Activity Based Video Content Trajectory Representation and Segmentation

Tao Xiang and Shaogang Gong
Department of Computer Science
Queen Mary, University of London, London E1 4NS, UK
{txiang,sgg}@dcs.qmul.ac.uk

Abstract

A novel approach is developed to segment continuous CCTV recordings according to the activities captured in the scene. This approach differs from previous approaches which are mostly based on shot change detection and shot grouping. Video content is represented by constructing a cumulative multi-event histogram over time. An on-line segmentation algorithm is then proposed to detect breakpoints in the video content, which is more robust to noise and computationally much more efficient compared to existing on-line segmentation algorithms. Its performance is compared to that of two off-line algorithms on surveillance videos monitoring an aircraft ramp area. Our experiments demonstrate that this activity based video content representation is superior to a colour histogram based representation for surveillance video segmentation.

1 Introduction

Video segmentation has been studied extensively in the past two decades. Traditionally, a four-layer hierarchical structure is adopted for video structure analysis which consists of a frame layer, a shot layer, a scene layer and a video layer [1]. At the bottom of the structure, continuous image frames taken by a single camera are grouped into shots. A series of related shots are then grouped into a scene. Shot change detection is performed as the first step for video segmentation by most previous approaches [7, 11, 12, 1]. The segmentation of shots and scenes heavily relies on a well-defined feature space usually dominated by colour and motion. For example, in [7], image frames were represented using a holistic colour histogram and the frame difference was exploited to detect shots. This structure is in general valid for constrained, well-structured broadcast videos of news and sports programmes. However, in many cases, especially when the video is taken continuously by a fixed camera *without* script-driven panning and zooming (e.g. in surveillance), global colour and motion information is either highly unreliable or unavailable [4]. More importantly, there is only one shot and any shot-change detection based segmentation approach would be unsuitable.

Recently, Dementhon et al. [2] proposed to represent a video as a high dimensional temporal trajectory based on colour histogram and treat video segmentation as a trajectory breakpoint detection problem. Compared to the thresholding based segmentation algorithms adopted for most previous video segmentation approaches [7, 11], a trajectory breakpoint detection based approach is more robust to local noise at individual frames because segmentation is performed holistically. Various approaches have been proposed to segment a continuous trajectory into segments through breakpoint detection [8, 10, 5].

BMVC 2004 doi:10.5244/C.18.20

Most of them are based on Piecewise Linear Approximation (PLA) or probabilistic graphical models such as Hidden Markov Models (HMMs) [3]. PLA refers to finding the best approximation of a trajectory using straight lines by either Linear Interpolation or Linear Regression. However, the computation is nontrivial when the dimensionality of the trajectory space is high [5], resulting in most of the existing PLA segmentation algorithms only being applied to trajectories in a space with a dimensionality no bigger than 3. On the other hand, HMMs have the potential to be robust to noise and are capable of dynamic time warping. However, a large number of parameters are needed to describe a HMM when the dimensionality of the trajectory space is high. This makes the HMM vulnerable to over-fitting when training data are insufficient.

Most existing trajectory segmentation algorithms are off-line. For the purpose of video segmentation, an on-line algorithm has its distinctive advantage due to the huge amount of data generated and more importantly the real-time nature of some of the applications, such as video surveillance. One of the most popular on-line segmentation algorithms for temporal trajectories is the Sliding Window (SW) algorithm based on the Sliding Window principle [9] and Piecewise Linear Approximation. However, the Sliding Window algorithm tends to over-segment [8, 9, 14]. One possible explanation is that the Sliding Window algorithm lacks a global view of the data since it only 'looks backward' without 'looking forward'. Keogh et al. [8] proposed a new on-line algorithm which combines the bottom-up off-line algorithm [9] with the Sliding Window principle. Nevertheless, this algorithm only works on trajectories with very short segments. It is thus impossible for existing on-line segmentation algorithms to run in real-time on a typical surveillance video sequence which comprises segments lasting over hours.

In this paper, we tackle both the representation and segmentation problems for CCTV surveillance videos. In Section 2, we propose to represent video content holistically in space and over time according to salient events detected automatically in the scene. In Section 3, we address the problem of video segmentation. Three approaches are considered which are suitable for high dimensional video content trajectory segmentation. We first consider a Multi-Observation Hidden Markov Model (MOHMM) which requires less parameters compared to a conventional HMM. It is thus more suitable for multi-dimensional video content trajectory segmentation. For simplicity and fast processing, an algorithm based on Discrete Curve Evolution (DCE) [10] is also evaluated. However, both MOHMM and DCE are off-line algorithms. For on-line segmentation, we propose a novel Forward-Backward Relevance (FBR) algorithm. Compared to a conventional Sliding Window algorithm, FBR is less sensitive to noise and more importantly, can be run in real time. In Section 4, comparative experiments are conducted on segmenting CCTV surveillance videos monitoring an aircraft ramp area. The advantage of our activity based video content representation over the traditional image feature based representation is made clear by our experiment results. The conclusion is drawn in Section 5.

2 Semantic Video Content Representation

We consider an activity based video content representation. Salient events are detected and classified automatically in the scene. The semantics of video content are considered to be best encoded in the occurrence of such events and the temporal correlations among them.

Detecting and Classifying Scene-Events

We wish to automatically detect scene-events that reflect significant changes in the image

over time *without* manual labelling or top-down hypothesising. To this end, an approach is adopted to detect pixel-level changes and group them into events for classification [16].

First, an adaptive Gaussian mixture background model [15] is adopted to detect foreground pixels which are modelled using Pixel Change History (PCH) [16]. Second, the foreground pixels in a vicinity are grouped into a blob using the connected component method. Each blob with its average PCH value greater than a threshold is then defined as a scene-event. A detected scene-event is represented as a 7-dimensional feature vector $\mathbf{v} = [\bar{x}, \bar{y}, w, h, R_f, M_p x, M_p y]$ where $(\bar{x}, \bar{y})$ is the centroid of the blob, (w, h) is the blob dimension, R_f is the filling ratio of foreground pixels within the bounding box associated with the blob, and $(M_p x, M_p y)$ are a pair of first order moments of the blob represented by PCH.

Classification is then performed in the 7D scene-event feature space over a collection of training videos using a Gaussian Mixture Model. The number of scene-event classes captured in the videos is determined by automatic model order selection based on Schwarz's Bayesian Information Criterion (BIC) [13, 16]. An example of event detection and classification is shown in Figure 1.

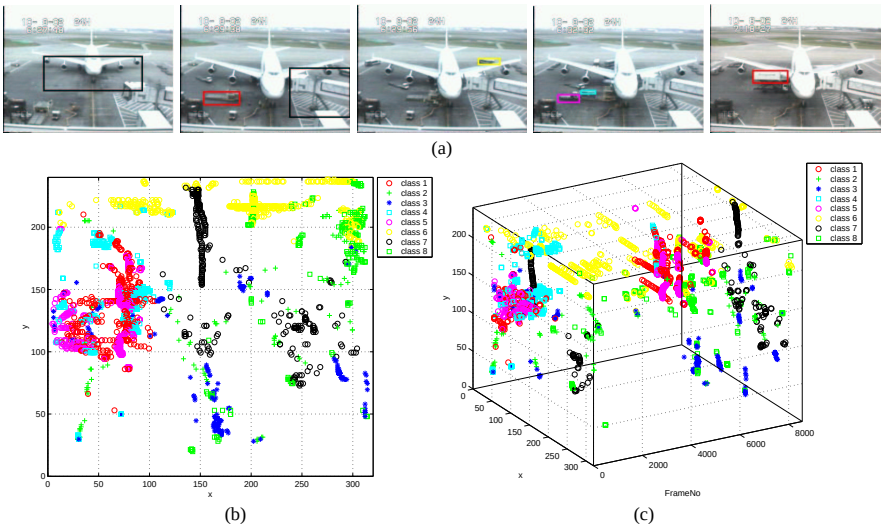


Figure 1: Event detection and classification in an aircraft docking scene video. Eight classes of events are detected automatically, each of which corresponds to different movement patterns in the image frame. (a) Events are detected and classified into different classes of events which are highlighted using bounding boxes in different colours. Spatial and temporal distribution of events of different classes are illustrated in (b) and (c) respectively, with centroids of different classes of events depicted using different colours.

Constructing a Scene Vector

The classified events can be considered as a 'snapshot' of the activities captured in the scene. To hide the image feature information and focus on the semantic video content, a scene vector is constructed for each frame of a video or live recording. A scene vector $\mathbf{sv}_t$ for a video frame t is defined as:

$$\mathbf{sv}_t = \left[s_t^1, ..., s_t^k, ..., s_t^K\right] \tag{1}$$

where K is the number of event classes detected over the training videos. The value of s_t^k

is the number of events of the k^{th} event class detected in the frame t.

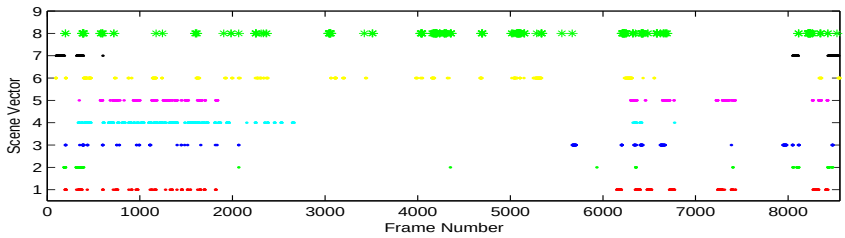


Figure 2: The example sequence shown in Figure 1 represented by scene vector over time. $K = 8$ for this sequence. s_t^k is depicted by a dot in colour only when $s_t^k > 0$. There are frequent but short inactivity breakups within activities (between frame 1 and frame 2000) the long inactivities between activities (between frame 3000 and frame 6000).

This scene vector gives a description of 'what is happening' in the scene through the label of detected events. It is thus a meaningful and concise representation of the video content at the semantic level (see Figure 2 for an example). However, directly using this scene vector to detect video content changes can cause problems. The value of the scene vector $\mathbf{sv}_t$ can become $\mathbf{0}$ (i.e. absent of any event at a given frame) frequently throughout the video sequence (see Figure 2). This can either be caused by the frequent but short inactivity breakups within activities or the long inactivities between activities. Each 'coming to zero' is reflected as a dramatic change in the value of $\mathbf{sv}_t$ due to the discrete nature of s_t^k. Those changes that correspond to real changes of video content can thus easily be overwhelmed by changes caused by the inactivity breakups within activities, which makes temporal segmentation of the video difficult.

Representing Video Content Over Time

To overcome this problem, let us now reconsider the video content being represented by a cumulative histogram of $\mathbf{sv}_t$ from frame 1 to the current frame t. The k^{th} element of the new scene vector (denoted as $\widetilde{\mathbf{sv}}_t$) is computed as:

$$\widetilde{s}_t^k = \sum_{i=1}^{t} s_i^k \tag{2}$$

The value of each element of $\widetilde{\mathbf{sv}}_t$ will increase monotonically with time (see Figure 3). Compared to the scene vector representation $\mathbf{sv}_t$, the short inactivity breakups at individual frames have little impact on the video content. It thus becomes easier to detect breakpoints that correspond to significant change of video content.

3 Temporal Segmentation of Surveillance Videos

It has been shown in the preceding section that a cumulative histogram of the scene vector can represent the video content at the semantic level over time and is capable of capturing video content changes despite variations in activity durations and inactivity occurrences. After mapping a video sequence into a cumulative scene vector histogram trajectory[1], the breakpoints on this trajectory correspond to the video content change points. We thus consider the video segmentation problem as a temporal video content trajectory breakpoint detection problem.

[1] More precisely, it is a video polyline due to the discrete nature of the image frames.

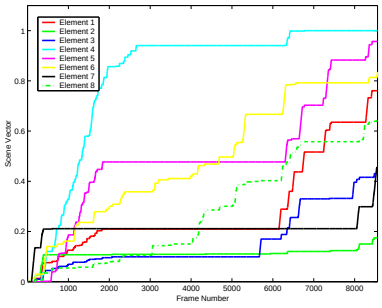
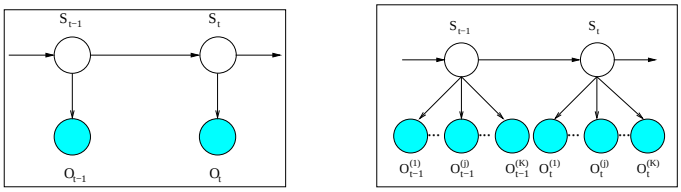


Figure 3: The 8 elements of $\widetilde{\mathbf{sv}}_t$ over time for the example sequence shown in Figure 1. Each element of $\widetilde{\mathbf{sv}}_t$ is normalised to have a value range of $[0, 1]$.

Multi-Observation Hidden Markov Model (MOHMM)



(a) HMM (b) MOHMM

Figure 4: HMM and MOHMM with the observation nodes illustrated as shaded circles and hidden nodes as clear circles.

We first consider detecting breakpoints on a temporal video content trajectory using a Multi-Observation Hidden Markov Model (MOHMM) (Figure 4 (b)). $O_t^{(j)}$ ($j \in \{1, ..., K\}$) in Figure 4 (b) is a 1D random variable corresponding to one of the K elements of $\widetilde{\mathbf{sv}}_t$ and hidden states variable S_t in the network corresponds to the video content. The conditional probability distributions (CPDs) of each observation variable are Gaussian for each of the N_s states of its parent node. The number of parameters needed for a HMM (Figure 4(a)) and a MOHMM are ${N_s}^2 + (K^2 + 3K + 2)N_s/2 - 1$ and ${N_s}^2 + (2K + 1)N_s - 1$ respectively when they are used to model a K-dimensional video content trajectory. Notice that O_t in Figure 4(a) is K-dimensional corresponding to $\widetilde{\mathbf{sv}}_t$. Needing less parameters means that a MOHMM is less vulnerable to over-fitting compared to a HMM.

The remaining problem is to automatically determine N_s, the number of hidden states which corresponds to the number of video segments. To this end, we adopt Schwarz's Bayesian Information Criterion (BIC) [13] to automatically determine N_s. For a model $\mathbf{m}_i$ parameterised by a C_i-dimensional vector $\mathbf{\Theta}_{\mathbf{m}_i}$, the BIC is defined as:

$$BIC = -2 \log L(\mathbf{\Theta}_{\mathbf{m}_i}) + C_i \log N \tag{3}$$

where $L(\mathbf{\Theta}_{\mathbf{m}_i})$ is the maximal likelihood under $\mathbf{m}_i$, C_i is the dimensionality of the parameters of $\mathbf{m}_i$ and N is the size of the dataset. For the MOHMM, $L(\mathbf{\Theta}_{\mathbf{m}_i})$ can be written as:

$$-2 \log \left\{ \sum_{S_t} \left\{ P(S_1) \prod_{t=2}^{T} P(S_t | Pa(S_t)) \prod_{t=1}^{T} \prod_{j=1}^{K} P\left(O_t^{(j)} | Pa(O_t^{(j)})\right) \right\} \right\}$$

where $Pa(S_t)$ are the parents of S_t and similarly, $Pa(O_t^{(i)})$ for observations. The search for the optimal N_s that produces the minimal BIC value involves parameter learning. More specifically, for each candidate state number, the corresponding parameters are learned iteratively using the Expectation-Maximisation (EM) algorithm. The E step, which involves the inference of hidden states given the parameters estimated in the last M step, can be implemented using an exact inference algorithm such as the junction tree algorithm [6]. After parameter learning, the BIC value can be computed using Equation (3) where $L(\mathbf{\Theta}_{\mathbf{m}_i})$ has been obtained from the final M step of EM for parameter learning.

Note that in theory the MOHMM can also be adopted for on-line segmentation provided that the occurrences of different activities and inactivities follow a certain temporal order. This is due to the cumulative nature of $\widetilde{\mathbf{sv}}_i$. However, there are considerable variations in both the temporal order and durations of activities and inactivities in a surveillance video. Thus the MOHMM is used only for off-line segmentation in our case.

Discrete Curve Evolution (DCE)

The inference and learning of a MOHMM is computational expensive when a video content trajectory is long and has a high dimensionality. For simplicity and fast processing, a computationally more efficient algorithm is evaluated which is based on Discrete Curve Evolution (DCE). DCE was originally proposed for 2-D shape decomposition [10]. It can readily be extended for detecting breakpoints of high-dimensional video content trajectories. The algorithm starts with the finest possible segmentation (i.e. $T-1$ segments for a trajectory of length T) and progresses with two segments being merged at each step. Specifically, the cost of merging all adjacent pairs of segments are computed and the pair with the minimal cost is merged. This process is performed iteratively until a stopping criterion is met. DCE computes the merging cost as a relevance measure R which is computed for every vertex v and depends on v and its two neighbour vertices u and w:

$$R(u,v,w) = d(v,u) + d(v,w) - d(u,w) \tag{4}$$

where function $d(.)$ can be any distance or similarity measures such as Euclidean distance. In each iteration, the vertex with minimal R is deleted until the minimal R exceeds certain threshold or only the user-specified number of vertices are left in the trajectory. The remaining vertices correspond to the breakpoints in the sequence. Although DCE is similar in spirit to the bottom-up segmentation algorithm based on Piecewise Linear Approximation (PLA) in time series data analysis [8], it is much faster compared to a bottom-up algorithm using either Linear Interpolation or Linear Regression, especially when the dimensionality of the trajectory space is high.

Forward-Backward Relevance (FBR)

Both MOHMMs and DCE are off-line algorithms. We propose a novel on-line algorithm based on the Forward-Backward Relevance (FBR) principle, as follows:

1. Set the starting point of the first segment at the first data point of the trajectory.
2. Move along the trajectory for the minimal segment length L_{min};
3. Compute the relevance of the current vertex as:
$R(currentVertex) = R(startingPoint, currentVertex, currentVertex + L_{min})$;
4. **If** the relevance $R(currentVertex)$ is smaller than a threshold Th, move to the next vertex and **goto** 3;
5. **Else** the current vertex is a breakpoint; set the starting point of the next segment at the current vertex and **goto** 2.

Two parameters, L_{min} and Th need to be determined. We perform DCE on the training data and L_{min} is set as half of the length of the shortest segment in the training data and Th is computed as the average value of $R(currentVertex)$ of the breakpoints detected by DCE. There are two important advantages of our algorithm compared to a conventional Sliding Window algorithm: (1) For each new data point, instead of fitting the trajectory to a straight line, FBR computes $R(currentVertex)$ which involves only computing 3 distances. The computation complexity is *independent* of the segment length and thus much lower than the Sliding Window algorithm and its variations; (2) FBR algorithm is more robust to noise because it has a global view of the trajectory by looking both forward and backward.

4 Experiments

Experiments were conducted on the representation and segmentation of CCTV surveillance videos monitoring an aircraft ramp area. A fixed CCTV analogue camera took continuous recordings. After digitisation, the final video sequences have a frame rate of 2Hz. Note that it is not uncommon to have such an extremely low frame rate for CCTV surveillance videos, which makes the video segmentation problem very challenging. Each image frame has a size of 320×240 pixels. Our database for the experiments consists of 7 sequences of aircraft docking lasting from 6470 to 17262 frames per sequence (around 50 to 140 minutes of recording), giving in total 72776 frames of video data that cover different times of different days under changing lighting conditions, from early morning, midday to late afternoon. They are referred as video1 to video 7 respectively. Among the 7 videos, videos 1 to 6 follow the typical video structure with cargo services performed before catering services while video 7 has a different video structure with cargo services performed after catering services. Around 15 different activities take place in the scene. Among them, seven activities are visually detectable. In the following we present results on (1) comparing a cumulative scene vector based video content representation with a colour histogram based one, and (2) comparative performance evaluation on video segmentation.

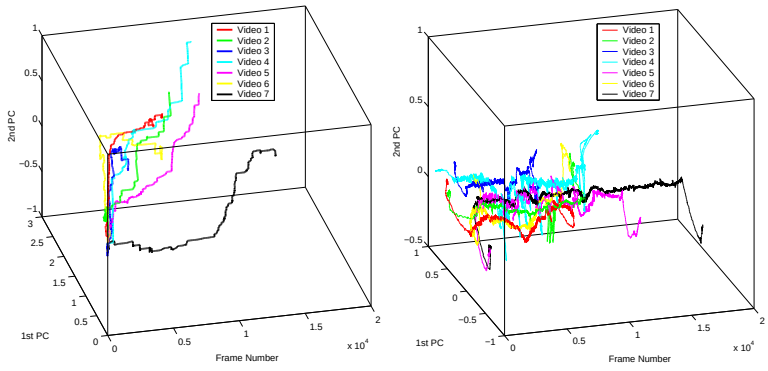


Figure 5: (a) and (b) show the first 2 principal components of $\widetilde{\mathbf{sv}}_t$ over time and colour histogram for seven different aircraft docking videos respectively. Note that PCA is just for visualisation here. Each element of $\widetilde{\mathbf{sv}}_t$ and colour histogram is normalised to have a value range of $[0, 1]$.

Video content representation — Eight classes of events were automatically detected

and classified from the aircraft docking video sequences. They were caused by the movements of the docking aircraft and various ground vehicles such as catering trucks, cargo trucks and cargo lifts (see Figure 1 for an example). A scene vector $\mathbf{sv}_t$ was constructed for each frame (Equation (1)) and its cumulative histogram $\widetilde{\mathbf{sv}}_t$ (Equation (2)) was used to represent the video content. Figure 5 (a) shows the evolution of the first two principal components of $\widetilde{\mathbf{sv}}_i$ for the seven different sequences. It can be seen that the video content trajectory of video 7 is distinctively different from those of the other 6 videos. For comparison, the same video sequences were also represented using colour histograms. We defined 4 histogram bins for each of the three colour channels of the RGB colour space. Separate RGB colour histograms were concatenated into one with 12 components in each frame. Figure 5 (b) illustrates the evolution of the first two principal components of the colour histogram representation over time from the same seven sequences as Figure 5 (a). It is evident from Figure 5 (b) that the colour histogram video content trajectories are less smooth compared to the video content trajectories based on cumulative scene vector histograms. It also appears that only the arrival and departure of the aircraft are captured well by the colour histogram change (see the beginning and ending part of the trajectories). It can be seen that the trajectory corresponding to video 7 is similar to other trajectories under this colour histogram representation.

Video segmentation — The 7 videos were first manually segmented into activities and inactivities to give the ground truth of the breakpoints. The number of segments in each sequence was between 6 and 11. We then compared the performance of the off-line Discrete Curve Evolution (DCE) and Multi-Observation Hidden Markov Model (MOHMM) algorithms and our on-line Forward-Backward Relevance (FBR) algorithm on the videos represented by scene vector $\widetilde{\mathbf{sv}}_i$ (Figure 5 (a)). Three videos were used as training data and the others as test data. This was repeated four times with different videos as the training and test data. Notice that although the videos are illustrated in PCA space in Figures 5(a), the 8-dimensional cumulative scene vector histograms were used directly for our segmentation algorithms.

	DCE	MOHMM	FBR
Correct Detection Rate	77.8 %	79.2 %	72.6%
False Positives	28.9 %	23.0 %	47.4 %

Table 1: Segmentation results on cumulative scene vector histogram trajectories.

The performance of segmentation was measured by the correct detection rate and false positive rate of breakpoint detection. A breakpoint detected within 100 frames of the true breakpoint was regarded as being correctly detected. Altogether, there were 135 true breakpoints in the test datasets. The distance measure in Equation (4) was Euclidean for our experiments using DCE and FBR. The FBR algorithm was run at 6Hz on an Athlon 1.5GHz platform with most of the computational load coming from the adaptive Gaussian mixture background model. The average speeds for the MOHMM and DCE were 0.31 and 1.29 second per frame respectively on the same platform. Table 1 shows that in general, satisfactory results were obtained by all three algorithms[2]. The FBR algorithm achieved slightly worse detection rate compared to the off-line algorithms and tended to

[2]The detection rate was computed by dividing the correct detections by the total number of true breakpoints instead of the total number of candidate breakpoints.

over-segment the video sequences. It is also noted that the starting and ending points of the frontal catering service were often mis-detected by all three algorithms. When the catering truck moved into position very quickly, the resultant change in the video content trajectory was similar to those caused by passing vehicles which were not involved in any activity. This type of change would not be picked up by the segmentation algorithms. On the other hand, the long inactivity breakups caused by the lack of movements during the frontal catering service were the main reason for false positives.

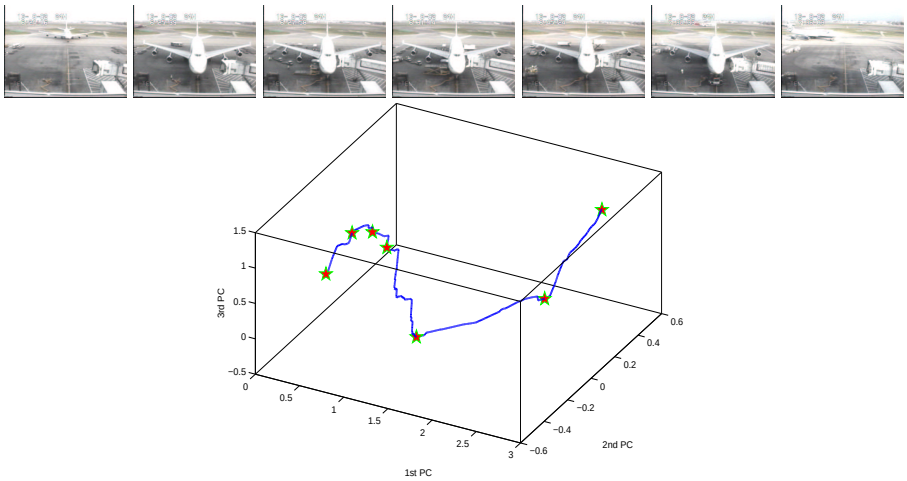


Figure 6: An example breakpoint detection result by FBR algorithm.

Figure 6 shows an example of FBR breakpoint detection results. The bottom plot in a 3D PCA space shows video 2 represented by the scene vector $\widetilde{\mathbf{sv}}_i$ with the 7 detected breakpoints overlapped on the video content trajectory (the trajectory evolves from left to right). The top row of Figure 6 shows the frames corresponding to the detected breakpoints. It was found that the fourth breakpoint was a false positive which was caused by a long inactivity breakup during the frontal cargo service. A breakpoint between the fifth and sixth detected breakpoints was mis-detected which corresponds the ending point of a frontal catering service. This was caused by the quick departure of the catering truck.

We also performed DCE, MOHMM and FBR on the colour histogram based video content representation following the same experimental procedure. In Table 2, the results demonstrate that all three algorithms failed to produce a meaningful segmentation of the same 7 aircraft docking videos. In particular, only the arrival and departure of the aircraft were detected reliably. The changes in the colour histogram caused by the start and end of other activities were small and can easily be overwhelmed by image noise and illumination changes frequently encountered in a typical outdoor environment. This resulted in the low detection rate and high false positive rate.

	DCE	MOHMM	FBR
Correct Detection Rate	30.4%	27.6%	28.9%
False Positives	76.3%	52.6%	65.9%

Table 2: Segmentation results on colour histogram trajectories.

5 Conclusion

In this paper, we have presented a novel approach for representing and segmenting the content of CCTV surveillance videos according to the activities captured in the scene. The video content is represented by constructing a cumulative multi-event histogram over time. An on-line Forward-Backward Relevance (FBR) algorithm was developed to detect breakpoints in the video content and segment a continuous video into activities, which is robust to noise and can be run in real-time. Its performance was compared with those of two off-line algorithms, Multi-Observation Hidden Markov Model (MOHMM) and Discrete Curve Evolution (DCE) on surveillance videos monitoring an aircraft ramp area. Our experiments demonstrate that this activity based video content representation is superior to a colour histogram based representation for surveillance video segmentation.

References

[1] N. Babaguchi, Y. Kawai, and T. Kitahashi. Event based indexing of broadcasting sports video by intermodal collaboration. *IEEE Transactions on Multimedia*, 4(1):68–75, 2002.

[2] D. DeMenthon, L. Latechi, A. Rosenfeld, and M. Stuckelberg. Relevance ranking of video data using hidden markov model distances and polygon simplification. In *Advances in Visual Information Systems*, 2000.

[3] X. Ge and P. Smyth. Segmental semi-markov models for endpoint detection in plasma etching. In *AEC/APC Symposium XII*, 2000.

[4] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *ICCV*, pages 742–749, 2003.

[5] P. Heckbert and M. Garland. Survey of ploygonal surface simplification algorithms. In *International Conference on Computer Graphics and Interactive Techniques*, 1997.

[6] C. Huang and A. Darwiche. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

[7] J.R. Kender and B.L. Yeo. Video scene segmentation via continuous video coherence. In *CVPR*, pages 367–373, 1998.

[8] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *ICDE*, pages 289–296, 2001.

[9] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

[10] L. Latecki and R. Lakamper. Convexity rule for shape decomposition based on discrete contour evolution. *CVIU*, 73:441–454, 1999.

[11] C.W. Ngo, T.C. Pong, and H.J. Zhang. Motion-based video representation for scene change detection. *IJCV*, 50(2):127–142, 1998.

[12] Y. Rui, S. Huang, and S. Mehrota. Exploring video structures beyond the shots. In *IEEE International Conference on Multimedia Computing and Systems*, pages 237–240, 1998.

[13] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[14] H. Shatkay and S. Zdonik. Approximate queries and representations for large data sequences. In *ICDE*, pages 546–553, 1996.

[15] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *PAMI*, 22(8):747–758, August 2000.

[16] T. Xiang, S. Gong, and D. Parkinson. Autonomous visual events detection and classification without explicit object-centred segmentation and tracking. In *BMVC*, pages 233–242, 2002.