

# Multimedia Data Mining Framework for Raw Video Sequences

JungHwan Oh, JeongKyu Lee, Sanjaykumar Kote, and Babitha Bandi

Department of Computer Science and Engineering  
University of Texas at Arlington  
Arlington, TX 76019-0015 U. S. A.  
e-mail: {oh, jelee, kote, bandi}@cse.uta.edu

**Abstract.** We extend our previous work [1] of the general framework for video data mining to further address the issue such as how to mine video data, in other words, how to extract previously unknown knowledge and detect interesting patterns. In our previous work, we have developed how to segment the incoming raw video stream into meaningful pieces, and how to extract and represent some feature (i.e., motion) for characterizing the segmented pieces. We extend this work as follows. To extract motions, we use an accumulation of quantized pixel differences among all frames in a video segment. As a result, the accumulated motions of segment are represented as a *two dimensional matrix*. We can get very accurate amount of motion in a segment using this matrix. Further, we develop how to capture the *location* of motions occurring in a segment using the same matrix generated for the calculation of the amount. We study how to cluster those segmented pieces using the features (the amount and the location of motions) we extract by the matrix above. We investigate an algorithm to find whether a segment has normal or abnormal events by clustering and modeling normal events, which occur mostly. In addition to deciding normal or abnormal, the algorithm computes *Degree of Abnormality* of a segment, which represents to what extent a segment is distant to the existing segments in relation with normal events. Our experimental studies indicate that the proposed techniques are promising.

**KEYWORDS:** Multimedia Data Mining, Video Segmentation, Motion Extraction, Video Data Clustering

## 1 Introduction

Data mining, which is defined as the process of extracting previously unknown knowledge, and detecting interesting patterns from a massive set of data, has been a very active research. As results, several commercial products and research prototypes are even available nowadays. However, most of these have focused on corporate data typically in alpha-numeric database.

Multimedia data mining has been performed for different types of multimedia data; image, audio and video. An example of image data mining is *CONQUEST* [2] system that combines satellite data with geophysical data to discover patterns in global climate change. The *SKICAT* system [3] integrates techniques for image processing and data classification in order to identify 'sky objects' captured in a very large satellite picture set. The *MultiMediaMiner* [4–6] project has constructed many image understanding, indexing and mining techniques in digital media. Some advanced techniques can be found in mining knowledge from spatial [7] and geographical [8] databases.

An example of video and audio data mining can be found in *Mining Cinematic Knowledge* project [9] which creates a movie mining system by examining the suitability of existing concepts in data mining to multimedia, where the semantic content is time sensitive and constructed by fusing data obtained from component streams. A project [10, 11] analyzing the broadcast news programs has been reported. They have developed the techniques and tools to provide news video annotation, indexing and relevant information retrieval along with domain knowledge in the news programs. A data mining framework in audiovisual interaction has been presented [12] to learn the synchronous pattern between two channels, and apply it to speech driven lip motion of facial animation system. The other example is a system [13] focusing on the echocardiogram video data management to exploit semantic querying through object state transition data modeling and indexing scheme. We can find some multimedia data mining frameworks [14–16] for traffic monitoring system. EasyLiving [17, 18] and HAL [19] projects are developing smart spaces that can monitor, predict and assist the activities of its occupants by using ubiquitous tools that facilitate everyday activities.

As mentioned above, there have been some efforts about video data mining for movies, medical videos, and traffic videos. Among them, the developments of complex video surveillance systems [20] and traffic monitoring systems [15, 16, 21–23] have recently captured the interest of both research and industrial worlds due to the growing availability of cheap sensors and processors at reasonable costs, and the increasing safety and security concerns. As mentioned in the literature [14], the common approach in these works is that the objects (i.e., person, car, airplane, etc.) are extracted from video sequences, and modeled by the specific domain knowledge, then, the behavior of those objects are monitored (tracked) to find any abnormal situations. What are missing in these efforts are first, how to index and cluster these unstructured and enormous video data for real-time processing, and second, how to mine them, in other words, how to extract previously unknown knowledge and detect interesting patterns.

In this paper, we extend our previous work [1] of the general framework for video data mining to further address the issues discussed above. In our previous work, we have developed how to segment the incoming video stream into meaningful pieces, and how to extract and represent some feature (i.e., motion) for characterizing the segmented pieces. We extend this work as follows.

- To extract motions, we use an accumulation of quantized pixel differences among all frames in a segment [1, 24]. As a result, the accumulated motions of segment are represented as a *two dimensional matrix*. By this way, we can get very accurate amount of motion in a segment. Further, we develop how to capture the *location* of motions occurring in a segment using the same matrix generated for the calculation of the amount.
- We study how to cluster these segmented pieces using the features (the amount and the location of motions) extracted above.
- We investigate an algorithm to find whether a segment has normal or abnormal events by clustering and modelling normal events which occur the most. In addition to deciding normal or abnormal, the algorithm computes *Degree of Abnormality* ( $\Psi$ ) of a segment, which represents to what extent a given segment is distant to the existing segments with normal events.

The main contributions of the proposed work can be summarized as follows.

- The proposed technique to compute motions is very cost-effective because an expensive computation (i.e., optical flow) is not necessary. The matrices representing motions are showing not only the amounts but also the exact locations of motions. Therefore, we can get more accurate and richer information of motion contents of segment. Because the motions are represented as a matrix, comparison among segments is very efficient and scalable.
- Many researches [25–28] have tried to find abnormal events by modelling abnormal events. Most of them define some specific abnormal event, and try to detect it in video sequences. However, a same specific abnormal event can occur in many different ways, and it is not possible to predict and model all abnormal events. To find the abnormality, our approach uses the normal events which occur everyday and easy to obtain. We do not have to model any abnormal event separately. Therefore, unlike the others, our approach can be used for any video surveillance sequences to distinguish normal and abnormal events.

The remainder of this paper is organized as follows. In Section 2, to make the paper self-contained, we describe briefly the video segmentation technique relevant to this paper, which has been proposed in our previous work [1, 24]. How to capture the *amount* and the *location* of motions occurring in a segment, how to cluster those segmented pieces, and how to model and detect normal events are discussed in Section 3. The experimental results are discussed in Section 4. Finally, we give our concluding remarks in Section 5.

## 2 Incoming Video Segmentation

In this section, we briefly discuss the details of the technique in our previous work [1] to group the incoming frames into semantically homogeneous pieces by real time processing (we called these pieces as ‘segments’ for convenience).

To find segment boundary, instead of comparing two consecutive frames (Figure 1(a)) which is the most common way to detect shot boundary [29–33], we

compare each frame with a *background* frame as shown in Figure 1(b). A background frame is defined as a frame with only non-moving components. Since we can assume that the camera remains stationary for our application, a background frame can be a frame of the stationary components in the image. We manually select a background frame using a similar approach as in [14, 21, 34]. The solid graph in the top of Figure 2 shows the color histogram difference of background with each frame in the sequence. The differences are magnified so that segment boundaries can be found more clearly. The algorithm to decompose a video sequence into meaningful pieces (segments) is summarized as follows. The Step.1 is a preprocessing by off-line processing, and the Step.2 through 5 are performed by on-line real time processing. Note that since this segmentation algorithm is generic, the frame comparison can be done by any technique using color histogram, pixel-matching or edge change ratio. We chose a simple pixel matching technique for illustration purpose.

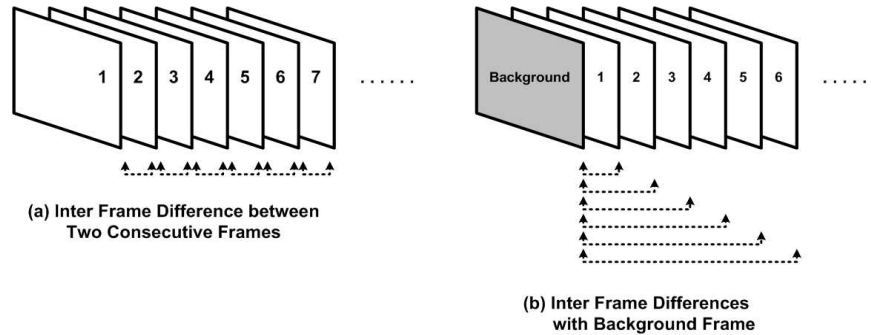
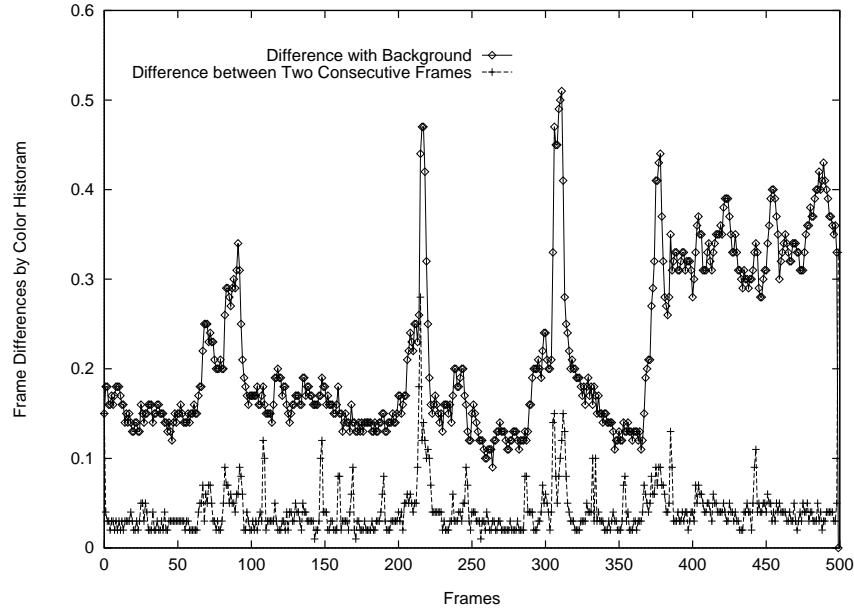


Fig. 1. Frame Comparison Strategies

- Step.1: A background frame ( $F^B$ ) is extracted from a given sequence as preprocessing, and its color space of each frame is quantized (i.e., from 256 to 64 or 32 colors) to reduce noises (false detection of motion which is not actually motion but detected as motion).
- Step.2: Each frame ( $F^k$ ) arriving to the system is also quantized in the same rate used to quantize the background in the previous step.
- Step.3: Compare all the corresponding (same position of) pixels of two frames (background and each frame). Compute the difference ( $D^k$ ) between the background ( $F^B$ ) and each frame ( $F^k$ ) as follows. Assume that the size of frame is  $c \times r$  pixels. Note that the value of  $D^k$  is always between zero and one.

$$D^k = \frac{\text{Total number of pixels in which their colors are different}}{c \times r} \quad (1)$$

- Step.4: Classify  $D^k$  into 10 different categories based on its value. Assign a corresponding category number ( $C_k$ ) to the frame  $k$ . We use 10 categories



**Fig. 2.** Two Frame Comparison Strategies

for illustration purpose, but this value can be changed properly according to the contents of video. The classification is stated below.

- Category 0 :  $D^k < 0.1$
  - Category 1 :  $0.1 \leq D^k < 0.2$
  - Category 2 :  $0.2 \leq D^k < 0.3$
  - Category 3 :  $0.3 \leq D^k < 0.4$
  - Category 4 :  $0.4 \leq D^k < 0.5$
  - Category 5 :  $0.5 \leq D^k < 0.6$
  - Category 6 :  $0.6 \leq D^k < 0.7$
  - Category 7 :  $0.7 \leq D^k < 0.8$
  - Category 8 :  $0.8 \leq D^k < 0.9$
  - Category 9 :  $D^k \geq 0.9$
- Step.5: For real time on-line processing, a temporary table such as Table 1 is maintained. To do this, and to build a hierarchical structure from a sequence, compare  $C_k$  with  $C_{k-1}$ . In other words, compare the category number of current frame with the previous frame. We can build a hierarchical structure from a sequence based on these categories which are not independent from each other. We consider that the lower categories contain the higher categories as shown in Figure 3. For example, one segment  $A$  of Cat. #1 starts with Frame # $a$  and ends with Frame # $b$ , and the other segment  $B$  of Cat. #2 starts with Frame # $c$  and ends with Frame # $d$ , then it is possible that  $a < c < d < b$ . In our hierarchical segmentation, therefore, finding segment boundaries means finding category boundaries in which we find a starting

Segment No.	Starting Frame No.	Ending Frame No.	Segment Length	Cat. ( $C_k$ )	Total Motion (TM)	Avg. Motion (AM)

Table 1. Segmentation Table

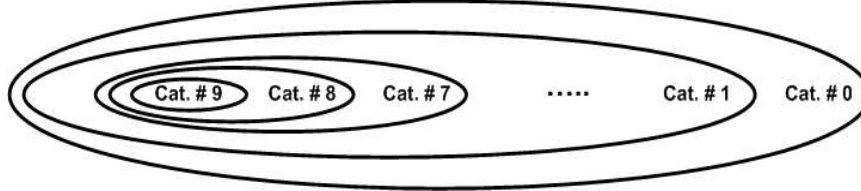


Fig. 3. Relationships (Containments) among Categories

frame ( $S_i$ ) and an ending frame ( $E_i$ ) for each category  $i$ . The following algorithm shows how to find these boundaries.

- If  $C_{k-1} = C_k$ , then no segment boundary occurs, so continue with the next frame.
- Else if  $C_{k-1} < C_k$ , then  $S_{C_k} = k$ ,  $S_{C_{k-1}} = k$ , ...  $S_{C_{k-1}+1} = k$ . The starting frames of category  $C_k$  through  $C_{k-1} + 1$  are  $k$ .
- Else, in other words, if  $C_{k-1} > C_k$ , then  $E_{C_{k-1}} = k - 1$ ,  $E_{C_{k-1}-1} = k - 1$ , ...,  $E_{C_k+1} = k - 1$ . The ending frames of category  $C_{k-1}$  through  $C_k + 1$  are  $k - 1$ .
- If the length of a segment is less than a certain threshold value ( $\beta$ ), we ignore this segment since it is too short to carry any semantic content. In general, this value  $\beta$  is one second. In other words, we assume that the minimum length of a segment is one second.

### 3 New Proposed Techniques

We propose new techniques to capture the *amount* and the *location* of motions occurring in a segment, to cluster those segmented pieces, and to model and detect normal events are discussed in this section.

#### 3.1 Motion Feature Extraction

We describe how to extract and represent motions from each segment decomposed from a video sequence as discussed in the previous section. We developed a technique for automatic measurement of the overall motion in not only two consecutive frames but also an entire shot which is a collection of frames in our previous works [24, 35]. We extend this technique to extract the motion from a segment, and represent it in a comparable form in this section. We compute *Total*

*Motion Matrix (TMM)* which is considered as the overall motion of a segment, and represented as a *two dimensional matrix*. For comparison purpose among segments with different lengths (in terms of number of frames), we also compute an *Average Motion Matrix (AMM)*, and its corresponding *Total Motion (TM)* and *Average Motion (AM)*.

The *TMM*, *AMM*, *TM* and *AM* for a segment with  $n$  frames are computed using the following algorithm (Step 1 through 5). We assume that the frame size is  $c \times r$  pixels.

- **Step.1:** The color space of each frame is quantized (i.e., from 256 to 64 or 32 colors) to reduce unwanted noises (false detection of motion which is not actually motion but detected as motion).
- **Step.2:** An empty two dimensional matrix *TMM* (its size  $(c \times r)$  is same as that of frame) for a segment  $S$  is created as follows. All its items are initialized with zeros.

$$TMM_S = \begin{pmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1c} \\ t_{21} & t_{22} & t_{23} & \dots & t_{2c} \\ \dots & \dots & \dots & \dots & \dots \\ t_{r1} & t_{r2} & t_{r3} & \dots & t_{rc} \end{pmatrix} \quad (2)$$

And *AMM<sub>S</sub>* which is a matrix whose items are averages computed as follows.

$$AMM_S = \begin{pmatrix} \frac{t_{11}}{n} & \frac{t_{12}}{n} & \frac{t_{13}}{n} & \dots & \frac{t_{1c}}{n} \\ \frac{t_{21}}{n} & \frac{t_{22}}{n} & \frac{t_{23}}{n} & \dots & \frac{t_{2c}}{n} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{t_{r1}}{n} & \frac{t_{r2}}{n} & \frac{t_{r3}}{n} & \dots & \frac{t_{rc}}{n} \end{pmatrix} \quad (3)$$

- **Step.3:** Compare all the corresponding quantized pixels in the same position of each and background frames. If they have different colors, increase the matrix value ( $t_{ij}$ ) in the corresponding position by one (this value may be larger according to the other conditions). Otherwise, it remains the same.
- **Step.4:** Step.3 is repeated until all  $n$  frames in a shot are compared with a background frame.
- **Step.5:** Using the above *TMM<sub>S</sub>* and *AMM<sub>S</sub>*, we compute a motion feature, *TM<sub>S</sub>*, *AM<sub>S</sub>* as follows.

$$TM_S = \sum_{i=1}^r \sum_{j=1}^c t_{ij}, \quad AM_S = \sum_{i=1}^r \sum_{j=1}^c \frac{t_{ij}}{n} \quad (4)$$

As seen in these formulae, *TM* is the sum of all items in *TMM* and we consider this as total motion in a segment. In other words, *TM* can indicate an amount of motion in a segment. However, *TM* is dependent not only on the amount of motions but also on the length of a segment. A *TM* of long segment with little motions can be equivalent to a *TM* of short segment with a lot of motions. To distinguish these, we simply use *AM* which is an average of *TM*.

### 3.2 Location of Motion

Comparing segments only by the amount of motion (i.e.,  $AM$ ) would not give very accurate results because it ignores the locality such that where the motions occur. We introduce a technique to capture locality information without using partitioning, which is described as follows. In the proposed technique, the locality information of  $AMM$  can be captured by two one dimensional matrices which are the summation of column values and the summation of row values in  $AMM$ . These two arrays are called as *Summation of Column (SC)* and *Summation of Row (SR)* to indicate their actual meanings. The following equations show how to compute  $SC_A$  and  $SR_A$  from  $AMM_A$ .

$$SC_A = \left( \sum_{i=1}^r a_{i1} \quad \sum_{i=1}^r a_{i2} \quad \dots \quad \sum_{i=1}^r a_{ic} \right)$$

$$SR_A = \left( \sum_{j=1}^c a_{1j} \quad \sum_{j=1}^c a_{2j} \quad \dots \quad \sum_{j=1}^c a_{rj} \right)$$

To visualize the computed  $TMM$  (or  $AMM$ ), we can convert this  $TMM$  (or  $AMM$ ) to an image which is called *Total Motion Matrix Image (TMMI)* for  $TMM$  (*Average Motion Matrix Image (AMMI)* for  $AMM$ ). Let us convert a  $TMM$  with the maximum value,  $m$  into a 256 gray scale image as an example. We can convert an  $AMM$  using the same way. If  $m$  is greater than 256,  $m$  and other values are scaled down to fit into 256, otherwise, they are scaled up. But the value zero remains unchanged. An empty image with same size of  $TMM$  is created as  $TMMI$ , and the corresponding value of  $TMM$  is assigned as a pixel value. For example, assign white pixel for the matrix value zero which means no motion, and black pixels for the matrix value 256 which means maximum motion in a given shot. Each pixel value for a  $TMMI$  can be computed as follows after it is scaled up or down if we assume that  $TMMI$  is a 256 gray scale image.

$$\text{Each Pixel Value} = 256 - \text{Corresponding Matrix Value}$$

Figure 4 shows some visualization examples of  $AMMI$ ,  $SC$  and  $SR$  such that how these  $SC$  and  $SR$  can capture where the motions occur. Two  $SR$ s in Figure 4 (a) are same, which means that the vertical locations of two motions are same. Similarly, Figure 4 (b) shows that the horizontal locations of two motions are same by  $SC$ s. Figure 4 (c) is showing the combination of two, the horizontal and vertical location changes.

### 3.3 Clustering of Segments

In our clustering, we employ a multi-level hierarchical clustering approach to group segments in terms of category, and motion of segments. The algorithm is implemented in a top-down fashion, where the feature, category is utilized at the top level, in other words, we group segments into  $k_1$  clusters according



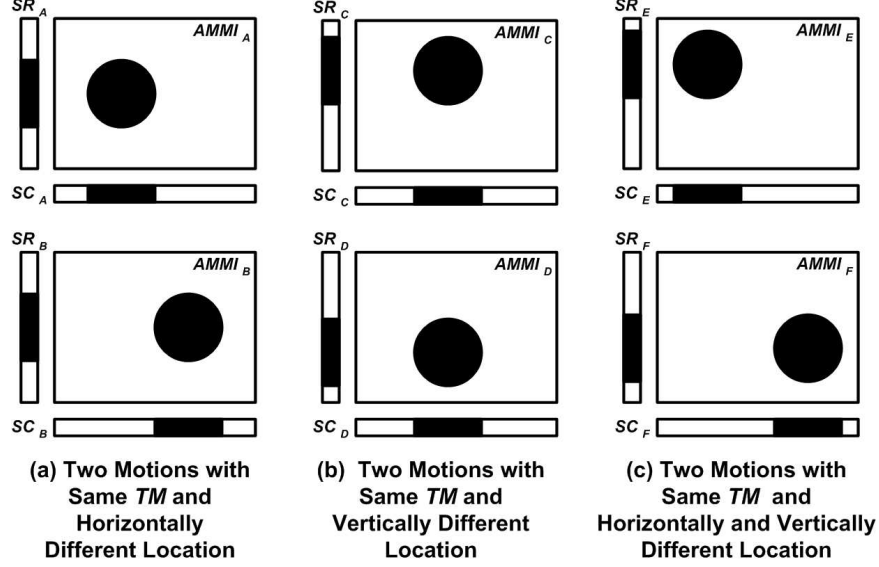


Fig. 4. Comparisons of Locations of Motions

to the categories. For convenience, we call this feature as *Top Feature*. Each cluster is clustered again into  $k_2$  groups based on the motion (AM) extracted in the previous section accordingly, which are called as *Bottom Feature*. We will consider more features (i.e.,  $SC$  and  $SR$ ) for the clustering in the future.

For this multi-level clustering, we adopted K-Means algorithm and cluster validity method studied by Ngo et. al. [36] since the algorithm is the most frequently used clustering algorithm due to its simplicity and efficiency. It is employed to cluster segments at each level of hierarchy independently. The K-Mean algorithm is implemented as follows.

– **Step.1:** The initial centroids are selected in the following way:

1. Given  $v$   $d$ -dimensional feature vectors, divide the  $d$  dimensions to  $\rho = \frac{d}{k}$ . These subspaces are indexed by  $[1, 2, 3, \dots, \rho]$ ,  $[\rho + 1, \rho + 2, \dots, 2\rho]$ , ...,  $[(k-1)\rho + 1, (k-1)\rho + 2, (k-1)\rho + 3, \dots, k\rho]$ .

2. In each subspace  $j$  of  $[(j-1)\rho + 1, \dots, j\rho]$ , associate a value  $f_i^j$  for each feature vector  $\mathcal{F}_i$  by  $f_i^j = \sum_{d=(j-1)\rho}^{j\rho} \mathcal{F}_i(d)$

3. Choose the initial cluster centroids  $\mu_1, \mu_2, \dots, \mu_k$ , by  $\mu_j = \arg_{\mathcal{F}_i} \max_{1 < i < v} f_i^j$

– **Step.2:** Classify each feature  $\mathcal{F}$  to the cluster  $p_s$  with the smallest distance.  $\bar{p}_s = \arg_{1 \leq j \leq k} \min D(\mathcal{F}, \mu_j)$ . This  $D$  is a function to measure the distance between two feature vectors and defined as

$$D(\mathcal{F}, \mathcal{F}') = \frac{1}{Z(\mathcal{F}, \mathcal{F}')} \left( \sum_{i=1}^v |\mathcal{F}(i) - \mathcal{F}'(i)|^k \right)^{\frac{1}{k}}$$

where  $\mathcal{Z}(\mathcal{F}, \mathcal{F}') = \sum_{i=1}^v \mathcal{F}(i) + \sum_{i=1}^v \mathcal{F}'(i)$  which is a normalizing function. In this function,  $k = 1$  for  $L_1$  norm and  $k = 2$  for  $L_2$  norm. The  $L_1$  and  $L_2$  norms are two of the most frequently used distance metrics for comparing two feature vectors. In practice, however,  $L_1$  norm performs better than  $L_2$  norm since it is more robust to outliers [37]. Furthermore,  $L_1$  norm is more computationally efficient and robust. We use  $L_1$  norm for our experiments.

- **Step.3:** Based on the classification, update cluster centroids as

$$\mu_j = \frac{1}{v_j} \sum_{i=1}^{v_j} \mathcal{F}_i^{(j)}$$

where  $v_j$  is the number of shots in cluster  $j$ , and  $\mathcal{F}_i^{(j)}$  is the  $i^{th}$  feature vector in cluster  $j$ .

- **Step.4:** If any cluster centroid changes the value in Step.3, go to Step.2, otherwise stop.

The above K-Mean algorithm can be used when the number of clusters  $k$  is explicitly specified. To find optimal number ( $k$ ) clusters, we have employed the cluster validity analysis [38]. The idea is to find clusters that minimize intra-cluster distance while maximize inter-cluster distance. The cluster separation measure  $\varphi(k)$  is defined as

$$\varphi(k) = \frac{1}{k} \sum_{i=1}^k \max_{1 \leq v \leq k} \frac{\eta_i + \eta_j}{\xi_{ij}}$$

where  $\eta_j = \frac{1}{v_j} \sum_{i=1}^{v_j} D(\mathcal{F}_i^{(j)}, \mu_i)$ ,  $\xi_{ij} = D(\mu_i, \mu_j)$ .  $\xi_{ij}$  is the inter-cluster distance of cluster  $i$  and  $j$ , while  $\eta_j$  is the intra-cluster distance of cluster  $j$ . The optimal number of cluster  $k'$  is selected as  $k' = \min_{1 \leq k \leq q} \varphi(k)$ . In other words, the K-Mean algorithm is tested for  $k = 1, 2, \dots, q$ , and the one which gives the lowest value of  $\varphi(k)$  is chosen.

In our multi-level clustering structure, a centroid at the top level represents the category of segments in a cluster, and a centroid at the bottom level represents the general motion characteristics of a sub-cluster.

### 3.4 Modelling and Detecting of Normal Events

As mentioned in the section 1, to find the abnormal event, we cluster and model the normal events which occur everyday and are easy to obtain. More precisely, the segments with normal events are clustered and modelled using the extracted features about the amount and location of motions. The algorithm can be summarized as follows.

- The existing segments are clustered into  $k$  number of clusters using the technique discussed in the section 3.3.

- We compute a *Closeness to Neighbors* ( $\Delta$ ) for a given segment ( $s_g$ ) as follows,

$$\Delta = \frac{\sum_{i=1}^m D(s_g, s_i)}{m} \quad (5)$$

where  $D(s_g, s_i)$  is a distance between  $s_g$  and  $s_i$ . This  $\Delta$  is an average value of the distances between a number ( $m$ ) of closest segments to a given segment  $s_g$  in its cluster. We can use the distance function defined in the Step.2 of the previous subsection (3.3) for the computation of  $D(s_g, s_i)$ . This is possible since a segment can be represented as a feature vector by the features used for the clustering in the above.

- Compute  $\Delta$  of all existing segments, and an average value of  $\Delta$ s of the segments in each cluster  $k_1$ , which is represented as  $\bar{\Delta}^{k_1}$ .
- If a new segment ( $S$ ) comes in, then decide which cluster it goes to, its  $\Delta_S$ . If it goes to the cluster  $k_1$ , we can compute whether it is normal or not as follows.

$$\text{If } \bar{\Delta}^{k_1} \geq \Delta_S, \text{ then } S = \text{Normal}, \quad \text{Otherwise } S = \text{Abnormal} \quad (6)$$

If  $S$  is abnormal. then its degree of abnormality ( $\Psi$ ) can be computed as follows, which is greater than zero.

$$\Psi = \left| \frac{\bar{\Delta}^{k_1} - \Delta_S}{\bar{\Delta}^{k_1}} \right| \quad (7)$$

In addition to determining normal or abnormal, we find to what extent a segment with event or events are distant to the existing segments with normal events. The idea is that if a segment is close enough to the segments with normal events, there are more possibilities in which a given segment can be normal. As seen in the equation (6), if the value of  $\Delta$  for a new segment is less than or equal to the average of the existing segments in the corresponding cluster, then the new segment can be normal since it is very close to the existing segments as we discussed in the beginning of this subsection. Otherwise, we compute a degree of abnormality using the differences between them.

## 4 Experimental Results

Our experiments in this paper were designed to assess the following performance issues.

- How does the proposed segmentation algorithm work to group incoming frames?
- How do  $TM(AM)$ ,  $SC$  and  $SR$  capture the amount and the location of motions in a segment?
- How do the proposed algorithms work for clustering and modelling of segments?

Our test video clips were originally digitized in AVI format at 30 frames/second. Their resolution is  $160 \times 120$  pixels. We used the rates of 5 and 2 frames/second as the incoming frame rates. Our test set has 111 minutes and 51 seconds of raw video taken from a hallway in a building which consists of total 17,635 frames.

#### 4.1 Performance of Video Segmentation

A simple segmentation example can be found in Figure 5 and Table 2. The fourth and fifth columns of the table show the length (number of frames) of each segment and its category. The next two columns show *Total Motion* and *Average Motion* for each segment computed using the equation (4). The proposed segmentation algorithm discussed in section 2 was applied to our test video sequence mentioned above. As results, four different hierarchical segments are partitioned in Figure 5. The most common content of this type of video is that the objects (i.e., people, vehicles, etc.) are appearing from and disappearing into with various directions. The segment #33 (Category #2) represents this type of content in which a person is appearing and disappearing in this case.

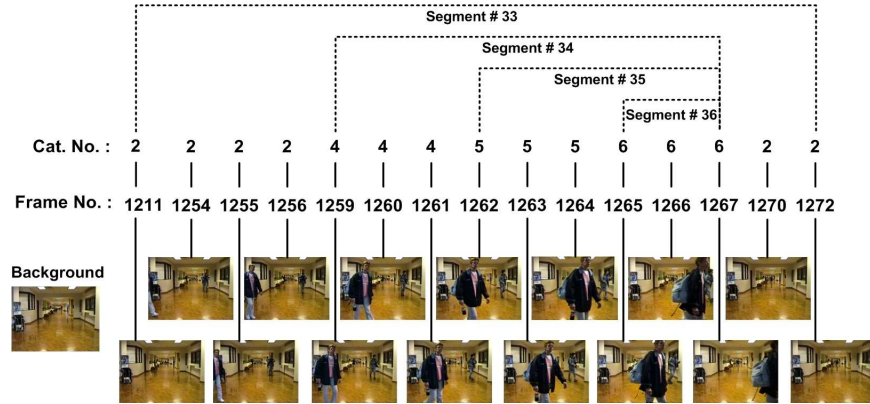


Fig. 5. Segmentation example

Table 3 shows the overall segmentation results for our test set. The second and the third columns of the table represent the number of frames per each category, and the accumulated number of frames up to the corresponding category. For example, the number, 6,440 in the row of cat. #3 indicates the sum of the number of frames (the second column) from the category #6 to the category #3. The fourth and fifth columns of the table indicate the number of segments and their average length for each category. The average motion ( $AM$ ) of segments in each category is shown in the sixth column. We can see the average value ( $\bar{\Delta}^k$ ) of  $\Delta$ s of the segments in each category in the seventh column. As seen in this table, the higher category segments can be hierarchical summaries for the lower category segments.

#### 4.2 Performance of Capturing Amount and Location of Motions and Clustering

Figure 6 shows some examples of  $TM$ ,  $AM$ ,  $SC$  and  $SR$  for a number of segments in various categories. These features are represented as the images (i.e.,

Segment No.	Starting Frame No.	Ending Frame No.	Segment Length (Frames)	Cat. ( $C_k$ )	Total Motion (TM)	Avg. Motion (AM)
33	1211	1272	62	2	303	4.9
34	1259	1267	9	4	105	11.7
35	1262	1267	6	5	75	12.5
36	1265	1267	3	6	40	13.3

Table 2. Segmentation Result for Figure 5

Category	No. of Frames	No. of Frames Accumulated	No. of Segments	Avg. No. of Frames / Segment	Average of AMs	Value of $\Delta^k$
Cat. # 1	999	17,635	36	490.0	3.7	0.45
Cat. # 2	10,196	16,636	237	70.2	4.7	0.09
Cat. # 3	3,928	6,440	224	28.8	6.0	0.07
Cat. # 4	1,904	2,512	82	30.6	7.8	0.18
Cat. # 5	577	608	46	13.2	8.9	0.65
Cat. # 6	31	31	8	3.9	10.0	1.95

Table 3. Overall Segmentation Results for Test Set

$TMMI$  and  $AMMI$  as discussed before). As seen in this figure, the amount and the location of motions are well-presented by these features. We will investigate the uniqueness of these  $SC$  and  $SR$ , and how to compare these in the future. Figure 6 shows a very simple example of clustering segments. As seen in this figure, the segments are clustered by category, and further partitioned using a motion feature,  $AM$ . The different categories have the different sizes and/or numbers of object(s), in other words, the segments in the higher categories have relatively larger or more objects. On the other hand, the average motions, represented by  $AM$  can distinguish the amount(degree) of motions in different segments. Also, we will consider  $SC$  and  $SR$  for more detail clustering in the future.

### 4.3 Performance of Computing Abnormality

A very simple example of computing abnormalities can be seen in Table 4. We consider that these segments in the table are segmented from new incoming stream. The values of  $\Delta_S$  for the segments (#130, #131, #133 and #134) are smaller than the values of  $\Delta^k$  for their corresponding categories. Therefore, the abnormality ( $\Psi$ ) for those segments can be represented as *normal* as we discussed in the section 3.4. However, since the  $\Delta_{132}$  (=0.15) is larger than  $\Delta^3$  (=0.07), the segment #132 is considered as an abnormal at the moment, and the actual value of abnormality ( $\Psi$ ) can be computed as 1.14 using the equation (7) as shown





























































Category	Segments	AM	TMMI	AMMI	SR	SC
1		3.8				
		3.9				
2		4.4				
		4.9				
3		6.1				
		5.8				
4		8.9				
		7.3				
5		9.9				
		9.5				
6		11.0				
		11.5				

Fig. 6. Sample Clustering Results

in the table (the last column). For better illustration, Figure 7 shows that a number of frames in the segment #132 and a typical segment in the category 3 to which the the segment #132 belongs. The new incoming segment #132 is different from a typical segment in the category 3 in terms of, for example, the size and the number of object(s). This difference is captured by our algorithm for computing the abnormality. Eventually, this segment #132 becomes a normal segment if this type of segment is occurring frequently (because there is nothing wrong actually). If more number of segments similar to this segment comes, then this kind of segment will be detected as normal at a certain point.

Segment No.	Number of Frames	Cat. ( $C_k$ )	Avg. Motion (AM)	Value of $\Delta_s$	Value of $\bar{\Delta}^k$	Abnormality $\Psi$
130	23	1	3.5	0.29	0.45	Normal
131	3	2	3.7	0.07	0.09	Normal
132	4	3	4.5	0.15	0.07	1.14
133	68	1	2.7	0.30	0.45	Normal
134	3	2	3.9	0.02	0.09	Normal

Table 4. Example of Computing Abnormalities

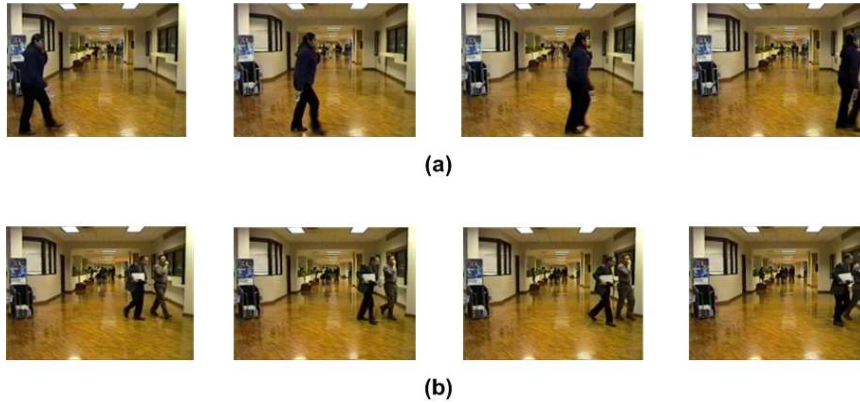


Fig. 7. (a) Four Frames in Segment #132. (b) Four Frames in a Typical Segment in Category 3.

## 5 Concluding Remarks

The examples of knowledge and patterns that we can discover and detect from a surveillance video sequence are object identification, object movement pattern recognition, spatio-temporal relations of objects, modelling and detection of normal and abnormal (interesting) events, and event pattern recognition. In this paper, we extend our previous work [1] about the general framework to perform the fundamental tasks for video data mining which are temporal segmentation of video sequences, and feature (motion in our case) extraction. The extension includes how to capture the *location* of motions occurring in a segment, how to cluster those segmented pieces, and how to find whether a segment has normal or abnormal events. Our experimental results are showing that the proposed techniques are performing the desired tasks. In the future study, we will consider the other features (objects, colors) extracted from segments for more sophisticated clustering and indexing.

## References

1. JungHwan Oh and Babitha Bandi. Multimedia data mining framework for raw video sequences. In *Proc. of ACM Third International Workshop on Multimedia Data Mining (MDM/KDD2002)*, Edmonton, Alberta, Canada, July 2002.
2. P. Stolorz, H. Nakamura, E. Mesrobian, R. Muntz, E. Shek, J. Santos, J Yi, K Ng, S. Chien, C. Mechoso, and J. Farrara. Fast spatio-temporal data mining of large geophysical datasets. In *Proc. of Int'l Conf. on KDD*, pages 300–305, 1995.
3. U. Fayyad, S. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. *Advances in Knowledge Discovery with Data Mining*, pages 471–493, 1996.
4. Z.-N Li, O.R. Zaiane, and Z. Tauber. Illumination invariance and object model in content-based image and video retrieval. *Journal of Visual Communication and Image Representation*, 1998.
5. O.R. Zaiane, J. Han, Z.-N. Li, S. Chee, and J. Chiang. Multimediaminer: A system prototype for multimedia data mining. In *Proc. of the 1998 ACM SIGMOD Conf.*, pages 581–583, 1998.
6. O.R. Zaiane, J. Han, Z.-N. Li, and J. Hou. Mining multimedia data. In *Proc. of the CASCON'98: Meeting of Minds*, pages 27–32, 1998.
7. K. Koperski, J. Adikary, and J. Han. Spatial data mining: Progress and challenges. In *Proc. of SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'96)*, pages 27–32, Montreal, Canada, 1996.
8. K. Koperski and J. Han. Mining knowledge in geographical data. In *Communication of ACM*, 1998.
9. D. Wijesekera and D. Barbara. Mining cinematic knowledge: Work in progress. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2000)*, pages 98–103, Boston, MA, August 2000.
10. K. Shearer, C. Dorai, and S. Venkatesh. Incorporating domain knowledge with video and voice data analysis in news broadcasts. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2000)*, pages 46–53, Boston, MA, August 2000.



11. V. Kulesh, V. Petrushin, and I. Sethi. The perseus project: Creating personalized multimedia news portal. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2001)*, pages 31–37, San Francisco, CA, August 2001.
12. Y. Chen, W. Gao, Z. Wang, J. Miao, and D. Jiang. Mining audio/visual database for speech driven face animation. In *Proc. of International Conference on Systems, Man and Cybernetics*, pages 2638–2643, 2001.
13. P.K. Singh and A.K. Majumdar. Semantic content-based retrieval in a video database. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2001)*, pages 50–57, San Francisco, CA, August 2001.
14. S. Chen, M. Shyu, C. Zhang, and J. Strickrott. Multimedia data mining for traffic video sequences. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2001)*, pages 78–86, San Francisco, CA, August 2001.
15. R. Cucchiara, M. Piccardi, and P. Mello. Image analysis and rule-based reasoning for a traffic monitoring system. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):119–130, June 2000.
16. D. Dailey, F. Cathey, and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, June 2000.
17. J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tacking for easyliving. In *Proc. of 3rd IEEE International Workshop on Visual Surveillance*, pages 3–10, 2000.
18. S. Shafer, J. Krumm, B. Meyers, B. Brumitt, M. Czerwinski, and D. Robbins. The new easyliving project at microsoft research. In *Proc. of DARPA/NIST Workshop on Smart Spaces*, pages 127–130, 1998.
19. M. Coen. The future of human-computer interaction on how I learned to stop worrying and love my intelligent room. *IEEE Intelligent Systems*, 14(2):8–10, March 1999.
20. I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: From the laboratory to the commercial world. *Proceedings of The IEEE*, 89(10):1478–1497, Oct. 2001.
21. S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Traffic monitoring and accident detection at intersections. In *IEEE Intenational Conference on Intelligent Tansportation Systems*, pages 703–708, Tokyo, Japan, 1999.
22. T. Huang, D. Koller, J. Malik, and G. Ogasawara. Automatic symbolic traffic scene analysis using belief networks. In *Proc. of AAAI, 12th National Conference on Artificial Intelligence (AAAI'94)*, pages 966–972, Seattle, WA, 1994.
23. D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *Proc. of European Conference on Computer Vision*, pages 189–196, Stockholm, Sweden, 1994.
24. JungHwan Oh and Praveen Sankuratri. Automatic distinction of camera and objects motions in video sequences. In *To appear in Proc. of IEEE International Conference on Multimedia and Expo (ICME 2002)*, Lausanne, Switzerland, Aug. 2002.
25. G.L. Foresti and F. Roli. Learning and classification of suspicious events for advanced visual-based surveillance. In *Multimedia Video-based Surveillance Systems: Requirements, Issues and Solutions*, pages 84–93, Norwell, MA: Kluwer, 2000.
26. C. Sacchi and C.S. Regazzoni. A distributed surveillance system for detection of abandoned objects in unmanned railway environments. *IEEE Transaction on Veh. Technol.*, 49:1355–1367, Sept. 2000.

27. J.A. Freer, B.J. Beggs, H.L. Fernandez-Canque, F. Chevrier, and A. Goryashko. Automatic intruder detection incorporating intelligent scene monitoring with video surveillance. In *Proc. Eur. Conf. Security and Detection, ECOS'97*, pages 109–113, 1997.
28. G. Foresti and B. Pani. Monitoring motorway infrastructures for detection of dangerous events. In *Proc. of IEEE Int. Conf. Image Analysis and Processing*, pages 1144–1147, 1999.
29. L. Zhao, W. Qi, Y. Wang, S. Yang, and H. Zhang. Video shot grouping using best-first model merging. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 262–269, San Jose, CA, Jan. 2001.
30. S. Han and I. Kweon. Shot detection combining bayesian and structural information. In *Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001*, pages 509–516, San Jose, CA, Jan. 2001.
31. JungHwan Oh and Kien A. Hua. An efficient and cost-effective technique for browsing and indexing large video databases. In *Proc. of 2000 ACM SIGMOD Intl. Conf. on Management of Data*, pages 415–426, Dallas, TX, May 2000.
32. JungHwan Oh, Kien A. Hua, and Ning Liang. A content-based scene change detection and classification technique using background tracking. In *SPIE Conf. on Multimedia Computing and Networking 2000*, pages 254–265, San Jose, CA, Jan. 2000.
33. Kien A. Hua and JungHwan Oh. Detecting video shot boundaries up to 16 times faster. In *The 8th ACM International Multimedia Conference (ACM Multimedia 2000)*, pages 385–387, LA, CA, Oct. 2000.
34. I. Haritaoglu, D. Harwood, and L.S. Davis. W4 - who, where, when, what: A real-time system for detecting and tracking people. In *IEEE Third International Conference on Face and Gesture Recognition*, pages 222–227, Nara, Japan, 1998.
35. JungHwan Oh and Tummala Chowdary. An efficient technique for measuring of various motions in video sequences. In *Proc. of The 2002 International Conference on Imaging Science, System, and technology (CISST'02)*, Las Vegas, NV, June 2002.
36. C.W. Ngo, T.C. Pong, and H.J. Zhang. On clustering and retrieval of video shots. In *Proc. of ACM Multimedia 2001*, pages 51–60, Ottawa, Canada, Oct. 2001.
37. P.J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987.
38. A. K. Jain. *Algorithm for Clustering Data*. Prentice Hall, 1988.