Real Time Video Data Mining for Surveillance Video Streams

JungHwan Oh, JeongKyu Lee, and Sanjaykumar Kote

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

1 Introduction

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 [2] and traffic monitoring systems [3] 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 [4], 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.

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.
– To find the abnormality, our approach uses the normal events which are occurring 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 have been proposed in our previous work [1, 5]. 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 [6–10], 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 [4]. 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 color histogram matching technique for illustration purpose.

– Step.1: A background frame is extracted from a given sequence as preprocessing, and its color histogram is computed. In other words, this frame is represented as a *bin* with a certain number (bin size) of quantized colors from the original. As a result, a background frame (F^B) is represented as

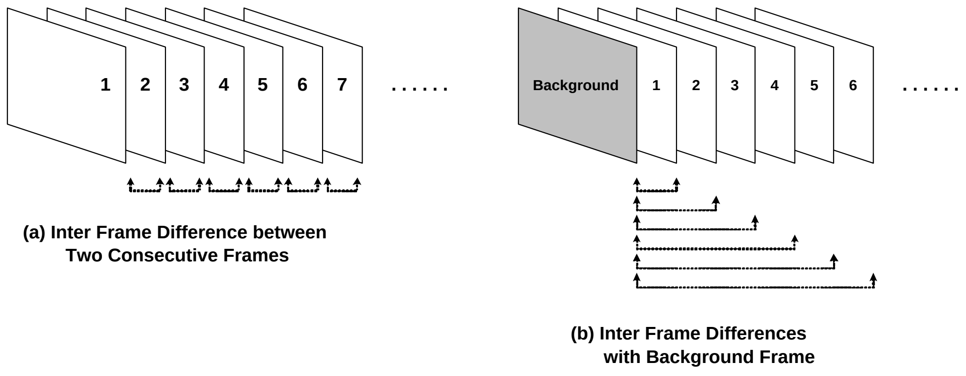

1
2
3
4
5
6
7
......
(a) Inter Frame Difference between Two Consecutive Frames
Background
1
2
3
4
5
6
......
(b) Inter Frame Differences with Background Frame

Fig. 1. Frame Comparison Strategies

follows using a *bin* with the size n. Note that P_T is representing the total number of pixels in a background or any other frame.

$$F^B = bin^B = (v_1^B, v_2^B, v_3^B, ..., v_n^B), \qquad where \sum_{i=1}^{n} v_i^B = P_T. \tag{1}$$

– Step.2: Each frame (F^k) arriving to the system is represented as follows in the same way, as the background is represented in the previous step.

$$F^k = bin^k = (v_1^k, v_2^k, v_3^k, ..., v_n^k), \qquad where \sum_{i=1}^{n} v_i^k = P_T. \tag{2}$$

– Step.3: Compute the difference (D^k) between the background (F^B) and each frame (F^k) as follows. Note that the value of D^k is always between zero and one.

$$D^k = \frac{F^B - F^k}{P_T} = \frac{bin^B - bin^k}{P_T} = \frac{\sum_{i=1}^{n}(v_i^B - v_i^k)}{P_T} \tag{3}$$

– 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 for illustration purpose, but this value can be changed properly according to the contents of video.
– Step.5: For real time on-line processing, a temporary table is maintained. To do this, and 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 2. In our hierarchical segmentation, therefore, finding segment boundaries means finding category boundaries in which we find a starting frame (S_i) and an ending frame (E_i) for each category i.

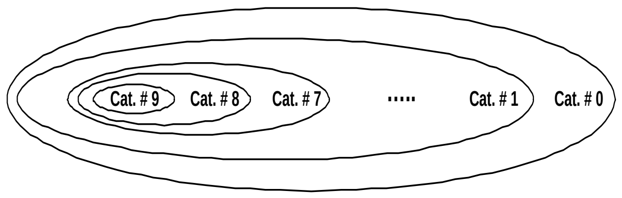


Fig. 2. Relationships (Containments) among Categories

3 New Proposed Techniques

We propose new techniques about 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 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 [5, 11]. 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 is 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.

$$TMM_S = \begin{pmatrix} t_{11} & t_{12} & t_{13} & ... & t_{1c} \\ t_{21} & t_{22} & t_{23} & ... & t_{2c} \\ ... & ... & ... & ... & ... \\ t_{r1} & t_{r2} & t_{r3} & ... & t_{rc} \end{pmatrix} \qquad (4)$$

And AMM_S which is a matrix whose items are averages computed as follows.

$$AMM_S = \begin{pmatrix} \frac{t_{11}}{n} & \frac{t_{12}}{n} & \frac{t_{13}}{n} & \dots & \frac{t_{1c}}{n} \\ \frac{t_{21}}{n} & \frac{t_{22}}{n} & \frac{t_{23}}{n} & \dots & \frac{t_{2c}}{n} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{t_{r1}}{n} & \frac{t_{r2}}{n} & \frac{t_{r3}}{n} & \dots & \frac{t_{rc}}{n} \end{pmatrix} \tag{5}$$

- **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 consecutive pairs of frames are compared.
- **Step.5**: Using the above TMM_S and AMM_S, we compute a motion feature, TM_S, AM_S as follows.

$$TM_S = \sum_{i=0}^{r}\sum_{j=0}^{c} t_{ij}, \qquad AM_S = \sum_{i=0}^{r}\sum_{j=0}^{c} \frac{t_{ij}}{n} \tag{6}$$

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 on not only the amount of motions but also 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, simply we 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. $SC_A = (\sum_{i=0}^{r} a_{i1} \; \sum_{i=0}^{r} a_{i2} \; \dots \; \sum_{i=0}^{r} a_{ic})$, $\quad SR_A = (\sum_{j=0}^{c} a_{1j} \; \sum_{j=0}^{c} a_{2j} \; \dots \; \sum_{j=0}^{c} a_{rj})$.

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. $Each\ Pixel\ Value\ =\ 256\ -\ Corresponding\ Matrix\ Value.$

Figure 3 shows some visualization examples of $AMMI$, SC and SR such that how these SC and SR can capture where the motions occur. Two SRs in Figure 3 (a) are same, which means that the vertical locations of two motions are same. Similarly, Figure 3 (b) shows that the horizontal locations of two motions are same by SCs. Figure 3 (c) is showing the combination of two, the horizontal and vertical location changes.

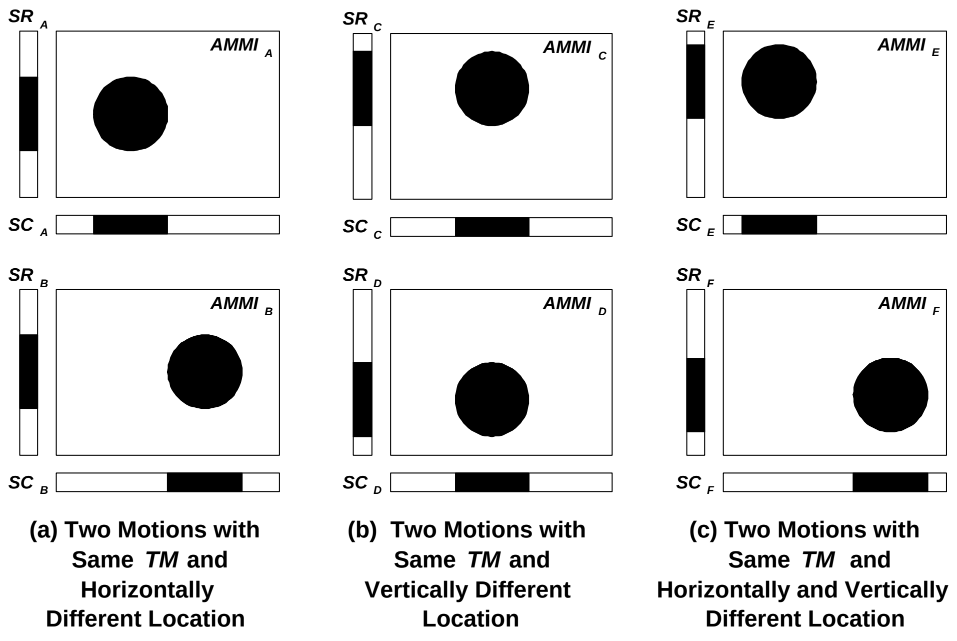

SR A
AMMI A
SC A
SR B
AMMI B
SC B
SR C
AMMI C
SC C
SR D
AMMI D
SC D
SR E
AMMI E
SC E
SR F
AMMI F
SC F
(a) Two Motions with Same TM and Horizontally Different Location
(b) Two Motions with Same TM and Vertically Different Location
(c) Two Motions with Same TM and Horizontally and Vertically Different Location

Fig. 3. Comparisons of Locations of Motions

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 to the categories. For convenience, we called 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-Mean algorithm and cluster validity method studied by Ngo et. al. [12] 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 $[1, 2, 3, ..., \rho]$, $[\rho, \rho+1, \rho+2, ..., 2\rho]$, ..., $[(k-1)\rho+1, (k-1)\rho+2, (k-1)\rho+3, ..., k\rho]$.
  2. In each subspace j of $[(j-1)\rho+1, ..., j\rho]$ associate a value f_i^j for each feature vector $\mathcal{F}_i$ by $f_i^j = \sum_{m=(j-1)}^{j\rho} \rho\mathcal{F}_i(d)$
  3. Choose the initial cluster centroids $\mu_1, \mu_2, ..., \mu_k$, by $\mu_j = \arg_{\mathcal{F}_i} \max_{1<i<v} f_i^j$
- **Step.2**: Classify each feature F to the cluster p_s with the smallest distance. $p_s = \arg_{1\leq j\leq k} \min D(\mathcal{F}, \mu_j)$. This D is a function to measure the distance between two feature vectors and defined as $D(\mathcal{F}, \mathcal{F}') = \frac{1}{\mathcal{Z}(\mathcal{F},\mathcal{F}')}(\sum_{i=1}^{v} |\mathcal{F}(i)-\mathcal{F}'(i)|^k)^{\frac{1}{k}}$, 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 [13]. 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 by 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 [14]. 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 $\eta_j = \frac{1}{v_j}\sum_{i=1}^{v_j} D(\mathcal{F}_i^j, \mu_i)$, $\xi_{ij} = D(\mu_i, \mu_j)$. ξ_{ij} is the inter-cluster distance of cluster i and j, while η_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 $k = 1, 2, ..., q$, 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 Modeling 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 modeled 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* ($\varDelta$) for a given segment (s_g) as follows,

$$\varDelta = \frac{\sum_{i=1}^{m} D(s_g, s_i)}{m} \tag{7}$$

  where $D(s_g, s_i)$ is a distance between s_g and s_i. This $\varDelta$ 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 $\varDelta$ of all existing segments, and an average value of $\varDelta$s of the segments in each cluster $k\ _1$, which is represented as $\bar{\varDelta}^{k_1}$.
- If a new segment (S) comes in, then decide which cluster it goes to, its $\varDelta\ _S$. If it goes to the cluster $k\ _1$, we can compute whether it is normal or not as follows.

$$If \quad \varDelta^{k_1} \geq \varDelta_S, \quad then \quad S = Normal, \qquad Otherwise \quad S = Abnormal \tag{8}$$

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

$$\varPsi = |\frac{\bar{\varDelta}^{k_1} - \varDelta_S}{\bar{\varDelta}^{k_1}}| \tag{9}$$

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 (8), if the value of $\varDelta$ 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 since we have already examined the issue, "how does the proposed segmentation algorithm work to group incoming frames?" in our previous work [1].

– 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 modeling of segments?

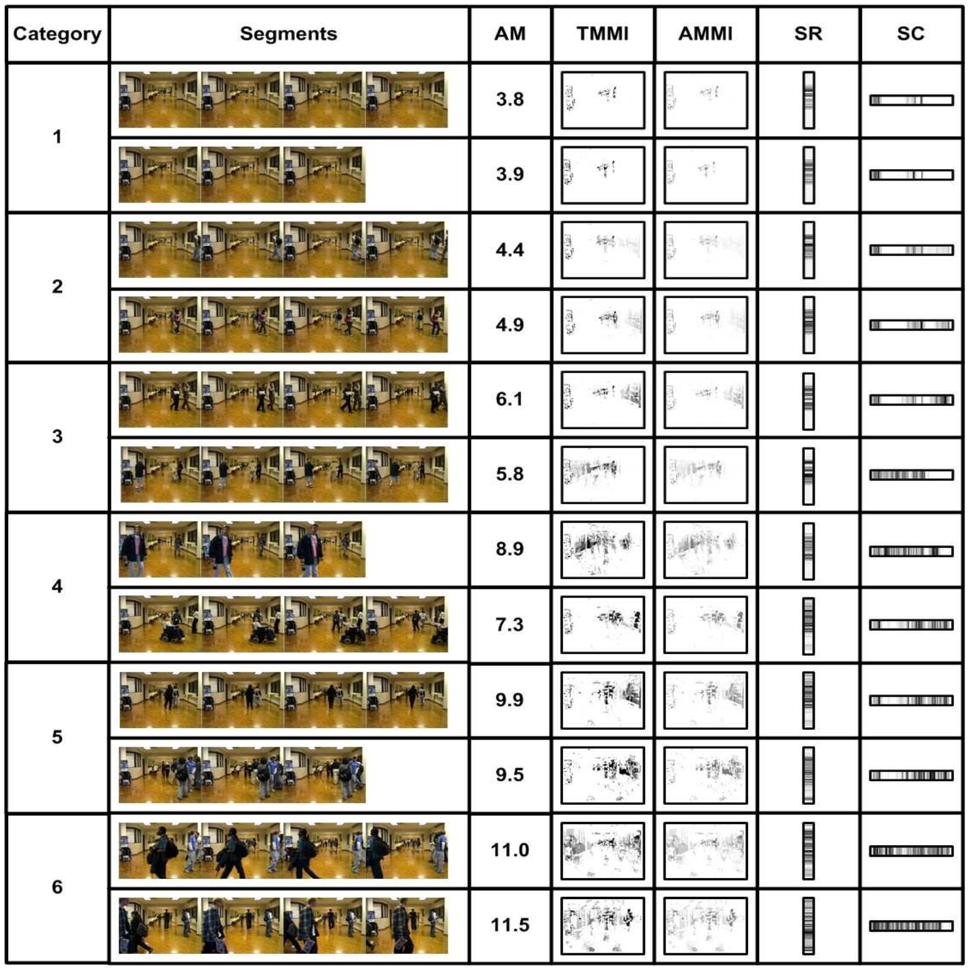
Our test video clips were originally digitized in AVI format at 30 frames/second. Their resolution is 160×120 pixels. We used the rates of 5 and 2 frames/second as the incoming frame rates. Our test set has 111 minutes and 51 seconds of raw video taken from a hallway in a building which consist of total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

4.1 Performance of Capturing Amount and Location of Motions and Clustering

Figure 4 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., $TMMI$ 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 4 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.

4.2 Performance of Computing Abnormality

A very simple example of computing abnormalities can be seen in Table 1. 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 Δ^k for their corresponding categories. Therefore, the abnormality (Ψ) 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 Δ^3 (0.07), the segment # 132 is considered as an abnormal at the moment, and the actual value of abnormality (Ψ) can be computed as 1.14 as shown in the table. For better illustration, Figure 5 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 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.



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. 4. Sample Clustering Results

Segment No.	Number of Frames	Cat. (C_k)	Avg. Motion (AM)	Value of $\triangle_s$	Value of $\triangle^k$	Abnormality
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 1. Example of Computing Abnormalities

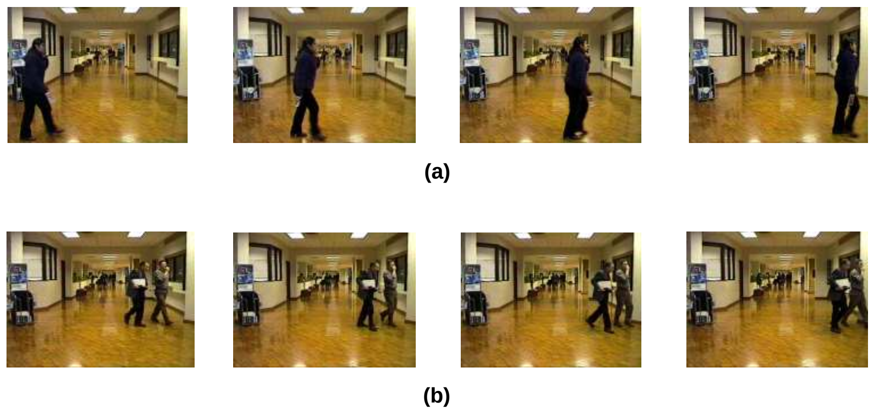


Fig. 5. (a) Four Frames in Segment # 132. (b) Four Frames in a Typical Segment in Category 3.

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, modeling 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 effectively and efficiently. In the future study, we will consider the other features (objects, colors) extracted from segments for more sophisticated clustering and indexing and deal with video files taken by moving camera.

References

1. Oh, J., Bandi, B.: Multimedia data mining framework for raw video sequences. In: Proc. of ACM Third International Workshop on Multimedia Data Mining (MDM/KDD2002), Edmonton, Alberta, Canada (2002)
2. Pavlidis, I., Morellas, V., Tsiamyrtzis, P., Harp, S.: Urban surveillance systems: From the laboratory to the commercial world. Proceedings of The IEEE **89** (2001) 1478–1497
3. Cucchiara, R., Piccardi, M., Mello, P.: Image analysis and rule-based reasoning for a traffic monitoring system. IEEE Transactions on Intelligent Transportation Systems **1** (2000) 119–130

4. Chen, S., Shyu, M., Zhang, C., Strickrott, J.: Multimedia data mining for traffic video sequences. In: Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2001), San Francisco, CA (2001) 78–86
5. Oh, J., Sankuratri, P.: 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 (2002)
6. Zhao, L., Qi, W., Wang, Y., Yang, S., Zhang, H.: Video shot grouping using best-first model merging. In: Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001, San Jose, CA (2001) 262–269
7. Han, S., Kweon, I.: Shot detection combining bayesian and structural information. In: Proc. of SPIE conf. on Storage and Retrieval for Media Databases 2001, San Jose, CA (2001) 509–516
8. Oh, J., Hua, K.A.: 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, Dallas, TX (2000) 415–426
9. Oh, J., Hua, K.A., Liang, N.: A content-based scene change detection and classification technique using background tracking. In: SPIE Conf. on Multimedia Computing and Networking 2000, San Jose, CA (2000) 254–265
10. Hua, K.A., Oh, J.: Detecting video shot boundaries up to 16 times faster. In: The 8th ACM International Multimedia Conference (ACM Multimedia 2000), LA, CA (2000) 385–387
11. Oh, J., Chowdary, T.: 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 (2002)
12. Ngo, C., Pong, T., Zhang, H.: On clustering and retrieval of video shots. In: Proc. of ACM Multimedia 2001, Ottawa, Canada (2001) 51–60
13. Rousseeuw, P., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley and Sons (1987)
14. Jain, A.K.: Algorithm for Clustering Data. Prentice Hall (1988)