9-1-1998

Al Hybrid Content-Based Retrieval Approach For Video Data

Serhan Dagtas
Purdue University School of Electrical and Computer Engineering

Wasfi Al-Kliatib
Purdue University School of Electrical and Computer Engineering

Ashfaq Khokhar
Purdue University School of Electrical and Computer Engineering

Arif Ghafoor
Purdue University School of Electrical and Computer Engineering

A HYBRID CONTENT-BASED RETRIEVAL APROACH FOR VIDEO DATA

SERHAN DAĞTAŞ
WASFI AL-KHATIB
ASHFAQ KHOKHAR
ARIF GHAFOOR

TR-ECE 98-13
SEPTEMBER 1998

SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

A Hybrid Content-Based Retrieval Approach For Video Data[1]

Serhan Dağtaş, Wasfi Al-Kliatib, Ashfaq Khokhar,
and Arif Ghafoor

School of Electrical and Computer Engineering

1285 Electrical Engineering Building

Purdue University

West Lafayette, IN 47907-1285

'This work has been partially supported by NSF Award # IRI-9619812

TABLE OF CONTENTS

Page

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT v

1. INTRODUCTION 1

2. THE PROPOSED WEB VIDEO SEARCHING SCHEME 4

2.1 Meta-data Models for Event Descriptions by Object Trails 7
2.1.1 Spatial-absolute Search 10
2.1.2 Spatial-invariant Search 11
2.1.3 Scale-invariant Search 11
2.1.4 Effectiveness of Trail Search Algorithms 12
2.2 Meta Model for Petri-Net Temporal Characterization 14
2.2.1 Example Query 17
2.2.2 Visual Query Formulation and Evaluation 19
2.2.3 Similarity Measure Computation 21

3. ARCHITECTURE OF A WEB-ENABLED VIDEO SEARCH ENGINE . 26

3.1 Video Motion Indexing Tool 28
3.2 Web Video Search Engine 31

4. CONCLUSION 32

LIST OF REFERENCES 33

LIST OF TABLES

Table Page

2.1 TRAIL-SEARCH : Object Motion Search based on trails 10

2.2 SCALE_INV_SEARCH(g_1, g_2) : Scale-Invariant Trail Search 13

2.3 EVAL_QUERY: Algorithm for Processing a Petri-Net Query 20

2.4 START-GEN: Evaluates starting time of each place in a Petri-Net 22

LIST OF FIGURES

Figure		Page
2.1	(a) The trail for Object 1. (b) The trail for Object 2. (c) Parallel layout as an augmented Petri-net refers to the event *cross-over*. (d) Sequential placement represents *bouncing off the wall*	6
2.2	Trails used in the trail-based match method	9
2.3	Recall-Precision Graphs for (a) spatial-absolute trail search and (b) spatial-invariant trail search. (c) Distance results between sample images for scale-invariant search with SCALE_INV_SEARCH	15
2.4	An example scenario and the corresponding Petri-net configuration as created by the user to query for the *touchdown* event.	18
2.5	Five structurally different Petri-nets: instances from these classes were used to evaluate our metric based on different assumptions	24
2.6	Recall-Precision graphs for (a) Petri-net structure-based classification and (b) Object-trail similarity-based classification.	25
3.1	Architectural Components of the Multimedia Web Search Engine	27
3.2	Video Motion Indexing Tool	28
3.3	Components of the Web Video Search Engine. The upper portion is used for sketching object motions that are used to construct the Petri-net shown in the second part. Every place in this part corresponds to a user-sketched trail (or delay) and the entire Petri-net represents a compact rnodel of an event to be used to search for similar events in the database	30

ABSTRACT

Increasing use of multimedia data makes it crucial to develop intelligent search mechanisms for retrieving multimedia data by content. Traditional text-based methods clearly do not suffice to describe the rich content of images, voice or video. Digital video requires the incorporation of temporal information for any effective content-based retrieval scheme. We present a novel technique which integrates object motion ancl temporal relationship information in order to characterize the events for subsequent search for "similar" clips. We propose a hybrid mechanism based on object motion trails similarity match and interval-based temporal modeling that leads to a unique framework for spatio-temporal content based access in digital video. We implemented the proposed methods and demonstrated that high-level query formulation can be achieved for the aforementioned purpose. Development of such technology will enable true multimedia search engines that will accomplish what current Internet search engines like Infoseek or Excite do today for textual data.

1. INTRODUCTION

The Internet has become a huge repository of data that is available to a diverse spectrum of users all over the world. Ever since the World Wide Web has been incorporated into the Internet, the need to organize, store, search and retrieve data has increased tremendously. During the early days of WWW, data, on the Internet was confined to textual documents where search by keywords fulfilled much of the information retrieval needs of users. However, advances in networking, storage and data creation technologies made it possible to support multimedia data both on stand-alone and networked environments such as the Web. Today, there is an increasing need for robust indexing and search mechanisms over the Web for effective use of multimedia data.

Traditionally, *content* of multimedia data has been represented either by simple textual techniques or low-level image features such as color indices., shape representation and transform domain features. However, these features are not sufficient to represent the rich semantics of digital video. The temporal information involving the relative movements of salient objects in a video must be incorporated in any effective content-based video indexing scheme. For example, the query *search the web for head-on car collisions* cannot be answered by indexing techniques based on color histograms or keyword annotations, but requires proper indexing of the video based on motion information and inter-relationships of the objects (cars) in the video.

With the rapid proliferation of the Web, it is becoming more desirable to provide a common visual query interface for video databases. The popularity of both the content and the interface tools of the Web make it a natural choice for accessing large repositories of video data to support newly emerging Internet applicsations. In a web

search system which is expected to be a real-time application, the need for efficient query processing is another reason for the use of indices. Such an indexing scheme, however, should not be restricted to pre-assigned keywords or just be simple raw data that carries no semantic meaning. In other words, an intermediate model is needed that provides enough semantic power while supporting an efficient mechanism for content-based access.

A number of researchers have proposed techniques to model temporal events. Sorne of these techniques rely on modeling the interplay among physical objects in time along with spatial relationships between these objects. In [6], spatial and temporal attributes of objects and persons are modeled through a directed graph model. Although a formal method of representing video data is proposed, underlying spatial models for motion-based characterization are not provided, imposing severe limitations on the system. An approach that uses spatial relations for representing video semantics is spatio-temporal logic [3]. A prototype image sequence retrieval system is developed, where images are processed and represented by spatio-temporal logic. The prototype provides a novel query interface by which query-by-sketch is employed to query video data. However, due to the limitations of the methodology used in this approach, the modeling of higher level concepts, such as spatio-temporal events, is not addressed. Moreover, spatial and temporal predicates are manually annotated in the database. The framework discussed in [7] defines a set of algebraic operators to allow spatio-temporal modeling, as well as video editing capabilities. After extracting trajectory of a macro-block in an MPEG video, all trajectories of macro-blocks of objects are averaged and a spatio-temporal hierarchy is established. for representing video. While providing a robust low-level analysis, this work does not address the handling of inter-object relationships which is crucial in video content description.

Other techniques to model temporal events depend on the semantic classification of video content. The video model proposed in [13] allows hierarchical abstraction of ***video expressions*** representing scenes and events, which can provide indexing and content-based retrieval mechanisms. It allows users to assign multiple coexisting

interpretations to the same video segment and provides functionalities for creating video presentations which include nested structures and temporal composition. A similar approach is taken to develop an object-oriented abstraction of video data in [10]. A video object in this approach is identical to a video expression in [13] and corresponds to semantically meaningful scenes and events. An object hierarchy is built using IS-A generalizations, and interval inclusion based inheritance is introduced to capture the hierarchical flow of information in this model. Since all these models are purely semantic-based, they have the inherent limitation of being annotation-based and target-user dependent that render their implementation cumbersome and domain-specific.

VideoQ [4] is one of the very few models that directly address motion-based content characterization, but lacks an explicit temporal formalism and comprehensive spakial search techniques that can handle different preferences such as translation spa tial-scale invariant searches.

Most of these approaches emphasize the formalism of semantic modeling but lack practicality in terms of easy-to-use query interfaces for novice Internet users. In this paper, we attempt to address this deficiency by presenting a novel technique of indexing video data which will make it possible to query such data,. The proposed technique will alleviate the limitations of keyword-based search techniques and provide an efficient Web-enabled video searching capabilities. This technique is based on a hybrid approach that effectively captures both the simple and complex semantics and introduces a visual query mechanism that allows fuzzy search of video data.

The remainder of this paper is organized as follows: The next section gives the theoretical foundation of our proposed technique that combines a trail-based representation of object motions with a Petri-net model. The Petri-net model is used for high-level complex event representation as well as imprecise query formulation over the Web. Section 3 describes the implementation of our video search engine, and section 4 concludes the paper.

2. THE PROPOSED WEB VIDEO SEARCHING SCHEME

In our earlier work, we presented a graphical model called Video Semantic Directed Graph(VSDG) as a representation scheme for temporal specification of video content [5]. VSDG provides effective means for inter-object temporal specification that enables spatio-temporal content based access to the semantics in digital video. Based on this foundation, we now present a new scheme that extends the functionality of VSDG in two aspects: (1) trail specification support that enables object movements to be modeled and queried (2) an effective temporal specification for description of more complex scenarios that involve multiple objects or events.

Both trail-based spatial specification methods and temporal motleling techniques have been subjects of research. Generally, simple events such as a ball which draws a spiral are described by simple trajectory representation techniques [7]. However, if the events involve multiple objects or consist of several simple events with a more complex description, a formal mechanism is needed to characterize the temporal relations that are an important part of the event description. Such a mechanism should be intuitive for common users of the Web and ideally, should not require the knowledge of any proprieraty query language.

We have proposed VSDG as a visual tool that represents the temporal layout of the occurrences of the object. A VSDG is a directed graph with a set of circular notles (P), a set of rectangular nodes (T), and a set of arcs *(A)* connecting circular nodes to rectangular nodes. A circular node has an attribute describing the duration for which an object (person, vehicle, etc.) appears in a video segment. A rectangular node corresponds to an event in a video clip whenever a new physical object appears. In other words, a rectangular node marks the start of a new segment, that differs

from its predecessor segment in terms of appearance of a new physical object. Each circular node has exactly one incoming arc and one outgoing arc.

As a graphical data structure, VSDG offers a visual environment where the temporal relations can be effectively represented and constructed. For a more complete representation scheme, however, the places represented by circular nodes must contain more information than mere occurrences of objects. In our new hybrid scheme, the places contain "events" described by object trajectories. A Petri-net based formalism can represent the inter-event relationships in order to construct higher-level event descriptions. The mechanism of searching for events in such a system involves both searching of simple events in the form of object trajectories(circular nodes) and their temporal relationships represented by the Petri-net model. This is discussed later in this section.

As an example to this hybrid framework, refer to Figure 2.1(a)-(d). In this framework, a user first defines simple "events" that involve an object tracing a path on screen, as in Figure 2.1(a) and 2.1(b). These simple events are then combined in a Petri-net based visual model in order to model more complex events for subsequent processing that involves searching for similar events in database. For example, if the object trails in Figure 2.1(a)&(b) represent left-to-right and right-to-left motions respectively, their concurrent alignment in Figure 2.1(c) may represent the event *cross-over* for two similar objects, whereas the sequential layout in Figure 2.1(d) may refer to the event *bouncing off the* **wall** of the same object.

While we provide effective techniques for both trail and temporal specification and searching, their integration within a hybrid framework is a unique scheme and stands out as the major contribution of our work. In addition, the following features are provided in this framework:

- A simple Web interface that allows the queries to be entered through conventional browsers with less ambiguity.

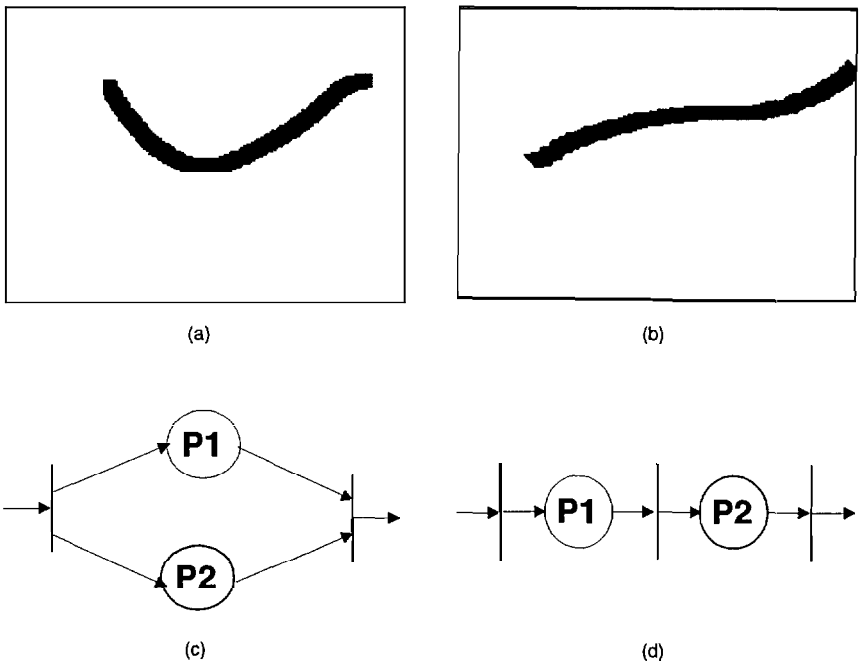


Fig. 2.1. (a) The trail for Object 1. (b) The trail for Object 2. (c) Parallel layout as an augmented Petri-net refers to the event *cross-over*. (d) Sequential placement represents *bouncing off the wall*

- Efficient search mechanisms for trail-based information (Section 2.1) and temporal specification represented by Petri-net based visual query model (Section 2.2).

It must be noted that the proposed method involves the difficult task of low-level processing of video data. Recognition of interesting objects and tracking their motion is not trivial in an automated system and has been a major subject of research for decades. However, we have demonstrated that using semi-automated methods where partial human effort is introduced, content indexing of real video for such a purpose is feasible. We have fully implemented the methods proposed in this paper from preprocessing of video to the presentation of results. In the next two sections, we cover the details of our trail-based spatial and Petri-net based temporal event searching techniques for video data. The implementation details are presented in Section **3.**

2.1 Meta-data Models for Event Descriptions by Object Trails

In order to process user-sketched queries like *give me the clips where an object draws a circle like this(a drawn trajectory),* a similarity measuring mechanism must be developed. Experience shows that in order to answer such queries, any similarity match between sketched queries and database items has to possess certain features. For example, while being specific enough to return a reasonable set of data items, the similarity mechanism should not be too rigid in imposing the matching criteria and allow certain flexibility. The nature of query processing in mulltimedia database systems is quite different from their traditional counterparts in this regard. We list the issues pertaining to the video database retrieval using trail information as follows:

- **Fuzziness** Given the ability to sketch arbitrary trails, it cannot be assumed that the user will sketch the desired scenario exactly. Furthermore, typically more than one item is expected to be returned for a query and these items will not be exact replicas of each other. For these reasons, the similarity criterion

should provide a fuzzy measure based on a metric scale by which the results can be ranked rather than providing *yes/no* answers.

- **Spatial Invariance** The queries may specify a translation-invariant trail which may be located anywhere in the spatial domain. This capability must be provided to the user in order for the queries to be answered regardless of the actual location of the motion on the screen. However, in some cases it may be desirable to restrict the location to exact coordinates of the spatial domain. Therefore, spatial translation invariance should be optional rather than default behavior.

- **Temporal(speed) Invariance** Similar to the spatial invariance, temporal invariance refers to the flexibility of the speeds of the object movements. Users may sketch the trails with arbitrary speeds and may be interested in the trail the objects follow rather than exact locations at exact time instances. For this reason, the similarity matching mechanism should be able to match trails regardless of the speeds of the objects when necessary. Similarly, there should be a mechanism to externally manipulate the general duration of the queried events for the users to have a control over the temporal features of the described scenario.

- **Efficiency** Computational complexity is an important and often underestimated factor in multimedia database applications. With unrestricted length of the database clips, it becomes more important for a trail similarity mechanism to function with an acceptable complexity for real-time query processing.

In order to optimally meet the above criteria, we use a trail-based model that captures the motion of salient objects over a sequence of frames. In this method, we highlight the areas covered by the queried objects as in Figure 2.2 throughout the course of the clip and produce a "picture" of the trajectory for a given clip. In a sense, this is equivalent to taking the mosaic image of the object trajectory in a clip. The trajectory comparison is then carried out through the trail pictures,, i.e. performing

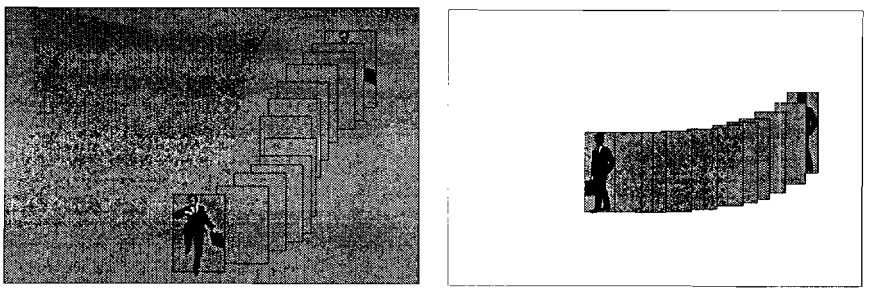


Fig. 2.2. Trails used in the trail-based match method

an image similarity comparison. The objects in Figure 2.2 these clips span similar trajectories which can be compared in different ways including spatial invariant or absolute, rotation-invariant etc.

An important factor to be considered in this method is the impact of the temporal length of the clips. When converted to trail images, very long clips will lose their trajectory information as the repeated scans of the same area is not reflected in the binary trail image representation. For the method to be effectively used, this factor has to be taken into account at the time of parsing the video data.

Another important issue is the handling of the camera motion. In our model, we do not directly deal with this issue and generally assume that screen is a fixed reference frame. However, techniques such as *Salient Stills* [12] and *Mosaicking* [11] offer a practical way to compensate for the camera motion. A still image :representation of a clip can be obtained by combining several consecutive frames of a clip which can also be used for detecting the movements of the salient objects.

Temporal invariance in trail-based models is given by default. In essence, time is frozen throughout the clip and the speed of the object is not reflected in its trail. However, the duration information of each trail is required so that it can be utilized by the high-level temporal model discussed in the next section. Such external control capability on temporal duration provides a great flexibility in user description of the complex events.

Both spatial absolute and invariant searches can be an option, for which a user may choose to restrict the starting point of the motion to what is entered in the

Table 2.1 TRAIL-SEARCH : Object Motion Search based on trails

Input: $DataTrajectories\ DT(k)$, k = 1 ... Number-of_$Clips$ and $QueryTrajectory$ QT
Output: $Similarity(k)$ between $DT(k)$ and QT)
For each $DT(k)$, k = 1 ... $Number\_of\_Clips$ do
    Construct the data and query images $D(i,j), Q(i,j)$: (Z^M x Z^N) $\rightarrow \{0,1\}$
        by assigning 0 to background and 1 to areas covered by trails
    If user-choice = Spatial-Absolute-Search then
        $Similarity(k) = sum(D * Q)/.5 * (sum(D) + sum(Q))$
    else if user-choice = Spatial-Invariant-Search then
        Compute the Fourier transforms of the images as $D_F = \mathcal{F}(D)$ and $Q_F = \mathcal{F}(Q)$
        Similarity(k) = $max(abs(\mathcal{F}^{-1}(D_F * Q_F)))/.5 * (sum(D) + sum(Q))$
    else if user-choice =Scale_Invariant_Search then
        $Similarity(k)$ =SCALE_INV_SEARCH(D,Q)

query or perform a translation invariant match for the desired trails regardless of the exact location on screen. A third case is where spatial scale invariance is introduced. The sketched trail in this case is searched independent of the size of the object or the dimensions of the trail. For example, a small circle and a big circle can be matched to each other and are considered similar in this method. Below, we propose algorithms that efficiently compute the similarity between two trails according to all three user preferences.

The algorithm TRAIL-SEARCH summarizes the steps of our trail search algorithm. According to the user's choice of the search type, the associated images that represent the clip as a motion trail are compared in three different ways. The output of the algorithm, Similarity is sorted and the "best N matches" are displayed according to the user's choice of the number N.

2.1.1 Spatial-absolute Search

For a spatial-absolute search, the user inquires for a motion trail that occurs in an absolute screen location. In this case, two trails such as those in Figure 2.2 are directly compared against each other for a pixel to pixel match. The fact that the trail images

are binary images provides a significant performance advantage, the comparisons are merely a bitwise multiplication between the corresponding pixels. The Similarity step in the algorithm in this case has a quadratic time complexity $O(N^2)$, N being the width or height dimension of the input trail images, which is generally proportional to the screen size.

2.1.2 Spatial-invariant Search

Spatial-Invariant match refers to the comparison of two trails in a translation-invariant fashion. This involves the comparison of two images for all possible possible translations in both dimensions and is computationally intensive. As an efficient way to compare the convoluted images to each other, we use the convolution property of the Fourier Transform which can be stated as

$$Conv[D,Q] = \mathrm{D} * \mathrm{Q} = \mathcal{F}^{-1}(\mathrm{D} \,.\, \mathrm{Q}^*) \tag{2.1}$$

With a Fast Fourier Transform(FFT) implementation of the Fourier transform, this step can be reduced to an $\mathrm{O}(\mathrm{N}^2 \log \mathrm{N})$ time complexity.

2.1.3 Scale-invariant Search

For matching two trails independent of both their starting points on the screen and the size of the object or the sketched trajectory drawn, we use a Mellin transform based scale invariant pattern recognition technique which is summarized in Algorithm SCALE_INV_SEARCH [2]. This method provides both spatial (shift) invariance and spatial scale invariance, due to the scale-invariant nature of the Mellin transform and the convolution scheme used in the algorithm.

Mellin transform of a discrete-time signal $x(k)$ is given by

$$[\mathcal{M}(x)](u) = \sum_{k=0}^{N-1} x(k) k^{-(ju+1)}$$

where j is the complex variable. Scale invariance of Mellin transform can be easily proven by substitution. For $x_\alpha(k) = x(\alpha k)$

$$[\mathcal{M}(x_\alpha)](u) = \alpha^{-ju} [\mathrm{M}(x)](\mathrm{u})$$

Therefore,

$$|[\mathcal{M}(x_\alpha)](u)| = |[\mathcal{M}(x)](u)|$$

Another property of the Mellin transform is its close relationship to Fourier transform. Mellin coefficients can be easily computed from Fourier coefficients by scaling the: input signal by a logarithmic scale. Substituting $l = \log k$ one can show that

$$[\mathcal{M}(x)](u) = [\mathcal{F}(x(e^l))](u) = [\mathcal{F}(x)](\log u)$$

It is worthwhile to comment on the shift and scale invariance option of the algorithm

TRAIL-SEARCH in more detail. Typical user queries do not specify the desired object motion in its exact scale and translation. In other words, it may be desirable to retrieve all object movements resembling a specified trajectory regardless of where it happens on screen or what the dimensionality of the trajectory is. For example, the query *give me all objects that draw a rectangle* will require all clips that involve objects that follow a rectangular path be retrieved. The trail images created for clips will have "pictures" of rectangles in this case and the shift and scale-invariant image match algorithm retrieves them effectively, as demonstrated in Figure 2.3

2.1.4 Effectiveness of Trail Search Algorithms

In order to validate the effectiveness of our proposed trajectory match solutions, we have tested each search case in the algorithm TRAIL-SEARCH and obtained the results shown in Figure 2.3. For the spatial-absolute and invariant cases our sample data set that contains three groups of motions: run, pass and slam. The run category includes players running from left to right in a football video clip, pass represents balls following a parabolic trajectory and slam refers to reflection of the ball as depicted in Figure 2.4. Each group contains an equal number of video clips that are pre-classified into the group manually. A member from each group then is picked as a query clip and compared against the entire data set. The resulting recall-precision graphs in Figure 2.3 indicates that all three algorithms generally provide

Table *2.2* SCALE_INV_SEARCH(g_1, g_2) : Scale-Invariant Trail Search

Input: $g_1(x,y), g_2(x,y) : (Z^N \times Z^N) \rightarrow \{0,1\}$
Output: *Similarity* between trail images g_1 and g_2
Step 1: Calculate the two-dimensional discrete Fourier transform $G;(_{f_x, f_y})$ of the image functions $g_i(x,y)$ where $f_x, f_y = -\frac{N}{2}, ..., 0, 1, ..., \frac{N}{2} - 1$, and $i = 1, 2$.
Step 2: Take the absolute value of the transform and normalize all values to the maximum value at zero frequency. $H_i(f_x, f_y) = \frac{|G_i(f_x, f_y)|}{G_i(0,0)}, i = 1, 2.$
Sitep 3: Logarithmically distort H_i in the direction of fx and fy, putting the result in $D_i, i = 1, 2$.
Sitep 4: Compute the measure function

$$M(k) = \frac{\left(\sum_{u=k}^{N-1} \sum_{v=k}^{N-1} [D_1(u,v) - D_2(u-k, v-k)]^2 \right)^{\frac{1}{2}}}{\left(\sum_{u=k}^{N-1} \sum_{v=k}^{N-1} [D_1(u,v)]^2 + [D_2(u,v)]^2 \right)^{\frac{1}{2}}}, k = 0, 1, ..., N-1$$

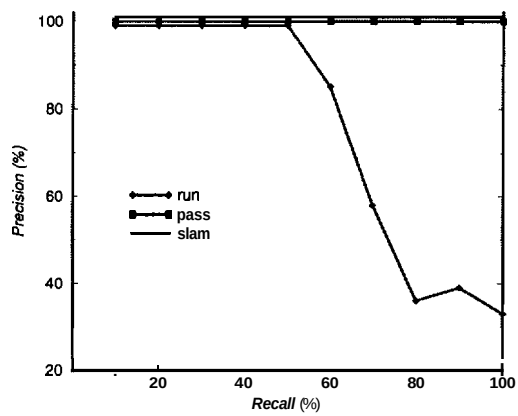
Step 5: Repeat Step **3** and Step 4 for either the upper left quadrant or the lower right quadrant.
Step 6: Compute the *Similarity* by inverting the dissimilarity measure $D\,S\,M = \min(M)$.

satisfactory results. In the spatial absolute search, the precision of the **run** query drops rapidly due to the larger size of the associated object (player): as it is easier for other objects to have large overlapping areas with a larger object for other objects. Higher precision for **run** in spatial invariant case in the second diagram indicates the importance of this option for better retrieval of the desired behavior. Low precision of the **pass** query in the same diagram is proof that in some cases a more robust search technique will be needed than a mere translation invariance.

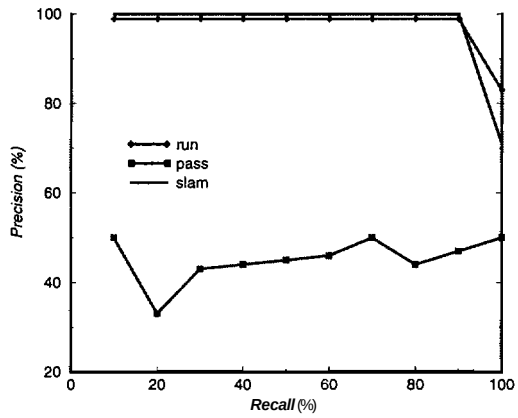
For testing the scale invariant search algorithm, we used a different data set and a rnore definitive metric. The data set in this case consists of three user-sketched trails in two different sizes that resemble the letters r, L and a rectangle, chosen for a better naming convention. The results are quite promising: in each of three categories, dissimilarity measure of Algorithm SCALE_INV_SEARCH(D,Q) gives distinctively close distance between associated classes(small_r to bigs etc.) In general terms, this search type gives the most natural and expected results, but has severe computational disadvantage. For this reason, it is concluded that the spatial-absolute and spatial-invariant searches uses should be used as "quick and dirty" searches and the scale-invariant algorithm should be deployed for higher precision searches.

2.2 Meta Model for Petri-Net Temporal Characterization

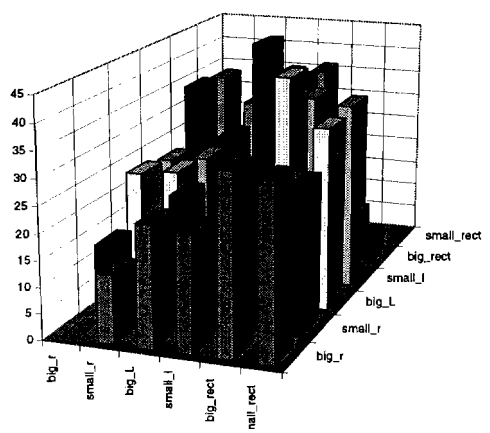
In the previous section, we presented the searching of single object trails, and proposed similarity measures to rank the retrieved video clips. 'Video semantics, however, may consist of complex events involving the interplay of several objects, as discussed earlier in this section. A formalism is needed that allows query specification of such interplay. Two important considerations are in focus for such formalism to be viable. First, it should allow users to create queries in a simple and intuitive manner, without sacrificing the expressive power needed to represent complex events. Second, at the time of query evaluation, the search mechanism should be efficient. In this section, we propose a unique formalism that satisfies these requirements, and provides a visual query mechanism suitable for common Web usage..


100
80
60
40
20
Precision (%)
run
pass
slam
20
40
60
80
100
Recall (%)

(a)


100
80
60
40
20
Precision (%)
run
pass
slam
0
20
40
60
80
100
Recall (%)

(b)


45
40
35
30
25
20
15
10
5
0
small_rect
big_rect
small_l
big_L
small_r
big_r
big_r
small_r
big_L
small_l
big_rect
null_rect

The proposed formalism is based on Allen's thirteen binary temporal intervals [1], that we generalized in [9]. In [8] an equivalence relationship between temporal intervals and time-augmented Petri-nets has been proposed with an application for tennporal integration of multimedia data. Such transformations allow us to use the interval-based Petri-net formalism for visual querying of video data. The key features of this formalism are as follows:

- It provides a formal procedure to specify temporal relations among trails of several objects. In addition, it allows to specify important temporal attributes such as durations of object movements, and their relative occurrences over the timeline.
- The graphical nature of Petri-net facilitates the usage of icon-based visual formulation of queries. The temporal layout of a Petri-net provides an intuitive and natural abstraction and ordering of events as perceived by the user.
- A search mechanism is provided that allows effective retrieval of data, based on object-trail and Petri-net models in an integrated fashion.

The Petri-net model can be extended to provide a high level r-epresentation of temporal relationships among movements of multiple objects. In particular, the trails of salient objects are characterized by not only the shapes of their mosaic trails but also a temporal interval specifying the approximate duration of movement of objects along such trails.

Formally, a Petri-net based scenario specification for visual querying can be defined as Follows:

Definition 1 A scenario specification consists of a set of places (P), a set of arcs (A), and a set of transitions (T). An arc connects a transition to a place or vice versa. Each place has exactly one incoming arc and one outgoing arc. Additionally, each place has the following attributes:

- S : P $\rightarrow$ String, is a mapping from a place to an event associated with the *mosaic trail* of a salient object.
- ψ (d) : P $\rightarrow \psi$, is a mapping from a place to the user-specified duration of the trail.
- A delay place has no symbolic representation.

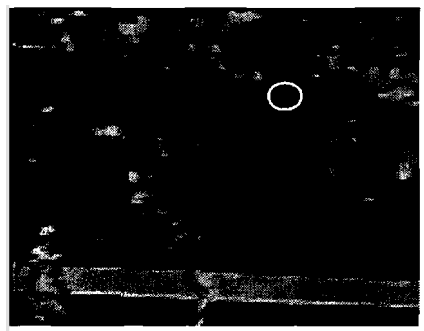
Places denoting delays are also used in order to cover the specification of all thirteen binary temporal relations [1]. Additional attributes for places can be specified by the user to narrow down the search space into a specific domain or object of interest. Such features may include color, shape, and texture. As :mentioned above the key feature of this formalism is its great flexibility in the range of results that may be specified by using a single augmented Petri-net.

2.2.1 Example Query

In order to present a comprehensive and tangible review of our proposed methodology, we provide an example scenario in this section. In this example, the user describes a touch-down event in a football clip from this year's Super Bowl (http://www.nfl.com). In this clip, *Player* 4 makes a touch-down pass while the defense player is blocked by *Player* 63. This involves *Player* 4 running to the right followed by the ball following a parabolic trajectory and slammed on the ground that correspond to the three example events in Section 2.1.4. We summarize the major steps that the user takes as follows

"Player 63 moving to right"
Represented by P2

"Player 4 moving to right"
Represented by P1



"Ball passed"
by P3

ll slammed(score)"
by 4

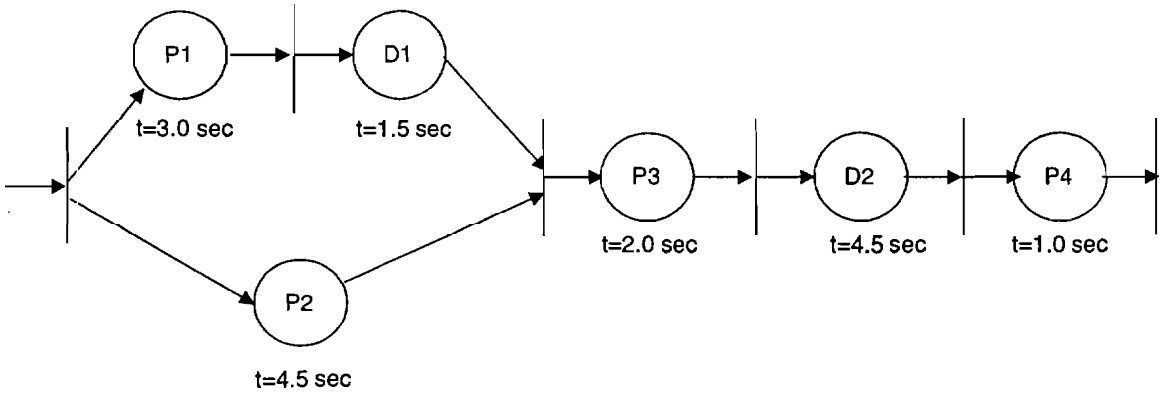


Fig. 2.4. An example scenario and the corresponding Petri-net configuration as created by the user to query for the touchdown event;.

Step 1 User sketches an object trail that indicates *player* 4 *moving left to right* and describes the **run** event.

Step 2 User sketches another object trail that indicates " player 63 moving left to right"

Step 3 User relates the two events as a Petri-net that consists of two places connected "concurrently" as in Figure 2.4

Step 4 User sketches the **pass** event as a parabolic curve and inserts this in the Petri-net sequentially following the two concurrent places of **Step 3**

Step 5 User inserts a delay place(D2) in the Petri-net to count for the falling down of the player. This helps differentiate between a touch-down and dropping of the ball.

Step 6 Similar to Steps 1, 2 and 4, user sketches the *slam* event and places that sequentially at the end of the Petri-net

Note that the durations are entered by the user as approximate and taken into account by the Petri-net searching algorithm in a fuzzy manner to allow for tolerance in temporal parameters.

2.2.2 Visual Query Formulation and Evaluation

Finding a match to a specified visual query is one of the most challenging problems in image/video databases. The neighborhood graph has been proposed as a means of measuring similarities between images based on the spatial relations between objects in these images. Such a measure, however, is too abstract and cannot be used in ranking images satisfying the same structure with different distances. Furthermore, as can be noted from the example in Figure 2.4, the duration parameters associated with places provided by the user may not be exact and can have a wide range of values. At query formulation time, this can result in imprecision in the retrieval process and hence can overload the user with unwanted output. In order to increase the efficiency of retrieval process and to increase performance, it is imperative that an appropriate similarity measure should be specified in order to guide the search strategy. Using

Table 2.3 EVAL_QUERY: Algorithm for Processing a Petri-Net Query

Input: Petri-Net Query Q_p
Output: M video clips that best satisfy Q_p, where $M \geq 0$
Step **1:** Compute the exact starting time of each place in Q_p
Step 2: Retrieve all objects $O_{1,i}$ that are 'similar' to the object(s) represented by the places in Q_p with starting time=1
Step **3:** For each object(s) $O_{1,i}$ do
3.1 for each scenario S_r fitting Q_p with object(s) $O_{1,i}$ at places with starting time = 1 do
3.1.1 Compute the similarity measure $SM(S_r)$
Step 4: Return the video clips of Scenarios S_r with $SM(S_r) < \epsilon$

our Petri-Net formalism, we developed algorithms to determine such ranking between different Petri-net structures, as well as ranking equivalent structures with different user-set parameters. Algorithm EVAL_QUERY in Table 2.2.2 describes the overall processing of Petri-Net queries. In order to compute the similarity measure between the query and possible scenarios, we have to decide on which parameters to use in order to carry out this computation. It is evident that three parameters affect the closeness of a scenario to satisfying a query: the similarity of the trajectory of an object in the database to that of an object in a Petri-Net place; the closeness of the relative starting time of an object in a scenario to that in a Petri-Net query, and the closeness of the duration of an object following its trajectory with its counterpart in the query. The similarity measure of the trajectories is obtained through the algorithms discussed earlier in this section. Durations of each place in the query are specified by the user, whereas durations for objects in a clip can be retrieved from the database. Step 1 involving computation of the starting time of each place in the petri-net is described in the START-GEN algorithm outlined in table 2.2.2. Step 2 limits the scope of scenarios to be checked to those containing trail(s) of object(s) similar to the one(s) specified by the user in his/her query having a starting time of one. In Step **3,** all scenarios that can be formulated and satisfy the condition in

Step 2 of the algorithm, are examined and assigned a similarity measure. Only those scenarios not exceeding a threshold, ϵ, will he returned.

2.2.3 Similarity Measure Computation

In the previous section, we introduced the general algorithm for query evaluation without mentioning the similarity measure used in comparing different high level queries. Based on the identified parameters that play a key role in shaping scenarios, narnely the dis-similarity measure between trajectories, the starting times, and the durations of each object trail and the desired query, we propose the following function in Equation 2.2 for measuring the similarity between a user query, Q, and data element P.

$$SM(Q,P) = \frac{1}{N}\sum_{i=1}^{N}\left\{(1 - T_{q_ip_i}) * \left[C_{start} * \left(1 - \frac{|S_{q_i} - S_{p_i}|}{(S_{q_i} + S_{p_i})}\right)\right] * \left[C_{duration} * \left(1 - \frac{|D_{q_i} - D_{p_i}|}{(D_{q_i} + D_{p_i})}\right)\right]\right\} \quad (2.2)$$

where

N : Number of object trails in Q and P, N ≥ 1
$T_{q_ip_i}$: Dis-similarity measure between the i^{th} trail of Q and P, $0 \leq T_{q_ip_i} \leq 1$, $i = 1, 2, ..., N$
S_{q_i} : Starting time of i^{th} trail of Q, $S_{q_i} \geq 1$, $i = 1, 2, ..., N$
S_{p_i} : Starting time of i^{th} trail of P, $S_{p_i} \geq 1$, $i = 1, 2, ..., N$
D_{q_i} : Duration of i^{th} trail of Q, $D_{q_i} > 0$, $i = 1, 2, ..., N$
D_{p_i} : Duration of i^{th} trail of P, $D_{p_i} > 0$, $i = 1, 2, ..., N$
C_{start} : Cost of error in starting time of a trail in Q and its corresponding trail in P.
$C_{duration}$: Cost of error in duration of a trail in Q and its corresponding trail in P.
$0 \leq C_{start}, C_{duration} \leq 1$, and $C_{start} + C_{duration} = 1$

This similarity measure is obviously non-contradictory since it giives a measure of 1.0 for an exact match of all 3 parameters in all object trails, which is the highest rank a scenario can assume. Moreover, the measure can become zero only in the case of a total mismatch between the object trails of the query and the video data scenario. It is implicitly assumed that scenarios having fewer number of trails to be checked are considered of similarity measure zero. It is evident that the measure puts great emphasis on the trajectory information, not neglecting the temporal scenario of events, returning higher matches for similar trails. This conforms with the perception of the user who rnay not be able to accurately specify the durations of events, but

Table 2.4 START-GEN: Evaluates starting time of each place in a Petri-Net

```
Input: Petri-Net Query Q_p
Output: Petri-Net Query Q_p with each place assigned a starting time
Local Variable: Queue Q initially containing the firing place of the Petri-Net
While (Q is not empty) do
    place = pop(Q);
    if (place reached through initial transition) then
        place.StartTime=1;
    else
        place.StartTime=place.ComingFromTransition.FireTime;
    end if;
    if ((place.StartTime+place.Duration) > place.GoingToTransition.FireTime) then
        place.GoingToTransition.FireTime = place.StartTime+place.Duration;
    if all places going to transition place.GoingToTransition are
      either in Q or already processed then
        push all places p where
            p.ComingFromTransition=place.GoingToTransition
        into Q;
```

tends to remember and/or know the sketch or the trajectory he/she is looking for. One advantage of using this metric and not the Euclidean distance metric is that our metric takes into consideration the relative errors rather than absolute errors, thus giving a better ranking.

In order to verify our claims, we initially classified our data into 5 classes based on the Petri-Net structure each data set assumes as shown in Figure 2.2.3. Each instance has equal durations for all object trails in that instance. Moreover, these durations are used in all five classes. We ran two experiments. In one experiment, we neglected the cost of object trail mismatch and considered the Petri-Net structure only, as well as the different durations. We chose an instance from the first class, perturbed the durations, and ran the EVAL_QUERY algorithm. As can be seen from the precision and recall of the algorithm, shown in Figure 2.6, based on the nurnber of retrieved instances from the class which the query belongs to, the algorithm performed very poorly. This should not surprise us, since the search space we are looking into is of high dimensionality, depending on the number of object trails in the query, their starting times, and their durations.

In the second experiment, we included the trajectory mismatch measure information and used the same query instance. We classified the data into three classes based on the Euclidean distance of the similarity measure of the object trails from the query trails. As the precision and recall values in Figure 2.6(b) show, our second classification gave much better results than the first one. Even when there are differences in the temporal interval-based specifications, highly similar trails give better accuracy.

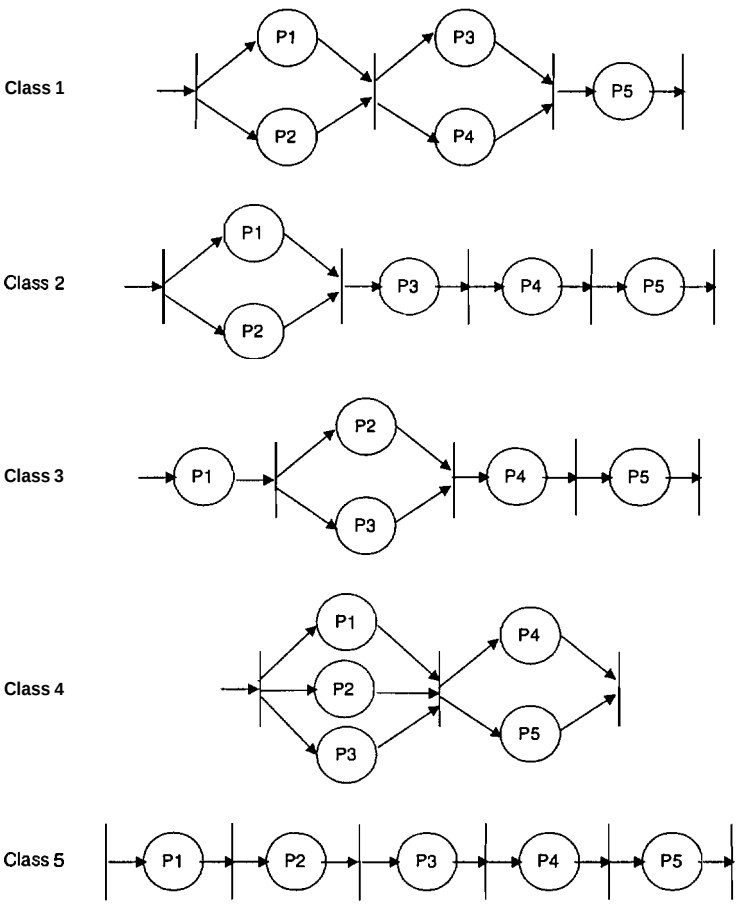
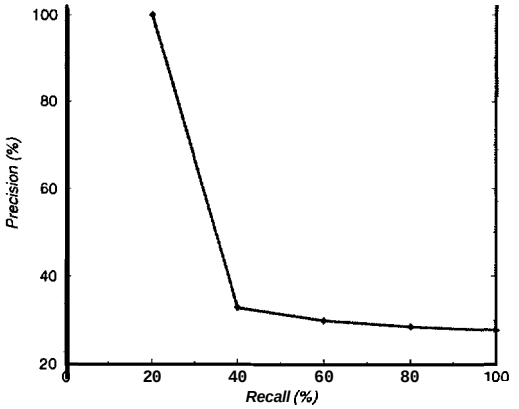
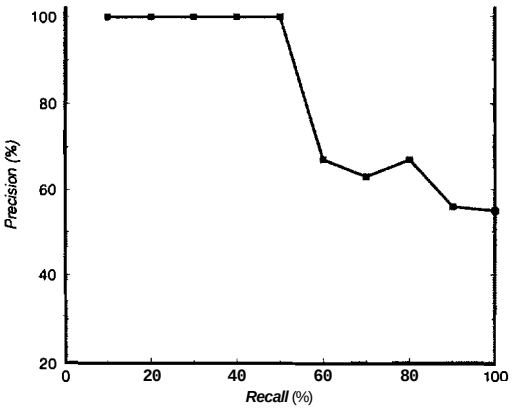


Fig. 2.5. Five structurally different Petri-nets: instances from these classes were used to evaluate our metric based on different assumptions



(a)



(b)

Fig. 2.6. Recall-Precision graphs for (a) Petri-net structure-based classification and (b) Object-trail similarity-based classification.

3. ARCHITECTURE OF A WEB-ENABLED VIDEO SEARCH ENGINE

We have built a Web-enabled prototype database system that employs the trail-based motion indexing and Petri-net based temporal query construction mechanisms discussed in the previous section.

The architectural components of this system is depicted in Figure 3.1. According to this model, a user can independently specify trajectories of individual objects and the temporal relationships between object trajectories. The temporal relationships are modeled as a Petri-net based graphical data structure. Query scenarios represented as a Petri-net carry two pieces of information used in the search process: (1) Trajectory information represented by object trails and (2) Temporal information reflected by the structure of the Petri-net. The former is included in the Petri-net as a circular node and the entire Petri-net is used as a representative of the described scenario.

Query processing involves two steps, the first being the comparisons of individual trail representations in three different search types (spatial-absolute, spatial-invariant and scale-invariant) using the Algorithm **TRAIL-SEARCH.** The resulting ranked indices(similarities) between the query trails and trails in the database for the subsequent temporal processing. In this step, which constitutes the second phase, the temporal features of the Petri-net based query is matched against the existing object compositions in the database using the Algorithm EVAL_QUERY and the final results are obtained. Accordingly, a number of clips that are ranked as the best matches are returned to the user.

In order for the video clips to be accessed by such a system, raw data has to be processed first. For this purpose, the existing data clips are first indexed similar to

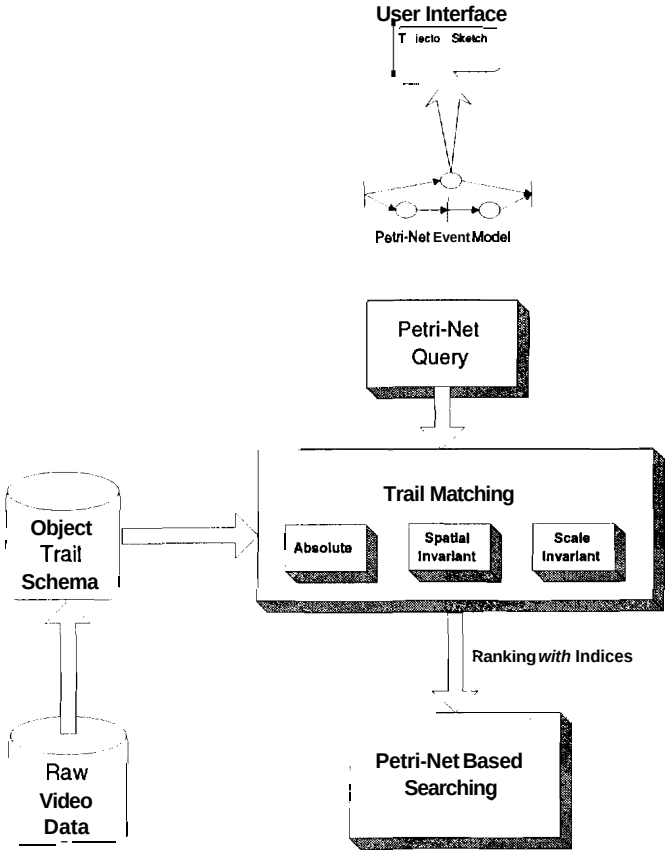


Fig. 3.1. Architectural Components of the Multimedia Web Search Engine

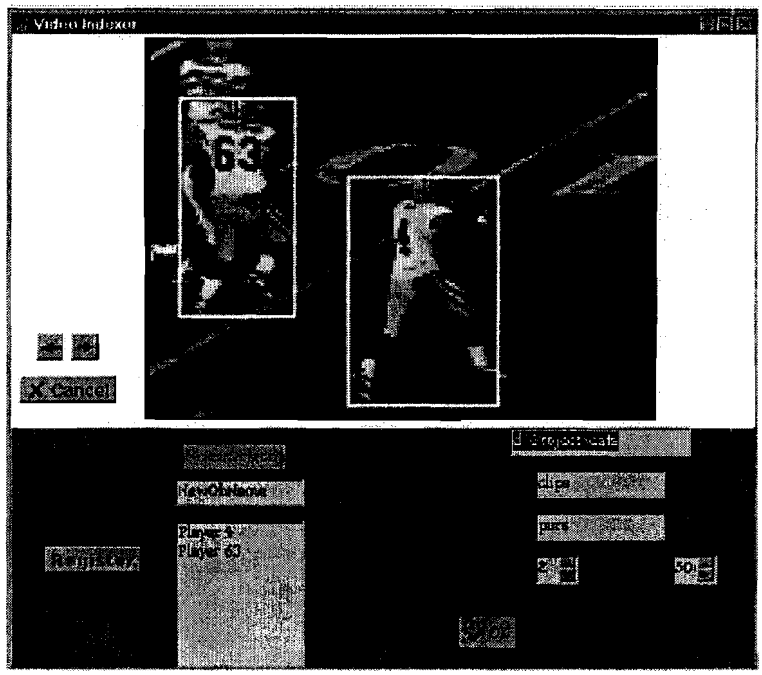


Fig. 3.2. Video Motion Indexing Tool

the construction of the sketched queries, and the Object Trail Schema is constructed as discussed in Section 2.1. This schema contains the trails of the salient objects much similar to the user sketches and is used in the Trail Matching step to search for queried clips based on their image representation.

We have implemented the above outlined system in Windows platform. The overall system consists of two components: The Video Indexing Tool and the Web Video Search Engine. In the next two sections, we cover some implementation details of these tools and explain their functioning briefly.

3.1 Video Motion Indexing Tool

Raw Video Data goes through a pre-processing step where the objects are identified and their position and size are specified with a bounding box, Minimum Bounding Rectangle (MBR). Despite their known limitations, MBRs provide an efficient way to represent approximate location of objects on the video coordinate space. During

the pre-processing step, the MBR of each user-identified object needs to be determined and recorded for each frame. When manually carried out, this is a tedious task especially at high frame rates. Our experience proved that frame sampling used with interpolation for "inter-frames" result with noticeably efficient results without compromising accuracy significantly. Therefore, we only index selected key-frames ancl interpolate the trajectories for inter-frames.

Automatic detection and recognition of objects is an extremely challenging task. Current technology offers only partially acceptable solutions that can be incorporated in a video indexing mechanism. It has been widely accepted that with the current state of the art in the technology, software tools can most effectively be used as an aid to human users for the purpose of extraction of the "interesting" information, but totally automated indexing is far from being accomplished. Towards this goal, we use a semi-manual object tracking tool for capturing MBRs. For practicality, this is done for each intra-frame, typically 2 or 4 frames per second and trajectories are interpolated for inter-frames.

Figure 3.2 shows the interface of the indexing tool we use to generate MBR-based indices in the database. Selected frames of a given clip are loaded one by one on the image area and bounding boxes are drawn by the users. The name of the corresponding object is entered or picked from the list for every MBR created. The generated index for each video clip contains the number of objects, their names, sizes (width and height), starting frame numbers and the duration lengths along with the X and Y coordinates at each frame instance which are later used to construct the motion trails in Algorithm TRAIL-SEARCH .

The requirement of pre-processing of video within the proposed framework is a severe shortcoming and is a general handicap in content-based multimedia access. The widespread use of the upcoming content-aware video representation standards such as MPEG4 or MPEG7 will help facilitate such systems on networlked environments such as the Web in the future.