Object Tracking and Multimedia Augmented Transition Network for Video Indexing and Modeling

Shu-Ching Chen
School of Computer Science
Florida International University
Miami, FL 33199
chens@cs.fiu.edu

Mei-Ling Shyu
Department of Electrical & Computer Engineering
University of Miami
Coral Gables, FL 33124
shyu@miami.edu

Chengcui Zhang
School of Computer Science
Florida International University
Miami, FL 33199
czhang02@cs.fiu.edu

R. L. Kashyap*
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
kashyap@ecn.purdue.edu

Abstract

In our previous work, a multimedia augmented transition network (ATN) model together with its multimedia input strings were proposed to model and structure video data. The multimedia ATN model is based on the ATN model that has been used in the artificial intelligence (AI) areas for natural language understanding systems and its inputs are modeled by the multimedia input strings. The temporal and spatial relations of semantic objects are captured by an unsupervised video segmentation method called simultaneous partition and class parameter estimation (SPCPE) algorithm and are modeled by the multimedia input strings. However, the segmentation method used is not able to identify the objects that are overlapped together within video frames. The identification of the overlapped objects is a great challenge. For this purpose, a backtrack-chain-updation split algorithm that identifies the split segment (object) and uses the information in the current frame to update the previous frames in a backtrack-chain manner is developed in this paper. The proposed split algorithm provides more accurate temporal and spatial information of the semantic objects for video indexing.

Key words: Multimedia *Augmented Transition Network (ATN), Multimedia Input String, Backtrack-Chain-Updation Split Algorithm, Multimedia Browsing, and Multimedia Database Systems*.

1. Introduction

As more information sources become available in multimedia systems, the need for efficient modeling, browsing, searching, and retrieving information, especially for video data, increases. Digital video has been widely used in many multimedia applications such as education and training, video on demand, video conferencing, and so on. Instead of sequential access to the video contents, how to structure and model video data so that users can quickly and easily browse and retrieve interesting materials becomes an important issue in designing multimedia information systems [Yeo97]. To better model and structure video data, a multimedia augmented transition network (ATN) model that is based on the augmented transition network (ATN) together with its inputs, the multimedia input strings, were proposed to model and structure video data [Chen99]. The augmented transition network (ATN), developed by Woods [Woods70], has been used in natural language understanding systems and question answering systems for both text and speech. Multimedia input strings are similar to regular expressions [Kleene56] and are used to represent the presentation sequences of temporal media streams, spatio-temporal relations of semantic objects, and keyword compositions. In our previous studies, we have already shown that the multimedia ATN model and multimedia input strings can model the multimedia presentations, multimedia browsing, multimedia database searching, and the temporal, spatial, or spatio-temporal relations of various media streams and semantic object [Chen99, Chen00].

Moreover, a multimedia information system should allow users to retrieve their interested video materials via database queries. To answer multimedia database queries related to the temporal or relative spatial positions of semantic objects, it is necessary to have object-based representation of video data. For this purpose, more and more attention has devoted to segmenting video frames into regions such that each region, or a group of regions, corresponds to an object that is meaningful to human viewers [Courtney97, Ferman97]. Therefore, key frame selection based on object-based representation is an important issue for video data [Arman94, Day95,

* This work has been partially supported by National Science Foundation under contract IRI 9619812.

Flickner95, Oomoto93, Yeo97]. A key frame selection approach that is based on the number, temporal, and spatial changes (represented by multimedia input strings) of the semantic objects in the video frames was developed in [Chen99].

The proposed key frame selection approach is based on the temporal and spatial relations of semantic objects in each shot and employs the simultaneous partition and class parameter estimation (SPCPE) algorithm [Sista99] to facilitate the multimedia ATN model. The temporal and spatial relations of semantic objects are captured by the SPCPE algorithm and are modeled by the multimedia ATNs and multimedia input strings. However, the SPCPE algorithm is not able to identify the objects that are overlapped together within video frames. In other words, the SPCPE algorithm cannot distinguish the overlapped objects, which may cause inaccurate answers to multimedia queries. Hence, if we can extend the functionality of the algorithm to allow the identification of the overlapped objects, more accurate information can be obtained to answer more detailed multimedia queries.

In this paper, we propose a backtrack-chain-updation split algorithm that can find the split segment (object) and use the information in the current frame to update the previous frames in a backtrack-chain manner. This split algorithm is an enhancement of the SPCPE algorithm to allow the distinguishing of two separate objects that were overlapped previously. That is, the proposed split algorithm provides more accurate temporal and spatial information of the semantic objects for video indexing. Hence, this algorithm can enhance the efficiency of the current ATN model by saving the storage and improving the accuracy of multimedia database queries related to the relative spatial positions of the semantic objects. Moreover, the proposed split algorithm is applied to a portion of a soccer game video clip and the experimental results show that our algorithm can recover the overlapped situation successfully and automatically without user intervention.

This paper is organized as follows. Section 2 gives a brief review of how to use ATNs and multimedia input strings to model video indexing and browsing. Section 3 discusses the mechanism of the proposed backtrack-chain-updation split algorithm. Conclusions are given in Section 4.

2. Using Multimedia ATNs and Multimedia Input Strings for Video Indexing and Browsing

As mentioned earlier, the multimedia ATNs and multimedia input strings are used to model the temporal and relative spatial relations of various media streams and semantic objects, multimedia database searching, multimedia browsing, and multimedia presentations. Multimedia input strings are the inputs for the multimedia ATNs and represent the presentation sequence of temporal media streams, spatio-temporal relations of semantic objects, and keyword compositions. Moreover, a key frame selection approach based on the temporal and spatial relations of semantic objects is adopted as low level indexing for video streams. The unsupervised video segmentation method, i.e., the SPCPE algorithm, can obtain the temporal and spatial relations of semantic objects required in the key frame selection approach. In the following subsections, we will first give an overview of the SPCPE algorithm, then briefly describe how to use multimedia ATNs and multimedia input strings to model video key frames. A portion of the soccer video clips is used to demonstrate how video indexing and browsing are modeled by the multimedia ATNs and multimedia input strings.

2.1 Overview of the SPCPE Algorithm

The SPCPE (Simultaneous Partition and Class Parameter Estimation) algorithm is an unsupervised video segmentation method to partition video frames. In the SPCPE algorithm, the partition and the class parameters are treated as random variables. The method for partitioning a video frame starts with an arbitrary partition and employs an iterative algorithm to estimate the partition and the class parameters jointly [Sista99]. Since the successive frames in a video do not differ much, the partitions of adjacent frames do not differ significantly. Each frame is partitioned using the partition of the previous frame as an initial condition so the number of iterations in processing can be greatly reduced. A randomly generated initial partition is used for the first frame since there is no previous frame available.

Suppose we have two classes. Let the partition variable be $\boldsymbol{c} = \{\boldsymbol{c_1}, \boldsymbol{c_2}\}$ and the classes be parameterized by $\theta = \{\theta_1, \theta_2\}$. Also, suppose all the pixel values y_{ij} (in the image data Y) belonging to class k $(k=1,2)$ are put into a vector $\boldsymbol{Y_k}$. Each row of the matrix Φ is given by $(1, i, j, ij)$ and a_k is the vector of parameters $(a_{k0}, \ldots, a_{k3})^T$.

$$y_{ij} = a_{k0} + a_{k1}i + a_{k2}j + a_{k3}ij, \forall(i,j) y_{ij} \in c_k$$

$$\boldsymbol{Y_k} = \Phi a_k$$

$$\hat{a}_k = (\Phi^T \Phi)^{-1} \Phi^T \ \boldsymbol{Y_k} \ a_k$$

We estimate the best partition as that which maximizes the a posteriori probability (MAP) of the partition variable given the image data Y. Now, the MAP estimates of $\boldsymbol{c} = \{\boldsymbol{c_1}, \boldsymbol{c_2}\}$ and $\theta = \{\theta_1, \theta_2\}$ are given by

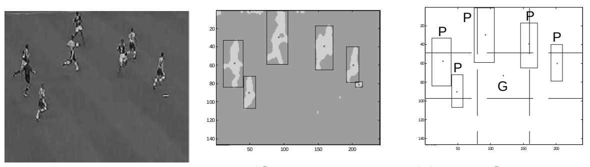
$$(\hat{c}, \hat{\theta}) = Arg \max_{(c,\theta)} P(c, \theta \mid Y)$$
$$= Arg \max_{(c,\theta)} P\ (c, \theta \mid Y) P(c, \theta)$$

Let J(***c*** , *θ)* be the functional to be minimized. With appropriate assumptions, this joint estimation can be simplified to the following form:

$$(\hat{c},\hat{\theta}) = Arg \min_{(c,\theta)} J(c_1,c_2,\theta_1,\theta_2)$$

$$J(c_1,c_2,\theta_1,\theta_2) = \sum_{y_{ij}\in c_1} -\ln p_1(y_{ij};\theta_1) + \sum_{y_{ij}\in c_2} -\ln p_2(y_{ij};\theta_2)$$

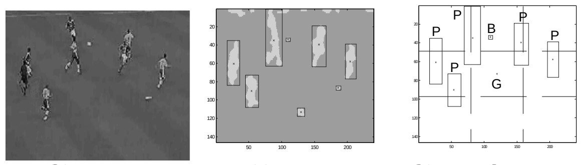
The algorithm starts with an arbitrary partition of the data and computes the corresponding class parameters. Using these class parameters and the data, a new partition is estimated. Both the partition and the class parameters are iteratively refined until there is no further change in them. After the segmentation, the minimal bounding rectangle (MBR) concept in R-tree [Guttman84] is adopted so that each semantic object is bounded by a rectangle (as shown in Figures 1(d), 1(e), and 1(f)). Moreover, the centroid point of each semantic object is mapped to a point object for spatial reasoning.



(a) Frame 1 (d) Partition (g) Bounding Boxes

Multimedia input string for frame 1:

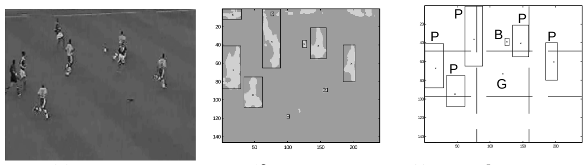
$\boldsymbol{G_1\&P_{10}\&P_{10}\&P_4\&P_4\&P_{19}}$



(b) Frame 5 (e) Partition (h) Bounding Boxes

Multimedia input string for frame 5:

$\boldsymbol{G_1\&P_{10}\&P_{10}\&P_4\&B_4\&P_4\&P_{19}}$



(c) Frame 8 (f) Partition (i) Bounding Boxes

Multimedia input string for frame 8:

$\boldsymbol{G_1\&P_{10}\&P_{10}\&P_{13}\&B_4\&P_4\&P_{19}}$

Figure 1: Key frames 1,5 and 8 for a soccer game video shot and their corresponding multimedia input strings. (a)-(c) are the original frames 1, 5 and 8; (d)-(f) are the segmentation results for frames 1, 5 and 8; (g)-(i) show the segments with bounding boxes and centroids for frames 1, 5 and 8. For each frame, there is a corresponding multimedia input string for it.

2.2 Using Multimedia ATNs and Multimedia Input Strings to Model Video Key Frames

A multimedia ATN can be represented diagrammatically by a labeled directed graph, called a *transition graph*. A multimedia input string is accepted by the grammar if there is a path of transitions which corresponds to the sequence of symbols in the string and which leads from a specified initial state to one of a set of specified final states.

A multimedia ATN can build up a video hierarchy [Chen99]. A video clip can be divided into *scenes*, a *scene* contains a sequential collection of *shots*, and each shot contains some contiguous frames that are at the lowest level in the video hierarchy [Yeo97]. It is advantageous to use several key frames to represent a shot instead of showing all these frames. Key frames play as the indices for a shot. The key frame selection approach proposed in [Chen99] is based on the number, temporal, and spatial changes of the semantic objects in the video frames. Other features may also be possible for the key frame selections, but we focus on the number, temporal, and spatial relations of semantic objects. Therefore, spatio-temporal changes in each shot can be represented by these key frames. For example, in each shot of a soccer game, players may change positions in subsequent frames and the number of players appearing may change at the time duration of the shot.

Figure 1 is an example of how to use multimedia input strings to model video key frames. A soccer game video shot from frames 1 to 10 is used as the inputs of the SPCPE video segmentation method. Frames 1, 5 and 8 are chosen as the key frames by using the key frame selection method introduced in [Chen99]. Figures 1(a)-(c) show the original frames, and Figures 1(d)-(f) give the resulting partitions (segments) for the frames. Since only the ball and the players are important from the content-based retrieval perspective, we use Figure 1(g)-(i) to simplify the segments for each frame. As introduced in [Chen00], one semantic object is chosen as the target semantic object in each video frame and the minimal bounding rectangle (MBR) concept is also used. In order to distinguish the relative positions, twenty-seven numbers multiple players. In the following sections of this paper, we will describe how to use the proposed split algorithm to identify the overlapped players. For this example, each frame is divided into nine subregions.

Each key frame is represented by an input symbol in a multimedia input string and the "&" symbol between two semantic objects is used to denote that the semantic objects appear in the same frame. The subscripted numbers are used to distinguish the relative positions of the semantic objects relative to the target semantic object "ground". So the multimedia input string to represent these three key frames is as follows:

$$\underbrace{(G_1\&P_{10}\&P_{10}\&P_4\&P_4\&P_{19})}_{K_1}\underbrace{(G_1\&P_{10}\&P_{10}\&P_4\&B_4\&P_4\&P_{19})}_{K_2}\underbrace{(G_1\&P_{10}\&P_{10}\&P_{13}\&B_4\&P_4\&P_{19})}_{K_3} \quad [1]$$

As shown above, there are three input symbols that are K_1, K_2 and K_3. The appearance sequence of the semantic objects in an input symbol is based on the spatial locations of the semantic objects in the video frame from left to right and top to bottom. For example, frame 1 is represented by input symbol K_1. $\boldsymbol{G_1}$ indicates that **G** is the target semantic object. $\boldsymbol{P_{10}}$ means the first and the second **P** is on the left of **G**, $\boldsymbol{P_4}$ means that the third and the fourth **P** is above **G**, and $\boldsymbol{P_{19}}$ means that the fifth **P** is on the right of **G**. For frame 5 (K_2), its multimedia input string is almost the same as that of frame 1 except that in which the soccer ball **B** appears above **G**. So the number of semantic objects increases from six to seven. This is an example to show how a multimedia input string can represent the change of the number of semantic object.

Figure 2 is the multimedia ATN for the key frames of the example soccer game video clip. The starting state name for this multimedia ATN is S/. As shown in Figure 2, there are three arcs with arc labels the same as the three input symbols in [1]. For example, the arc symbol K_1 corresponds to the first input symbol of [1]. The different state nodes in the multimedia ATN model the temporal relations of the selected key frames. The relative spatial relations of the semantic objects are modeled by the multimedia input strings.

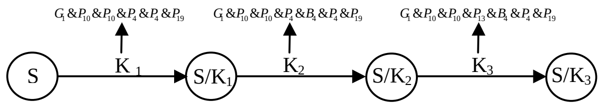

$G_1\&P_{10}\&P_{10}\&P_4\&P_4\&P_{19}$
$G_1\&P_{10}\&P_{10}\&P_4\&B_4\&P_4\&P_{19}$
$G_1\&P_{10}\&P_{10}\&P_{13}\&B_4\&P_4\&P_{19}$
S
K 1
S/K1
K2
S/K2
K3
S/K3

Figure 2: Multimedia Augmented Transition Network and multimedia input strings for modeling the key frames of soccer game video shot *S*.

3. Improve the Segmentation Method Using Backtrack-Chain-Updation Split Algorithm

As mentioned earlier, the current SPCPE algorithm cannot distinguish the overlapped segments within video frames. Since a multimedia database query may involve information about moving objects and their locations in video sequences, it is very crucial to be able to distinguish the overlapped objects. The more accurate the segmentation algorithm is, the more efficient the key frame selection mechanism is (will be shown in section 4). The ability to distinguish the overlapped objects in the video sequences is beneficial for searching/retrieving multimedia information. Here we proposed a *backtrack-chain-updation split algorithm* that can find the split segment (object) and use the information in the current frame to update the previous frames in a backtrack-chain manner. This split algorithm is an enhancement of the SPCPE algorithm to be able to distinguish two separate objects that were overlapped previously. We applied the proposed algorithm on a soccer game video. The experimental results show that our algorithm can recover the separate situation successfully and automatically without any user intervention, which can well comply with the unsupervised feature of the SPCPE algorithm.

3.1 The Proposed Backtrack-Chain-Updation Split Algorithm

In this section, the *backtrack-chain-updation split algorithm* is proposed to enhance the functionality of the current SPCPE algorithm. The proposed backtrack-chain-updation split algorithm considers the situation when overlapping happens between two objects that separate from each other in a later frame. Assume that there are no major differences on the sizes and the shapes of those two objects, and the sizes and shapes of the same object do not change a lot in the consecutive frames.

3.1.1 Object Tracking

In order to detect the situation of overlapping, first we must have the ability to track the objects (segments) within the successive video frames. Note that we use the partition of the previous frame as initial condition in partitioning the current frame. Intuitively, we connect two segments that are spatially the closest in the adjacent frames. Euclidean distance is used here to measure the distance between their centroids. Let ctr_k and ctr_l be the centroids of segments *k* and *l*.

$$dist\,(ctr_k - ctr_l) = \| ctr_k - ctr_l \|_2 \leq \delta$$

Besides using Euclidean distance, we also apply some size restriction into the process of object tracking. If two segments in successive frames represent the same object, the difference between their sizes should not be large.

Let us think about what happens when overlapped segments separate from each other in successive frame. When the split happens, some segment with overlapping in the previous frame may not find its corresponding part in the current frame since either the centroid or the size changes a lot. As a result, there may be some segments in the current frame that cannot be tracked back to the segments in the previous frame. We call these segments as the *unidentified segments*. Then, we try to build up the relationship between those in previous frame and those in current frame based on the information (i.e., the size and position information of those *unidentified segments*) we get. In our algorithm, the concept of MINDIST in [Roussopoulos95] is adopted.

Definition 1: A bounding box B (of dimension 2), will be defined by the two endpoints S and T of its major diagonal:

B=(S, T), where

S=[s_1, s_2] and T=[t_1,t_2] and $s_i \leq t_i$ for i=1,2.

Definition 2: The distance of a point P = [p_1, p_2] from a rectangle B in the same space, denoted *MINDIST(P, B)*, is defined as follows.

$$MINDIST(P,B) = \sum_{i=1}^{2} \left| p_i - r_i \right|^2 \text{ , where}$$

$$r_i = \begin{cases} s_i & \text{if } p_i < s_i \\ t_i & \text{if } p_i > t_i \\ p_i & \text{otherwise} \end{cases}$$

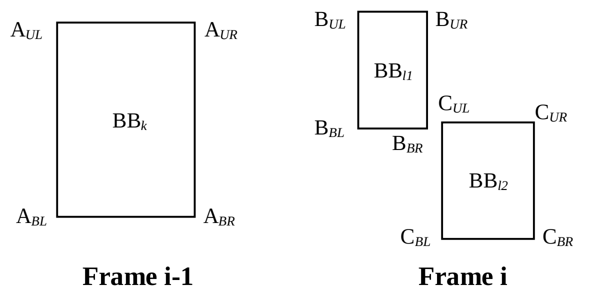
The *MINDIST* is a variation of the classic Euclidean distance applied to a point and a rectangle. When the point is inside the rectangle, the distance between them is zero. Whereas when the point is outside the rectangle, the square of the Euclidean distance between the point and the nearest edge of the rectangle is used. The square of the Euclidean distance is used since fewer and less costly computations are involved.

3.1.2 Backtrack-Chain-Updation Split Algorithm

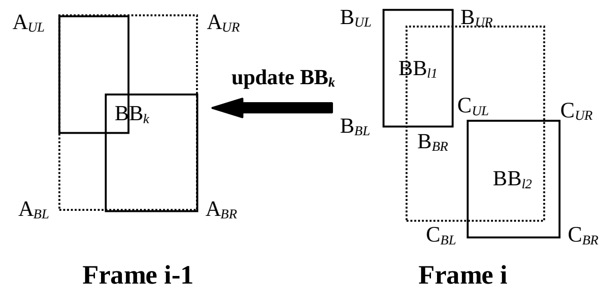
Let ctr_k and BB_k be the centroid and bounding box of segment k in frame i-1, ctr_l and BB_l be the centroid and bounding box of segment l in current frame i. The backtrack-chain-updation split algorithm is given as follows. Figures 3(a)-(c) are used to illustrate the algorithm.

Step 1: *Identify the related segments as many as possible.*
If $dist(ctr_k - ctr_l) = \| ctr_k - ctr_l \|_2 \leq \delta$ **AND** $|size\ (BB_k) - size\ (BB_l)| \leq \beta$, where β is a threshold value for sizes, then segment l in frame i is related to segment k in frame i-1. Mark segments l and k as "Identified". Let segment k be the "parent" of segment l, and let segment l be the "child" of segment k.

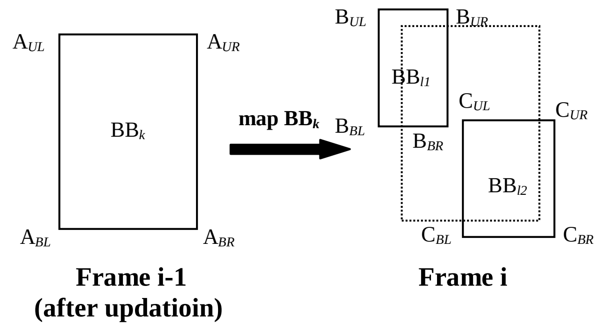
Step 2: *Find out the split segments in current frame*
Select one segment that is not identified in frame i-1. Let its centroid and bounding box be ctr_k and BB_k. Based on the size of BB_k, find out all the unidentified segments in frame i whose bounding box BB_l overlaps with BB_k and $\beta_{\text{Min}} < (size(BB_k)/size(BB_l)) < \beta_{\text{Max}}$. Note that we apply a size restriction to avoid the interference of some small segments ("noise"). Select the first 2 segments (say BB_{l1} and BB_{l2}) whose $MINDIST(ctr_k, BB_{l1})$ and $MINDIST(ctr_k, BB_{l2})$ are the smallest and the second smallest, and mark them as "Identified". Then build up the *parent-child* relationship between BB_k and BB_{l1}, and between BB_k and BB_{l2}.



(a): Bounding boxes of segment k in frame i –1 and segments $l1$, $l2$ in frame i.



(b): Find out the split segments in frame i.



(c): To find out the recovery vertexes for BB_{l1} and BB_{l2} and update BB_k in frame *i-1.*

Figure 3: The basic workflow of the backtrack-chain-updation algorithm.

In Figure 3(b), it shows how to map BB_k into frame i and to find out all the unidentified segments in frame i whose bounding boxes overlap with BB_k. BB_{l1} and BB_{l2} in frame i overlap with the boundary of BB_k, and they are selected as the children segments of segment k in frame i-1. Now there is a parent segment with bounding box BB_k in frame i-1, and there are two children segments with bounding boxes BB_{l1} and BB_{l2} in frame i (as shown in Figure 3(a)). The vertices of each bounding boxes are given with the subscripts "UR", "UL", "BR", "BL" which represent the upper-right, upper-left, bottom-right, and bottom-left, respectively.

Step 3: *Do segmentation on the next frame and get the parameter for size adjustment in step 4*

Once the split segments are identified, we can use the information we get so far to update the parent segment *k*'s bounding box in frame *i*-1. The main idea is to find the *recovery vertex* in parent segment's bounding box, then "paste" the children's bounding boxes into the previous frame without changing their sizes and shapes. Remember we assume that the sizes and shapes of the same object (segment) do not change a lot in the consecutive frames, but we do allow some small changes in length or width of its bounding box. In such cases, sometimes the changes may exceed the updated bounding boxes which results in unsatisfactory recovery results if no adjustment is applied.

Let the current frame be frame *i*, and the previous one be frame *i*-1. In this step, we do segmentation on frame *i*+1 and build up the parent-child relationship between frame *i* and frame *i*+1. For the split segments we just identified in frame *i*, there may exist their children segments in frame *i*+1. If that is true, then the ratios of size changes on length and width for each split segment in frame *i* are calculated. For example, suppose the parent segment in frame *i* is segment *l* (BB_l), and the corresponding child segment in frame *i*+1 is segment *p* (BB_p). Then for segment *l*, its parameters are

$$Width\_Sensitivity_l = \frac{\left|Width_{BB_p} - Width_{BB_l}\right|}{Width_{BB_l}}$$

$$Length\_Sensitivity_l = \frac{\left|Length_{BB_p} - Length_{BB_l}\right|}{Length_{BB_l}}$$

If $Width\_Sensitivity_l > Length\_Sensitivity_l$, we say that segment *l* is more *width-sensitive*. This means the ratio of width changes is more than that of length changes, or the width changes of that segment/object may be more frequent and significant than that of length changes. Otherwise, it is said to be more *length-sensitive*. There is another possibility that we cannot find the corresponding child segment in frame *i*+1 due to object merging or disappearing. In this case,

$$Width\_Sensitivity_l = \frac{Width_{BB_l}}{Length_{BB_l}}$$

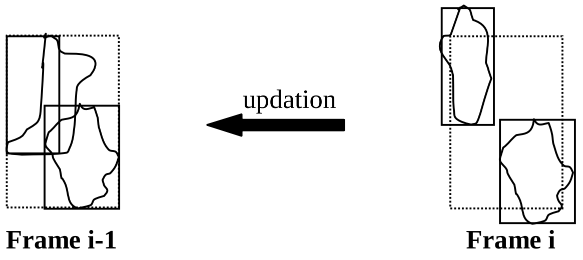
$$Length\_Sensitivity_l = \frac{Length_{BB_l}}{Width_{BB_l}}$$

In step 4, we will show how to use this sensitivity parameter for size adjustment during the backtrack-chain-updating process.

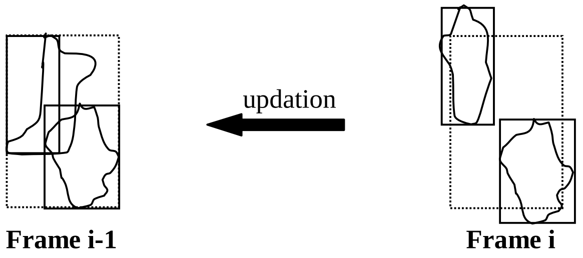
Step 4: *Backtrack and update the previous frames plus size adjustment*

After we find out the split segments (i.e., the children segments BB_{l1} and BB_{l2}) in frame *i*, we can use this information to update the previous frame *i*-1. The goal is to distinguish the separate bounding boxes on the parent segment *k* (BB_k) with overlapping. First check the *MINDIST* from the four vertices of BB_k's to the children's bounding boxes respectively. For BB_{l1}, the vertex *P* on BB_k with the minimum *MINDIST*(*P*, BB_{l1}) is selected as the *recovery vertex* for BB_{l1}. According to this *recovery vertex*, there is a *corresponding vertex* in BB_{l1}. So the next step is to move the *corresponding vertex* to the *recovery vertex*, and to copy the bounding box BB_{l1} within the boundary of BB_k in frame *i*-1. The same procedures are applied to BB_{l2}.

As shown in Figure 3(b), BB_k is mapped to frame i and used to computer *MINDIST*. For example, for BB_{l1}, compute *MINDIST*(A_{vex}, BB_{l1}), where *vex* is one of "UR", "UL", "BR", and "BL". Choose the one with minimum *MINDIST* as the *recovery vertex* for BB_{l1}. Here, A_{UL} is chosen as the recovery vertex for BB_{l1}, and B_{UL} is chosen as the *corresponding vertex*. Similarly, for BB_{l2}, A_{BR} and C_{BR} are selected as the recovery vertex and the corresponding vertex. To update the bounding box BB_k in frame *i*-1, the bounding box BB_{l1} is copied into BB_k with B_{UL} overlapping with A_{UL} and BB_{l2} is copied into BB_k with C_{BR} *overlapping* with A_{BR}. Notice that all the "copy" should be within the boundary of BB_k. By doing so, the updated version of frame *i* with separate bounding boxes can be obtained (as shown in Figure 3(c)).



(a) Update frame *i*-1 without size adjustment



(b) Bounding boxes of frame *i*-1 after size adjustment

Figure 4: Size adjustment after updation for frame *i*-1

In many cases, it seems satisfactory to just copy the children's bounding boxes to their parent's bounding box without any size adjustment. But there are also lots of situations which require necessary size adjustments to reduce the recovery error and achieve better results. For example, as shown in Figure 4(a), we can see separate bounding boxes in frame *i*-1, but the upper bounding box seems a little narrow due to the length change of that bounding box. It seems somewhat unsatisfactory for human eyes and for the purpose of video indexing. The parameters obtained in step 3 will be helpful for size adjustment. In this case, we can decide the upper segment in frame *i*-1 is *length-sensitive*, so what we do is to adjust the length of that bounding box to best fit its shape in frame *i*-1. Figure 4(b) shows the result after size adjustment. It looks pretty good.

This algorithm can be applied to update more previous frames by utilizing the information obtained so far. This is called *backtrack-chain-updation*.

3.2 Results of Applying Backtrack-Chain-Updation Split Algorithm on Video Segmentation

The proposed *backtrack-chain-updation* split algorithm is applied to an example soccer game video. The soccer game video is a gray scale video that shows the part of the game. Each frame is of size 146 rows and 240 columns. We use frames 1 to 10 to illustrate how the split algorithm works.

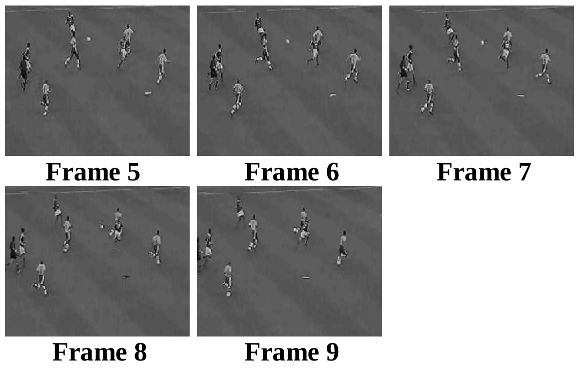


Figure 5: Original soccer video Frames 5 to 9

The soccer game video frames are partitioned using the SPCPE algorithm with 2 classes. Frames 5 to 9 are chosen to demonstrate the effects of the split algorithm. Figure 5 shows the original Frames 5 to 9, while the original segmentation for Frames 5 to 9 are shown in Figure 6(a). Notices that Frame 9 consists of the split segments (objects) and Frames 5 to 8 are the pervious frames that have the overlapped segment. First we use the information available in Frame 9 to adjust the corresponding objects in Frame 8. Then update Frames 7 to 5 one by one in a chain-updation manner. The centroid of each segment is marked with an “x” and the segment is shown with a bounding box around it. Figure 6(b) shows the segmentation results using our backtrack-chain-updation split algorithm without size adjustments, while the segmentation results with size adjustments are given in Figure 6(c).

As can be seen from Figure 6(a), the upper-left two players are overlapped in Frame 8 but are separated in Frame 9. Under the original *SPCPE* algorithm, there is no way to distinguish these two players in Frame 8 even though they are separated in Frame 9. However, after applying our proposed split algorithm on Frame 8, these two objects can be identified successfully and automatically (as shown in Frame 8 on Figure 6(b)). In addition, the same procedure has been applied to any previous frame that has the overlapped segment to identify the separated objects based on the information in the current frame in a backtrack-chain manner. In other words, our proposed split algorithm can distinguish two separate objects that were overlapped previously.

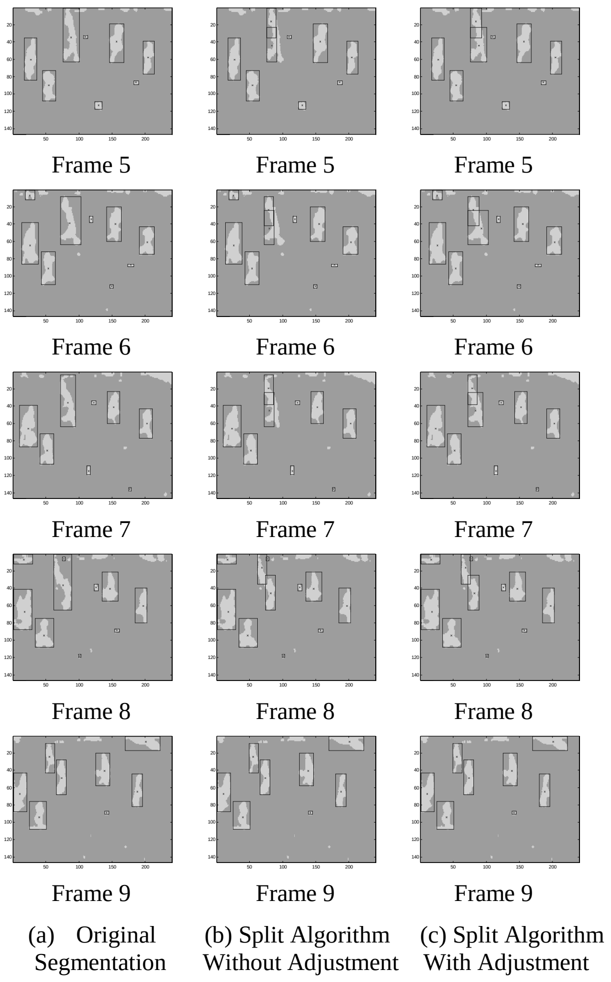


Figure 6: Applying backtrack-chain-updation split algorithm on soccer game video segmentation

However, even though our split algorithm can identify the overlapped objects successfully, the effects of recovery are not such satisfactory since the bounding boxes with fixed sizes do not always fit the changing shapes of those segments (objects) (as shown in Figure 6(b)). Based on the split algorithm, the size adjustments can be applied to refine the segmentation results. As Figure 6(c) shows, the sizes of recovered bounding boxes are not fixed any more. In fact, the size adjustments let the sizes of bounding boxes dynamically fit the changing shapes of segments during the process of backtrack-chain-updation.

4. Conclusions and Future Work

In this paper, a backtrack-chain-updation split algorithm is presented. The proposed split algorithm can distinguish two separate objects that were overlapped previously in video sequences to provide more accurate temporal and spatial relations of the semantic objects. The temporal and spatial relations of the semantic objects are captured by the unsupervised SPCPE video segmentation method, and are modeled by the multimedia ATN model and its multimedia input strings. The proposed split algorithm is an enhancement of the SPCPE method. By obtaining more accurate temporal and spatial relations of the semantic objects from the proposed split algorithm, more accurate multimedia database queries can be answered. A portion of a soccer game video clip is used to demonstrate the effectiveness and accuracy of the proposed split algorithm. The experimental results show that the proposed split algorithm can recover the overlapped situation successfully and automatically without any user invention.

Currently, the proposed split algorithm works on splitting two overlapped semantic objects in video sequences. In our future work, a more general way to solve the object overlapping problem as well as the merging situations so that the video indexing information obtained from segmentation can be more accurate and provide more semantic meaning.

5. References

[Arman94] F. Arman, R. Depommer, A. Hsu, and M.Y. Chiu, "Content-based Browsing of Video Sequences," *ACM Multimedia 94*, pp. 97-103, Aug. 1994.

[Chen99] S.-C. Chen, S. Sista, M.-L. Shyu, and R. L. Kashyap, "Augmented Transition Networks as Video Browsing Models for Multimedia Databases and Multimedia Information Systems," the *11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'99)*, pp. 175-182, November 1999.

[Chen00] S.-C. Chen and R. L. Kashyap, "A Spatio-Temporal Semantic Model for Multimedia Presentations and Multimedia Database Systems," accepted for publication *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[Courtney97] J.D. Courtney, "Automatic Video Indexing via Object Motion Analysis," *Pattern Recognition*, vol. 30, no. 4, pp. 607-625, 1997.

[Day95] Y. F. Day, S. Dagtas, M. Iino, A. Khokhar, and A. Ghafoor, "Object-Oriented Concept Modeling of Video Data," *IEEE Int'l Conference on Data Engineering*, pp. 401-408, March 1995.

[Ferman97] A.M. Ferman, B. Gunsel, and A.M. Tekalp, "Object Based Indexing of MPEG-4 Compressed Video," in *Proc. SPIE: VCIP*, pp. 953-963, vol. 3024, San Jose, USA, February 1997.

[Flickner95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," *IEEE Computer*, Vol. 28, No. 9, pp. 23-31, September 1995.

[Guttman84] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Search," in *Proc. ACM SIGMOD*, pp. 47-57, June 1984.

[Kleene56] S.C. Kleene. *Representation of Events in Nerve Nets and Finite Automata, Automata Studies.* Princeton University Press, Princeton, N.J., pp. 3-41, 1956.

[Oomoto93] E. Oomoto, and K. Tanaka, "OVID: Design and Implementation of a Video Object Database System," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, No. 4, pp. 629-643, August 1993.

[Roussopoulos95] N. Roussopoulos, C. Faloutsos, and T. Sellis, "Nearest Neighbor Queries," *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp. 71-79, 1995.

[Sista99] S. Sista and R. L. Kashyap, "Unsupervised video segmentation and object tracking," in *IEEE Int'l Conf. on Image Processing*, 1999.

[Woods70] W. Woods, "Transition Network Grammars for Natural Language Analysis," *Comm. of the ACM*, 13, October 1970, pp. 591-602.

[Yeo97] B-L Yeo and M.M. Yeung, "Retrieving and Visualization Video," *Comm. of the ACM*, Vol. 40, No. 12, December 1997, pp. 43-52.