to 300, the elapsed time goes up to 34.72 seconds. This shows that even if we keep track of the motion of all the macroblocks we have a time increase of 6%.

The gain in MPEG compression is mostly achieved by exploiting temporal redundancy. MPEG avoids coding the same block twice by storing/sending over the displacement vector from the previous image. Thus, the basic assumption is that the frame pattern used for MPEG compression is going to contain P- and B-frames.

Our algorithm for motion tracing would have very limited application if the stream to be encoded is using only I-frames. In that case, there the motion algorithm cannot find any motion vectors to take advantage of. If high quality of encoded video is crucial to the application at hand, then the algorithm has to be rewritten, so that motion estimation is performed using some imaginary frame pattern which would not have any impact on the encoded video stream. Then the motion-tracing algorithm would be performed on the obtained motion estimates. In this case, the motion information that is obtained from the encoder is in the forward vectors of the P-frames only. From the P-frame to the next I-frame we do not have any motion information. We have several choices:

(a) We can make a prediction for the motion vector between the P-frame and the next I-frame. This prediction is a guess that we can use the same

motion vector as the vector for the P-frame. This solution does not introduce additional overhead. The problem is that it relies not only on the assumption for the continuity of motion but also assumes that the motion is constant.

(b) We can perform the actual search and compute the motion vector for the blocks from the P- to the next I-frame. This means that we will be adding much more compute cycles than it is necessary for the encoding process.

We also need much more complicated motion models to recover the true motion of the objects in the case of complicated camera motion. For example, when we have the camera focus on a moving object, then the object appears to be stationary. The motion of the object is implied by the macroblock vectors of the background. More-sophisticated relative-motion detection algorithms are needed. This work is part of our ongoing SunSet Multimedia Information System project [Golshani and Dimitrova 1994; Michael 1994].

6. CONCLUSIONS

From the point of view of video retrieval, the video technology has not seen much progress from the days when film editors examined each and every frame by hand in order to find the exact place of each cut. In fact, despite the introduction of many video editing systems such as VideoShop and Adobe's Premiere, much of retrieval is done by either time pointers (e.g., the frame counter), visual proxies, or various types of graphical or descriptive pointers. What is clearly missing from the video technology is the ability to locate and retrieve video clips that contain an object with specific characteristics, particularly with respect to movements. Video databases can be useful to many application areas such as education, business, medicine, and more prominently, entertainment. As such, the value of better and more-equipped video systems are becoming clearer. While many aspects of video systems, such as presentation editing tools, have seen significant improvement, our progress on content-based retrieval has not been as forthcoming.

We believe that our attempts to address the above needs must start with a modeling mechanism that allows for the representation of semantic knowledge from both spatial and temporal features of the objects in video sequences. Computing high-level motion description can be done independently of recognizing objects [Allmen 1991]. We elaborate on this property by showing that the recovery of object trajectories can be performed without prior knowledge of objects undergoing motion. The goal is to have both: independent retrieval along the temporal and the spatial hierarchies as well as retrieval of combined features from the spatial and the temporal hierarchies. We treat motion vectors extracted during the motion compensation phase of video encoding as coarse-level optical flow that is further used for intermediate- and high-level motion description. Motion information extraction is then carried out at low level by motion vector detection, at the intermediate level by motion tracing, and the high level by associating an object and a set of trajectories with recognizable activities.

In our object motion representations, we provide various levels of precision of trajectory representation. Retrieval functions based on these representations offer a wide spectrum of approximation in the process of matching. We need to relate the motion at a higher level of abstraction of the object to the detailed motion of parts of objects. Events can be represented in a form that is common in the image-understanding and interpretation area: predicates, temporal networks, etc.

ACKNOWLEDGMENTS

The authors gratefully acknowledge assistance from the guest editors. We would like to thank Tom Foley for clarifying our ideas on B-spline representation. We would like to thank Guru Banavar for the numerous discussions and insightful comments on the visual language. The authors would also like to thank K. P. Lee for suggesting Tcl/Tk as the implementation platform for the interface.

REFERENCES

ALLMEN, M. C. 1991. Image sequence description using spatiotemporal flow curves: Toward motion-based recognition. Ph.D. thesis, Univ. of Wisconsin, Madison, Wisc.

ARMAN, F., DEPOMMIER, R. HSU, A., AND CHIU, M.-Y. 1994. Content-based browsing of video sequences. In *Proceedings of ACM Multimedia '94* (San Francisco, Calif.). ACM Press, New York, 97–103.

BOBICK, A. F. 1993. Representational frames in video annotation. In *Proceedings of the 27th Annual Conference on Signals, Systems and Computers*. IEEE Computer Society Press, Los Alamitos, Calif.

BUCHANAN, M. C. AND ZELLWEGER, P. T. 1993. Automatically generating consistent schedules for multimedia documents. *Multimedia Syst. J. 1*, 2 (Sept.).

DAVIS, M. 1993. Media streams: An iconic visual language for video annotation. In *Proceedings of the IEEE Symposium on Visual Languages* (Bergen, Norway). IEEE, New York, 196–202.

DIMITROVA, N. 1995. Content classification and retrieval of digital video based on motion recovery. Ph.D. thesis, Arizona State Univ., Tempe, Ariz.

DIMITROVA, N. AND GOLSHANI, F. 1994. Rx for semantic video database retrieval. In *Proceedings of ACM Multimedia '94* (San Francisco, Calif.). ACM Press, New York, 219–226.

DUDA, R. O. AND HART, P. E. 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.

FARIN, G. 1990. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, New York.

FURHT, B. 1994. Multimedia systems: An overview. *IEEE Multimedia 1*, 1 (Spring), 47–59.

GODDARD, N. 1992. The perception of articulated motion: Recognizing moving light displays. Ph.D. thesis, Univ. of Rochester, Rochester, N.Y.

GOLSHANI, F. AND DIMITROVA, N. 1994. Retrieval and delivery of information in multimedia database systems. *Inf. Softw. Tech. 36*, 4 (May), 235–242.

GROSKY, W. AND MEHROTRA, R. 1989. Image database management. *IEEE Comput. 22*, 12 (Dec.), 7–8.

GROSKY, W. I. 1994. Multimedia information systems. *IEEE Multimedia 1*, 1 (Spring), 12–24.

GUPTA, A., WEYMOUTH, T., AND JAIN, R. 1991a. Semantic queries in image databases. In *Visual Database Systems II*, E. Knuth and L. Wegner, Eds. Elsevier Science Publishers (North-Holland), Amsterdam, 201–215.

GUPTA, A., WEYMOUTH, T., AND JAIN, R. 1991b. Semantic queries with pictures: The VIMSYS Model. In the *Conference on Very Large Data Bases*. VLDB Endowment, Saratoga, Calif.

HAMPAPUR, A., WEYMOUTH, T., AND JAIN, R. 1994. Digital video segmentation. In *Proceedings of ACM Multimedia '94* (San Francisco, Calif.). ACM Press, New York, 357–364.

HORN, B. K. AND SCHUNCK, B. G. 1981. Determining optical flow. *Artif. Intell. 17*, 1–3, 185–203.

JOHANSSON, G. 1976. Spatio-temporal differentiation and integration in visual motion perception. *Psychol. Res. 38*, 4, 379–393.

KOLLER, D., WEBER, J., AND MALIK, J. 1993. Robust multiple car tracking with occlusion reasoning. Tech. Rep. CSD-93-780, EECS Dept., Univ. of California, Berkeley, Calif. Nov.

KUBOTA, H., OKAMOTO, Y., MIZOGUCHI, H., AND KUNO, Y. 1993. Vision processor system for moving-object analysis. *Mach. Vis. Appl. 7*, 1, 37–43.

LEGALL, D. 1991. MPEG: A video compression standard for multimedia applications. *Commun. ACM 34*, 4 (Apr.), 46–58.

LITTLE, T., AHANGER, G., FOLZ, R., GIBBON, J., REEVE, F., SCHELLENG, D., AND VENKATESH, D. 1993. A digital on-demand video service supporting content-based queries. In *Proceedings of ACM Multimedia '93* (Anaheim, Calif.). ACM Press, New York, 427–436.

LITTLE, T., GHAFOOR, A., CHEN, C., CHANG, C., AND BERRA, P. 1991. Multimedia synchronization. *Data Eng. 14*, 3, 26–35.

MATTISON, P. E. 1994. *Practical Digital Video with Programming Examples in C.* John Wiley and Sons, New York.

MICHAEL, N. 1994. VEENA—a visual query language. M.S. thesis, Arizona State Univ., Tempe, Ariz.

NAGASAKA, A. AND TANAKA, Y. 1992. Automatic video indexing and full-video search for object appearances. In *Visual Database Systems II*, E. Knuth and L. Wegner, Eds. Elsevier Science Publishers (North-Holland), Amsterdam, 113–127.

OSTERHOUT, J. K. 1994. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, Mass.

OTSUJI, K. AND TONOMURA, Y. 1993. Projection detection filter for video cut detection. In *Proceedings of ACM Multimedia '93* (Anaheim, Calif.). ACM Press, New York, 251–257.

PATEL, K., SMITH, B. C., AND ROWE, L. A. 1993. Performance of a software MPEG video decoder. In *Proceedings of ACM Multimedia '93* (Anaheim, Calif.). ACM Press, New York, 75–82.

RASURE, J., ARGIRO, D., SAUER, T., AND WILLIAMS, C. 1990. Visual language and software development environment for image processing. *Int. J. Imaging Syst. Tech. 2*, 2, 183–199.

ROHR, K. 1994. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding 59*, 1 (Jan.), 94–115.

ROWE, L. A. AND SMITH, B. C. 1992. A continuous media player. In the *3rd International Workshop on Network and OS Support for Digital Audio and Video.* Springer-Verlag, Berlin, 334–344.

ROWE, L. A., BORECZKY, J. S., AND EADS, C. A. 1994. Indexes for user access to large video databases. In *Proceedings of SPIE IS and Symposium on Storage and Retrieval for Image and Video Databases* (San Jose, Calif.). SPIE.

SCHALKOFF, R. J. 1989. *Digital Image Processing and Computer Vision.* John Wiley and Sons, New York.

SWANBERG, D., SHU, C.-F., AND JAIN, R. 1993. Knowledge guided parsing in video databases. In *Image and Video Processing Conference; Symposium on Electronic Imaging: Science and Technology.* Vol. 1908. IS & T/SPIE, 13–24.

TEODOSIO, L. AND BENDER, W. 1993. Sallient video stills: Content and context preserved. In *Proceedings of ACM Multimedia '93* (Anaheim, Calif.). ACM Press, New York.

TEODOSIO, L. AND MILLS, M. 1993. Panoramic overviews for navigating real-world scenes. In *Proceedings of ACM Multimedia '93* (Anaheim, Calif.). ACM Press, New York.

WEISS, R. 1994. Content-based access to algebraic video. Tech. Rep., Massachusetts Inst. of Technology, Cambridge, Mass.

ZHANG, H., GONG, Y., SMOLIAR, S., AND TAN, S. Y. 1994. Automatic parsing of news video. In *Proceedings of the International Conference on Multimedia Computing and Systems* (Boston, Mass.). IEEE Computer Society Press, Los Alamitos, Calif., 45–54.

Received August 1994; revised February 1995; accepted July 1995