データベースシステム 128−46
（２００２．７．１８）

Searching Videos by Their Time-Varying Features

Zaher AGHBARI, and Akifumi MAKINOUCHI

Graduate School of Information Science and Electrical Engineering,
Department of Intelligent Systems, Kyushu University 6-10-1 Hakozaki,
Higashi-ku, Fukuoka-shi 812-8581, Japan
E-mail: {zaher,akifumi}@is.kyushu-u.ac.jp

Abstract

In this paper, we present a video search system based on the time-varying features of objects. In this system, video shots are modeled at three levels, object level, frame level, and shot level. The model captures the visual features of individual objects at the object level, visual-spatio-temporal (VST) relationships between objects at the frame level, and time-varying visual features and time-varying VST relationships at the shot level. We call the combination of the time-varying visual features and the time-varying VST relationships a *content trajectory*, which is used to represent and index a video shot. A novel query interface, that allows users to describe queries, by sketch and feature specification, is presented. Our experimental results proves the effectiveness of modeling and querying video shots using the content trajectory approach.

1 Introduction

The recent extensive research in multimedia database and the rapid spread of the Web have resulted in the emergence of new applications, such as digital libraries, surveillance systems, news-on-demand, distance learning, etc., which utilize video data. Nowadays, there are many video libraries that include sport clips, news clips, animation clips, etc. that are accessible by a wide range of users through the World Wide Web.

The main difference between still images and videos stems from the motion of objects in videos. As video objects move, the visual features (color, motion, etc.) of objects, and the relationships between multiple objects (such as the visual-spatio-temporal, *VST*, relationships shown in Table 1) may change over time. Therefore, an effective video representation should take into account the time-varying visual features, which we call a *feature trajectory* ($\mathcal{F}$), of individual video objects and the time-varying *VST* relationships, which we call a *relationship trajectory* ($\mathcal{R}$), between objects. We call the combination of all $\mathcal{F}$ s and $\mathcal{R}$ a *content trajectory* (Θ).

Several works have proposed models to index, and retrieve videos based on the time-varying characteristics of video data. An approach to query videos by the time-varying directions of moving objects is proposed by [4], but it does not support queries that explicitly investigate spatio-temporal relationships between multiple objects. Other models, such as [8] and [9], proposed a representation of videos by the spatio-temporal features of moving video objects based on Egenhofer's spatial relationships, [6], and Allen's interval temporal relationships, [1]. Also, [8] and [9] proposed a motion trajectory to represent the time-varying directions of objects. In [5], a video is indexed by an object's trail (the area covered by the queried object throughout the clip), but this model does not support querying of multiple objects. All of the models, [4][5][8][9], do not represent time-varying visual features other than direction, and their query

Table 1: VST relationships between objects

Relationship set	**Member Relationships**
Egenhofer's spatial topological	*equal*,*inside*,*contains*,*covers*,*coveredBy*, *overlaps*, *meets*, *disjoint*
Allen's interval temporal	*equal*, *before*, *after*, *during*, *contains*, *overlaps*,*overlapedBy*, *meets*, *metBy*, *starts*, *startedBy*, *finishes*, *finishedBy*
relColor	*sameColor*, *lighter*, *darker*
relSize	*sameSize*, *bigger*, *smaller*
relPosition	*samePosition*,*above*,*below*,*toTheRight*,*toTheLeft*
relSpeed	*sameSpeed*, *slower*, *faster*
relDirection	*sameDirection*,*opposite*,*perpendicular*, *sameWithAngle*, *oppositeWithAngle*, *towards*, *awayFrom*
relAppearanceTime	*inBefore*,*inWith*,*inAfter*,*outBefore*, *outWith*, *outAfter*

tools do not allow users to specify the appearance and disappearance of an object trajectory in reference to trajectories of other objects.

In this paper, we introduce: *(i)* an effective hierarchal video representation that describes a video at three levels, OL, FL, and SL. *(ii)* a novel query interface which allows users to describe feature trajectories (time-varying color, direction, speed, and position) of individual objects and a relationship trajectory (time-varying VST relationships) between multiple objects in a simple and intuitive manner; in addition, users can specify the appearance and disappearance of an object trajectory in reference to other objects' trajectories. *(iii)* several experiments that prove the effectiveness of our approach.

A long version of this paper can be found at [3].

2 Video Model

Each video, $\mathcal{V}$, undergoes several preprocessing steps before the start of a hierarchal representation of a video:

(i) Shot segmentation. A video is segmented into a number of *shots* $(S_1, S_2, \ldots, S_n)$. Where, a *shot, S, is a consecutive sequence of frames that constitute one camera operation.*

(ii) Event detection. Each shot constitutes one or more *events* $(E_1, E_2, \ldots, E_h)$. Where, an *event, E_i, is a subsequence of consecutive frames that express a particular activity and contain a fixed number of semantically meaningful objects.* The start and end of an event are detected by the appearance of a new object into the scene or the disappearance of an existing object from the scene.

(iii) Keyframe selection. Each event is represented by at least two keyframes (first and last frames of the event), but if the event is longer than one second, one keyframe per second is extracted.

(iv) Object segmentation and Tracking. From each keyframe, a set of semantically meaningful *video objects* $(O_1, O_2, \ldots, O_q)$ are extracted. There are many methods, e.g. [4][7], for segmenting and tracking objects in videos that are coded by non-object-oriented encoders, such as MPEG-1 or MPEG-2. In this paper, we assume that videos are coded by an object-oriented video encoder, such as MPEG-4; therefore, segmentation and tracking information of video objects are provided in the video input stream.

3 Content Representation

Our model support a hierarchal representation of video objects. That is, it represents video shots at three levels, object level (OL), frame level (FL), and shot level (SL).

3.1 Object Level

At this level, individual objects are represented by their visual features, color, motion, and absolute position.

(i) Color. The colors of an object at keyframe κ_i are represented by *color histogram* C:

$$C : \{(r_1, g_1, b_1, p_1), (r_2, g_2, b_2, p_2), \ldots, (r_l, g_l, b_l, p_l), \kappa_i\} \tag{1}$$

where r, g, and b are the red, green, and blue components of an RGB color, respectively. And, l is the maximum number of colors in the color histogram. In our experiments, we set $l = 10$, which are enough to represent colors of a single object.

(ii) Motion. We consider an object to be *rigid*, that is at any time instant an object moves to one direction, η, at one speed, v. Then, the motion histogram M of an object at keyframe κ_i is:

$$M : \{(\eta, v), \kappa_i\} \tag{2}$$

(iii) Position. We approximate each object by its Minimum Bounding Rectangle (MBR). Thus, we only need two points, $(x_1, y_1), (x_2, y_2)$, to determine the absolute position, P, of an object at keyframe κ_i.

$$P : \{(x_1, y_1), (x_2, y_2), \kappa_i\} \tag{3}$$

3.2 Frame Level

At this level, the *VST* relationships, which are shown in Table 1, between multiple objects are computed. These *VST* relationships are: Egenhofer's spatial relationships, Allen's interval temporal relationships, relative color, relative Size, relative Position, relative Speed, relative Direction, and relative Appearance/disappearance Time. The definitions of those relationships are based on the visual features (C, M, and P) of objects, [2].

At each keyframe, the relationships between any pair of objects (e.g. O_i and O_j) are expressed by a *bit vector*, $\vec{\beta}_{O_i,O_j}$. Each bit, β_h, in $\vec{\beta}_{O_i,O_j}$ represents one member relationship of Table 1.

If a keyframe contains n objects, where $n > 2$, the number of possible pairs of objects, N_{pp}, is equal to $\frac{n^2-n}{2}$. We call the collection of the bit vectors at keyframe κ_j a *container*, ϱ_j.

$$\varrho_j : \vec{\beta}_1, \vec{\beta}_2, \ldots, \vec{\beta}_{N_{pp}} \tag{4}$$

3.3 Shot Level

At this level, a *feature trajectory*, $\mathcal{F}$, a *relationship trajectory*, $\mathcal{R}$, and a *content trajectory*, Θ, are computed. First, let ζ represents the instantaneous visual features (C, M, and P) of still object O_j at keyframe κ_i.

$$\zeta_{\kappa_i}(O_j) : (C_i, M_i, P_i) \tag{5}$$

(i) We view $\mathcal{F}$ as a sequence of the object's instantaneous visual features in the keyframes in which this object exists.

$$\mathcal{F}(O_j) : \zeta_{\kappa_1}(O_j), \zeta_{\kappa_2}(O_j), \ldots, \zeta_{\kappa_h}(O_j) \tag{6}$$

Where the sequence, $\kappa_1, \kappa_2, \ldots, \kappa_h$, is not necessarily consecutive since an object may disappear and reappear several times during a shot. One $\mathcal{F}$ is computed for each object in a video shot.

(ii) The *VST* relationships between the objects that exist at keyframe κ_i are represented by *container* ϱ_i, as we discussed in Section 3.2. We call the sequence of containers a *relationship trajectory*, R.

$$\mathcal{R} : \varrho_1, \varrho_2, \ldots, \varrho_{N_k} \tag{7}$$

Where, N_o is the total number of objects in a shot, and N_k is the total number of extracted keyframes from a shot. One $\mathcal{R}$ is computed for every shot.

*(iii)*We call the combination of all $\mathcal{F}$ s and $\mathcal{R}$ a *content trajectory*, Θ, which is used to index a shot.

$$\Theta : \rho_{s_1}, \rho_{s_2}, \ldots, \rho_{s_{N_k}} \tag{8}$$

Where each ρ_{s_i} (*shotPacket*) corresponds to one keyframe, κ_i. Each ρ_{s_i} contains the instantaneous features of N_o objects that exist at κ_i and one container, ϱ_i, that holds the *VST* relationships between the objects.

$$\rho_{s_i} : \zeta_{\kappa_i}(O_1), \zeta_{\kappa_i}(O_2), \ldots, \zeta_{\kappa_i}(O_{N_o}) \quad + \quad \varrho_i \tag{9}$$

4 Query Processing

Our model supports similarity retrieval. Therefore, we built an interface that allows a user to describe, by sketch (see Figure 1) and feature specification (see Figure 2), time-varying visual features of individual objects and time-varying *VST* relationships between multiple objects in a simple and intuitive manner. This Query interface has two major contributions: *(i)* it allows users to specify by sketch the appearance and disappearance of an object trajectory in reference to trajectories of other objects. *(ii)* it allows a user to specify the changes in visual features of objects and changes in *VST* relationships between multiple objects along the paths of the objects.

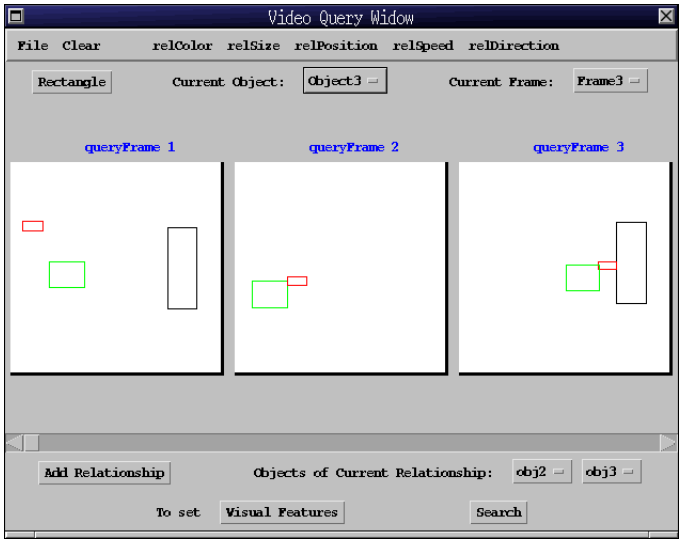

Figure 1: Video query trajectory sketch window

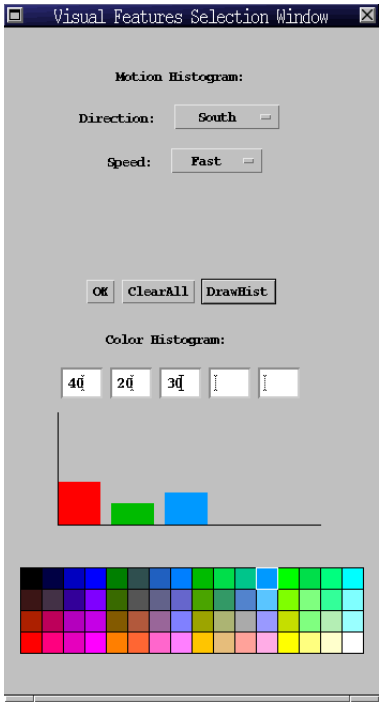

Figure 2: Visual features specification window

4.1 Query Formulation

A query interface, shown in Figure 1, contains a sequence of sketch areas, which we call *queryFrames*. In Figure 1, three queryFrames are shown, but a user can get more by using the scroll bar. Each object is represented by a uniquely-colored rectangle. Rectangles are drawn on the queryFrames to represent the time-varying changes on objects' paths and the interaction between objects. At each queryFrame, a user can specify direction, speed, and color(s) of a selected object in a separate pop up window, Figure 2. Moreover, a user can specify *VST* relationships between object at any queryFrame.

From the sketch and the specified visual features and *VST* relationships, a *query trajectory*, Q, is generated.

$$Q : \rho_{u_1}, \rho_{u_2}, \ldots, \rho_{u_{N_q}} \tag{10}$$

Where N_q is the number of sketched queryFrames. Each ρ_{u_i} (*queryPacket*) contains the instantaneous visual features of the queried objects and their *VST* relationships at one queryFrame.

4.2 Query Evaluation

To retrieve a certain shot, a user query (which is represented by *query trajectory* Q) is matched against every shot (which is represented by *content trajectory* Θ) in the database. The Match_Trajectories algorithm summarizes the steps of matching Q against Θ. The distance between every ρ_{u_i} in Q and every ρ_{s_j} in Θ is computed by the Match_Packets algorithm. Since our model supports similarity retrieval, the number of *queryPackets* is not necessarily equal to the number of *shotPackets*. Therefore, it is crucial to correspond each ρ_{u_i} to its most similar (distance is smallest and less than σ) ρ_{s_j} when matching Q against Θ. If the smallest distance of ρ_{u_i} is greater than σ, ρ_{u_i} is considered dissimilar to all ρ_{s_j}.

Algorithm: Match_Trajectories
Input: Q and Θ
Output: *shot_dist*
for each ρ_{u_i}, i=1..$Num\_queryPackets$ in Q do
  for each ρ_{s_j}, j=1..$Num\_shotPackets$ in Θ do
    $packet\_dist[i][j]$ = Match_Packets(ρ_{u_i},ρ_{s_j})
set low_limit = 1
for each ρ_{u_i}, i=1..$Num\_queryPackets$ in Q do
  for each ρ_{s_j}, j=low_limit..$Num\_shotPackets$ in Θ do
    find $smallest\_dist$ in $packet\_dist[i][j]$
    if($smallest\_dist \leq \sigma$)then
      sum=sum+(packet_dist[i][j]) & $low\_limit = j + 1$
    else sum = sum + 1
$shot\_dist$ = sum / Num_queryPackets

The Match_Packets algorithm, matches each ρ_{u_i} in Q against every ρ_{s_j} in Θ by computing the distances between the visual features of individual objects and VST relationships in ρ_{u_i}, and their corresponding visual features and VST relationships in ρ_{s_j}, respectively.

Algorithm: Match_Packets
Input: ρ_{u_i} and ρ_{s_j}
Output: *packet_dist*
for each O_i, i=1..$Num\_queryObjects$ in ρ_{u_i} do
  for each O_j, j=1..$Num\_shotObjects$ in ρ_{s_j} do
    compute $color\_dist(I_{O_i}, J_{O_j})$ & $motion\_dist(I_{O_i}, J_{O_j})$ by
    $= \left| \sum_{i=1}^{g} [(I_i - J_i) + \sum_{\substack{j=1 \\ j \neq i}}^{l} a_{ij} * (I_j - J_j)] \right|$
    I & J : color (or motion) histograms of O_i & O_j
    a_{ij} : perceptual similarity between I_j and J_j
    add and average $color\_dist(I_{O_i}, J_{O_j})$ into avC.
    add and average $motion\_dist(I_{O_i}, J_{O_j})$ into avM.
for each $pair_i$, i=1..$Num\_queryPairs$ in ρ_{u_i} do
  for each $pair_j$, j=1..$Num\_shotPairs$ in ρ_{s_j} do
    compute $VST\_rel\_dist(X_{pair_i}, Y_{pair_j})$ by
    = distance between relationships X and Y as
    defined by neighborhood graph method, [2].
    add and average $VST\_rel\_dist(X_{pair_i}, Y_{pair_j})$ into avR.
$packet\_dist = avC$*weight1+avM*weight2+avR*weight3
return *packet_dist*

5 Experimental Results

In order to validate the effectiveness of our model, we built a prototype and performed several experiments on a collection of 190 video shots categorized into sports, movies, and animation. The first experiment computes *recall*, which measures the ability of a model to retrieve relevant shots, and *precision*, which measures the ability of a model to reject false alarms [4]. We issued 20 separate queries and compared them against their established ground truth. The return list is increased from 1 to 10 shots and the resulting average precision and recall curves are plotted in Figure 3. Notice that precision (Figure 3.a) starts high and decreases as the number of returned shots increases due to the high possibility of retrieving more irrelavant shots in the returned list as the returned list grows bigger. Recall of the system (Figure 3.b) shows a steady increase of the recall value as the number of shots in the returned list increases. Therefore, both Figures 3.a and 3.b proves the effectiveness of our model.

The second experiment measures the effectiveness of the length (number of queryPackets) of the query trajectory to get a particular shot in the return list. We issued 20 separate queries

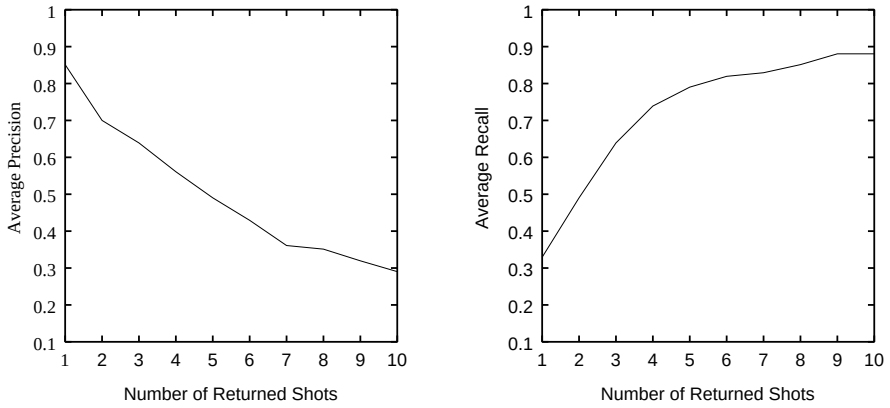

Figure 3: Average precision and recall curves.

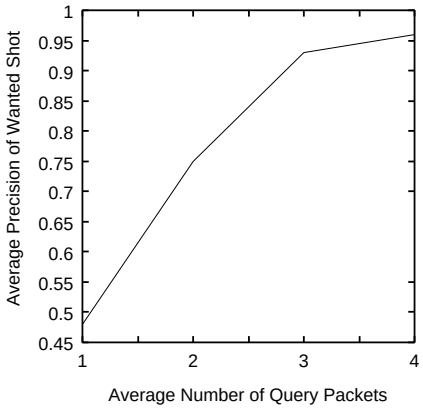

Figure 4: Average number of query packets needed to reach a video shot.

and varied their lengths from 1 to 4 queryPackets to find the correct shot. The average results are plotted in Figure 4 which shows that precision increases as the length of Q increase. An increase in precision means that a desired shot moves closer to the top of a returned list. Figure 4 emphasizes the importance of trajectories in video querying.

In Figure 5, we show the effect of the number of feature trajectories (the number of queried objects) on the precision of a wanted shot. Since one feature trajectory is generated for every object, we control the number of feature trajectories by changing the number of objects in Q. This experiment was performed on 20 randomly selected video shots. For each shot, we formulate a query Q with several query packets and a combination of visual features and VST relationships, then we change the number of objects in Q from 1 to 5. Figure 5 shows the average precision of a wanted shot. We notice that precision of a wanted shot increases as the number of feature trajectories increases .

6 Conclusion

We presented a video model that efficiently represents videos hierarchally at three levels, object level, frame level, and shot level. The model captures the visual features of individual objects in the object level, the VST relationships in the frame level, and the time-varying visual features

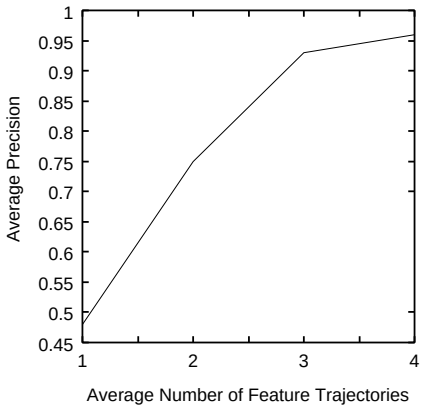

Figure 5: Effect of the number of feature trajectories on precision of a query to retrieve a particular shot.

and time-varying *VST* relationships in the shot level. The query interface allows users to describe queries, by sketch and feature specification, in a simple and intuitive manner. The results of the conducted experiments show the effectiveness of our model and prove the importance of trajectory_based queries to improve precision of the returned list of video shots.

References

[1] J.F.Allen. *Maintaining knowledge about temporal intervals.* Communications of ACM, vol.26(11), pp.832-843. Nov.1983.

[2] Z.Aghbari, K.Kaneko, A.Makinouchi. *VST-Model: A Uniform Topological Modeling of the Visual-Spatio-Temporal Video Features.* IEEE Int'l Conference on Multimedia Computing and Systems (ICMCS'99), june 1999.

[3] Z.Aghbari, K.Kaneko, A.Makinouchi. *Content-Trajectory Approach for Searching Video Databases.* IEEE Transactions on Multimedia. To appear.

[4] S.F.Chang, W.Chen, H.J.Meng, H.Sundaram, D.Zhong. *VideoQ: An Automated Content Based Video Search System Using Visual Cues.* ACM Multimedia Conf. Seattle, 1997.

[5] S.Dagtas, W.Al-Khatib, A.Ghafoor, A.Khokhar. *Trail-based approach for video data indexing and retrieval.* ICMCS'99, Florence, Italy, Jun. 1999.

[6] M.J.Egenhofer. *Point-Set Topological Spatial Relations.* int'l Journal of Geographical Information Systems, vol.5(2), pp.161-174, 1991.

[7] A.M.Ferman, A.M.Tekalp, R.Mehrotra. *Effective Content Representation for Video.* IEEE ICIP. Chicago, Oct. 1998.

[8] J.Z.Li,M.T.Ozsu,D.Szafron. *Modeling of Moving Objects in a Video Database* IEEE ICMCS'97. pp.336-343, June 1997.

[9] M.Nabil, A.H.Ngu, J.Shepherd. *Modeling Moving Objects in Multimedia Databases.* Fifth Int'l Conference on Database Systems for Advanced Applications. Melbourne, Australia, April, 1997.