MULTIMEDIA DATA MINING FRAMEWORK FOR RAW VIDEO SEQUENCES

JungHwan Oh, *Babitha Bandi*

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

ABSTRACT

In this paper, we propose a general framework for real time video data mining to be applied to the raw videos (traffic videos, surveillance videos, etc.). We investigate whether the existing techniques would be applicable to this type of videos. Then, we introduce new techniques which are essential to process them in real time. The first step of our frame work for mining raw video data is grouping input frames to a set of basic units which are relevant to the structure of the video. We call this unit as *segment*. This is one of the most important tasks since it is the step to construct the building blocks for video database and video data mining. The second step is characterizing each segment to cluster into similar groups, to discover unknown knowledge, and to detect interesting patterns. To do this, we extract some features (motion, object, colors, etc.) from each segment. In our framework, we focus on *motion* as a feature, and study how to compute and represent it for further processes. The third step of our framework is to cluster the decomposed segments into similar groups. In our clustering, we employ a multi-level hierarchical clustering approach to group segments using category and motion. Our preliminary experimental studies indicate that the proposed framework is promising.

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

1. INTRODUCTION

Data mining, which is defined 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. Even though relatively less research has been performed, very interesting and important studies and systems have been published in the areas of multimedia data mining.

Multimedia data mining has been performed for the different types of multimedia data; image, audio and video. An example of image data mining is *CONQUEST* [1] system that combines satellite data with geophysical data to discover patterns in global climate change. The *SKICAT* system [2] integrates techniques for image processing and data classification in order to identify 'sky objects' captured in a very large satellite picture set. The *MultiMediaMiner* [3] project have constructed many image understanding, indexing and mining techniques in digital media.

An example of video and audio data mining can be found in *Mining Cinematic Knowledge* project [4] 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 [5, 6] 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 audio-visual interaction has been presented [7] to learn the synchronous pattern between two channels, and apply it to speech driven lip motion facial animation system. The other example is a system [8] 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 [9, 10, 11] for traffic monitoring system. EasyLiving [12, 13] and HAL [14] 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 in the above, there have been some efforts about video data mining for movies, medical videos, and traffic videos. Generally, there are three types of videos; the produced, the raw, and the medical video. The examples of produced video are movies, news videos, dramas, etc. And, those of raw video are traffic videos, surveillance videos, etc. Ultra sound videos including echocardiogram can be an example of the medical videos. In fact, the developments of complex video surveillance systems [15] and traffic monitoring systems [10, 11, 16, 17, 18] have captured the interest of both research and industrial worlds recently due to the growing interest availability of cheap sensors and processors at reasonable costs, and the increasing safety and security concerns. As mentioned in the literature [9], 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.

These different types of videos need to be treated differently to achieve these missing parts due to their different characteristics. In this paper, we propose a general framework for video data mining to be applied to the *raw videos* in *real time*. We investigate whether the existing multimedia data mining techniques would be applicable to this type of videos. Then, we introduce new techniques which are essential to process them in real time. Figure 1 is showing the proposed framework which can be summarized as follow.

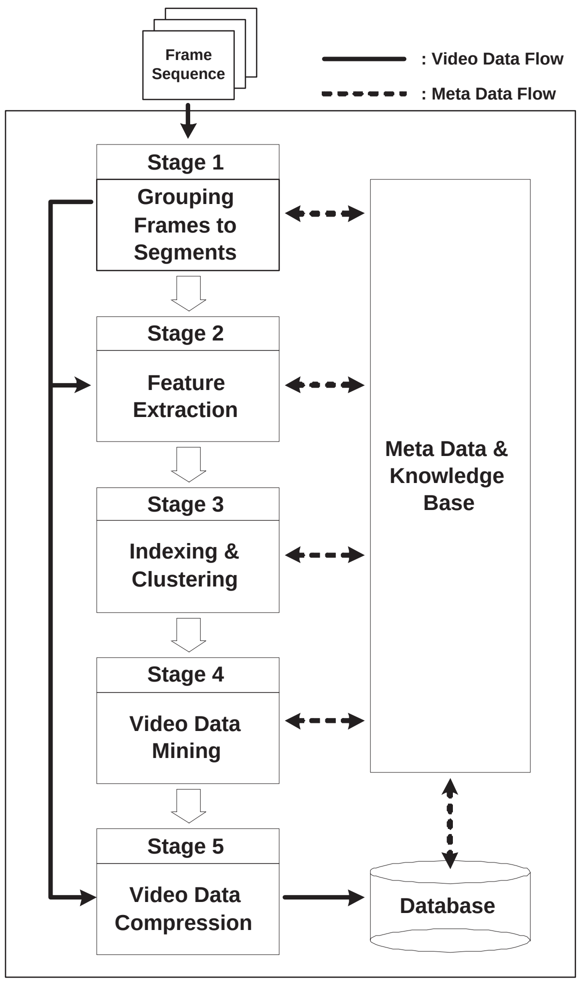

Fig. 1: Proposed Framework for Video Data Mining

- The first stage (Stage 1 in Figure 1) of our frame work for mining raw video data is grouping input frames to a set of basic units which are relevant to the structure of the video. This is one of the most important tasks since it is the first step to construct the building blocks of the video database, and to convert videos from raw materials to *data* with semantic information. In general, the most widely used basic unit in produced videos (i.e., movies, news videos) is a *shot* which is defined as collections of frames recorded from a single camera operation. Raw videos are usually recorded from a single fixed camera or multiple cameras with very limited camera motion without any camera on-off. Therefore, the concept of the shot is not relevant since whole video would be a shot by the above definition. In this paper, we investigate how to group the incoming frames into meaningful pieces in real time processing in which the traditional concept of shot is not applicable. This piece is called as *segment* to distinguish from shot. In addition to this linear decomposition, we build a hierarchical structure of segments. Therefore, we call our segmentation as *hierarchical segmentation*, and each segment is classified into the different category. Another advantage of this hierarchical segmentation is that it can give us various lengths of summaries for incoming videos automatically. More detail will be discussed in the next section.

- The second stage (Stage 2 in Figure 1) is characterizing each segment to cluster into similar groups, to discover unknown knowledge, and to detect interesting patterns. We need to extract the features such as motions, objects, colors, etc., to characterize these segments. Not only the features themselves but also how to represent them in comparable forms is very important because we need to compare the decomposed segments to characterize them as mentioned in the above. For our framework, we consider three features (motions, objects, colors) extracted from each segment. Among these features, *motion* is investigated in this time, and the other features will be studied in near future. To extract motions, we use an accumulation of quantized pixel differences among all frames in a segment [19]. As a result, accumulated motions of segment are represented as a *two dimensional matrix*. The technique to compute motions is very cost-effective because an expensive computation (i.e., optical flow) is not necessary. Because the motions are represented as a matrix, comparison among segments is very efficient and scalable.

- The third stage (Stage 3 in Figure 1) of our framework is to cluster the decomposed segments into similar groups. In our clustering, we employ a multi-level hierarchical clustering approach to group segments with similar categories in the top level, and similar motions in the bottom level. We use K-Mean algorithm and cluster validity method [20] due to its simplicity and efficiency. This clustering is a fundamental step for future knowledge discovery and pattern detection.

- The next stages (Stage 4 and 5 in Figure 1) are actual mining of raw video sequences processed throughout the above three stages, and video data compression for storage of these raw videos. The Meta Data & Knowledge Base in the figure is a module to store the results from each stage and provide the necessary information to the stages. The example of knowledge and patterns that we can discover and detect are object identification, object movement pattern recognition, spatio-temporal relations of objects, modeling and detection of normal and abnormal (interesting) events, event pattern recognition. We plan to develop techniques to perform the above mining tasks in near future. Also, a suitability and availability of various video compression techniques including MPEG will be investigated to store these video data in database physically.

The remainder of this paper is organized as follows. In Section 2, we describe a technique to group incoming frames into segments. A motion feature extraction technique is discussed in Section 3. In section 4, we propose a multi-level hierarchical clustering approach to group segments based on the categories, and the motions. The experimental results are discussed in Section 5. Finally, we give our concluding remarks in Section 6.

2. VIDEO SEGMENTATION

In this section, we discuss the detail of the technique to group the incoming frames into semantically homogeneous pieces by real time processing (we called these pieces as ‘segments’ as mentioned in the previous section). We first, look at the existing video partitioning techniques based on the concept of ’shot’ to figure out what the limitations and the problems they have when they are applied to raw videos in which the definition of shot cannot be applicable. Then we introduce a novel technique to decompose this type of videos.

2.1. Existing Techniques for Video Segmentation

In many literatures, the process for video segmentation is referred to as *shot boundary detection* (SBD) in general since they dealing with shot as a unit for segmentation. This SBD has been an area of active research. Many techniques have been developed to automatically detect transitions from one shot to the next. These schemes differ mainly in the way that the inter-frame difference is computed. The main idea of these techniques is that if the difference between the two consecutive frames (see Figure 2(a)) is larger than a certain threshold value, then a shot boundary is considered between two corresponding frames. The difference can be determined by comparing the corresponding pixels of two images [21]. Color or grayscale histograms can also be used [22]. Alternatively, a technique based on changes in edges has also been developed [23]. Other schemes use domain knowledge [24] such as predefined models, objects, regions, etc. Hybrids of the above techniques have also been investigated [25, 26, 27, 28, 29].

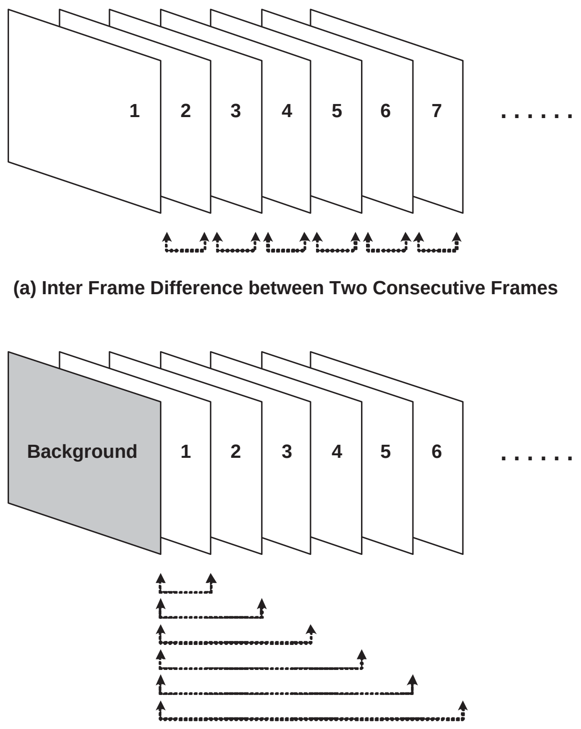
1
2
3
4
5
6
7
......
(a) Inter Frame Difference between Two Consecutive Frames
Background
1
2
3
4
5
6
......
(b) Inter Frame Differences with Background Frame

Fig. 2: Frame Comparison Strategies

However, this technique is not working for the raw videos in which there is little camera motions in most sequences. The dotted curve in the bottom of Figure 3 shows the color histogram differences between two consecutive frames in a raw video sequence. Note that this sequence was taken from a crowded hallway in a building, and digitized as 5 frames per second. As shown by this curve, there is not much difference between two consecutive frames. In fact, most of them are less than 10 %. In other words, if we use the differences between consecutive frames, most frames are to be considered as very similar. Therefore, it is difficult to find clear boundaries for segments. To address this drawback, we propose a new technique for raw video segmentation in the following subsection.

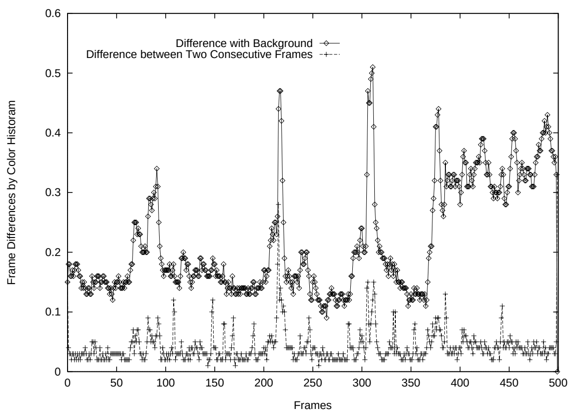

Fig. 3: Two Frame Comparison Strategies

2.2. New Technique for Raw Video Segmentation

The idea of new technique is very simple. Instead of comparing two consecutive frames, we compare each frame with a *background* frame as shown in Figure 2(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. In this work, we manually select a background frame using similar approach in [9]. The solid curve in the top of Figure 3 shows the color histogram differences between a background with each frames in the sequence. The differences are magnified so that segment boundaries can be found more clearly. The algorithm to decompose a raw 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 6 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. Usually the bin size is 128, 64 or 32 if the RGB value of a pixel in the original frame is 256. As a result, a background frame (F^B) is represented as follows using a bin with the size n. Note that P_T is representing the total number of pixels in a background or any frame.

$$\begin{aligned} F^B &= bin^B \\ &= (v_1^B, v_2^B, v_3^B, ..., v_n^B) \end{aligned} \quad (1)$$

where $\sum_{i=1}^{n} v_i^B = P_T$.

- Step.2: Each frame (F^k) arrived to system is represented using the same way to represent the background in the previous step as follows.

$$\begin{aligned} F^k &= bin^k \\ &= (v_1^k, v_2^k, v_3^k, ..., v_n^k) \end{aligned} \quad (2)$$

where $\sum_{i=1}^{n} v_i^k = P_T$.

- 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.

$$\begin{aligned} D^k &= \frac{F^B - F^k}{P_T} \\ &= \frac{bin^B - bin^k}{P_T} \\ &= \frac{\sum_{i=1}^{n}(v_i^B, -v_i^k)}{P_T} \end{aligned} \quad (3)$$

- Step.4: Classify D^k into 10 different categories as follows based on its value. Assign a corresponding category number (C_k) to the frame k.
  - Category 0 : $D^k < 0.1$
  - Category 1 : $0.1 \le D^k < 0.2$
  - Category 2 : $0.2 \le D^k < 0.3$
  - Category 3 : $0.3 \le D^k < 0.4$
  - Category 4 : $0.4 \le D^k < 0.5$
  - Category 5 : $0.5 \le D^k < 0.6$
  - Category 6 : $0.6 \le D^k < 0.7$
  - Category 7 : $0.7 \le D^k < 0.8$
  - Category 8 : $0.8 \le D^k < 0.9$
  - Category 9 : $D^k \ge 0.9$

- Step.5: For real time on-line processing, a temporary table such as Table 1 is being filled in and maintained. To do these and build a hierarchical structure from a sequence as mention in section 1, compare C_k with C_{k-1}. In other words, compare the category number of current frame with the one of previous frame. We can build a hierarchical structure from a sequence based on these categories which are not independent from each other. We consider the lower categories contain the higher categories as shown in Figure 4.

  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 become finding category boundaries in which we find a starting frame (S_i) and an

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

Table 1: Segmentation Table

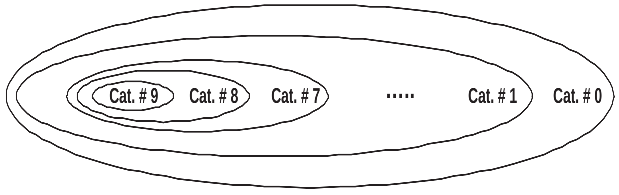

Fig. 4: Relationships (Containments) among Categories

ending frame (E_i) for each category i. The following algorithm is showing how to find these boundaries.

- If $C_{k-1} = C_k$, then no segment boundary occurs, and continue to 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 (β), we ignore this segment since it is too short to carry to any semantic contents. In general, this value β is one second. In other words, we assume that the minimum length of a segment is one second.

• Step.6: As mentioned in the previous section, without any extra computation, we can have several different versions of summaries for the incoming video which have different lengths, in other words, different levels of abstraction. The simple method is to pick all frames which category value is greater than or equal to C, where $1 \leq C \leq 9$. As results, we can have up to 9 different versions of summaries.

3. MOTION FEATURE EXTRACTION

In this section, we describe how to extract and represent motions from each segment decomposed from a raw 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 whole shot which is a collection of frames in our previous works [19]. 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 unexpected 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. It's all items are initialized with zeros.

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

And AMM_S which is a matrix of which items are average as computed as follows.

$$AMM_S = \begin{pmatrix} \frac{t_{11}}{n-1} & \frac{t_{12}}{n-1} & \frac{t_{13}}{n-1} & \cdots & \frac{t_{1c}}{n-1} \\ \frac{t_{21}}{n-1} & \frac{t_{22}}{n-1} & \frac{t_{23}}{n-1} & \cdots & \frac{t_{2c}}{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{t_{r1}}{n-1} & \frac{t_{r2}}{n-1} & \frac{t_{r3}}{n-1} & \cdots & \frac{t_{rc}}{n-1} \end{pmatrix} \quad (5)$$

- **Step.3**: Compare all the corresponding quantized pixels in the same position of two consecutive 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 without any increasing or decreasing.
- **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.

$$TM_S = \sum_{i=0}^{r}\sum_{j=0}^{c} t_{ij}, \quad AM_S = \sum_{i=0}^{r}\sum_{j=0}^{c} \frac{t_{ij}}{n-1} \quad (6)$$

As seen in these formulas, TM is a 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.

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.

$$Each\ Pixel\ Value\ = \\ 256\ -\ Corresponding\ Matrix\ Value \tag{7}$$

4. SEGMENTS CLUSTERING

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*.

For this multi-level clustering, we adopted K-Mean algorithm and cluster validity method studied by Ngo et. al. [20] 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 features 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 μ_1, μ_2, ..., μ_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 [30]. 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 value by Step.3, goto 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 [31]. 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.

5. EXPERIMENTAL RESULTS

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

- How well does the proposed segmentation algorithm to group incoming frames work?
- How well do TM, AM and the proposed algorithm for clustering of segments work?

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.

5.1. 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 (Total Motion and Average Motion) will be discussed in the following subsection. 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 and disappearing with various directions. The segment # 4 (Category # 2) represents this type of content which a person is appearing and disappearing in this case.

Segment No.	Starting Frame No.	Ending Frame No.	Segment Length	Cat. (C_k)	Total Motion (TM)	Avg. Motion (AM)
1	206	219	14	2	63	4.5
2	206	214	9	3	28	3.1
3	206	211	6	4	15	2.5
4	207	209	3	5	3	1.0

Table 2: Segmentation Table for Figure 5

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, 3,871 in the row of cat. #3 indicates the sum of the number of frames (the second column) from the category # 9 to the category # 3. As seen in this table, the higher category segments can be hierarchical summaries for the lower category segments.

Category	No. of Frames	No. of Frames Accumulated	No. of Segments	Avg. No. of Frames / Segment
Cat. # 0	2877	17,635	-	-
Cat. # 1	6533	14,758	309	47.8
Cat. # 2	4354	8,225	216	38.1
Cat. # 3	3580	3,871	183	21.2
Cat. # 4	244	291	36	8.1
Cat. # 5	32	47	10	4.7
Cat. # 6	12	15	4	3.8
Cat. # 7	3	3	1	3
Cat. # 8	0	0	0	0
Cat. # 9	0	0	0	0

Table 3: Overall Segmentation Results for Test Set

5.2. Performance of TM, AM and Clustering

Before we discuss the performance of the proposed algorithm for clustering, we show some examples of TM, and AM in Table 2. Figure 7 shows $TMMI$ and $AMMI$ for the segments (#1, #2, #3 and #4) in Figure 5. Throughout this figure, we can see that the TMs and the AMs represented by $TMMI$s and $AMMI$s are able to measure the exact amounts(degrees) of the motions in each segment accurately.

As mentioned in the previous section, first, the segments are clustered by the categories assigned to segments. In the next level, each cluster is partitioned into smaller sub-clusters using AM. 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 sizes of object(s) are distinguished by the category, in other words, the segments in the higher categories have relatively lager or more objects. On the other hand, the average motions, represented by AM can distinguish the amounts (degrees) of motions in different segments.

6. CONCLUDING REMARKS

The example of knowledge and patterns that we can discover and detect from the raw video sequences are object identification, object movement pattern recognition,

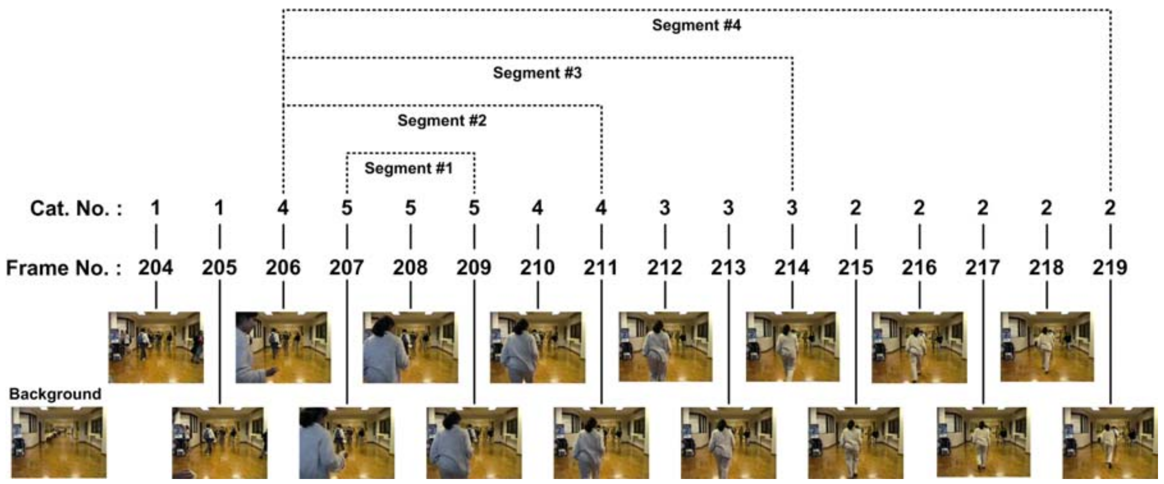
Segment #4
Segment #3
Segment #2
Segment #1
Cat. No. : 1 1 4 5 5 5 4 4 3 3 3 2 2 2 2 2
Frame No. : 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219
Background

Fig. 5: Segmentation example

Category	Segments	AM
2		2.3
		1.7
3		1.9
		1.2
4		1.5
		2.0
5		1.5
		2.5

Fig. 6: Sample Clustering Results

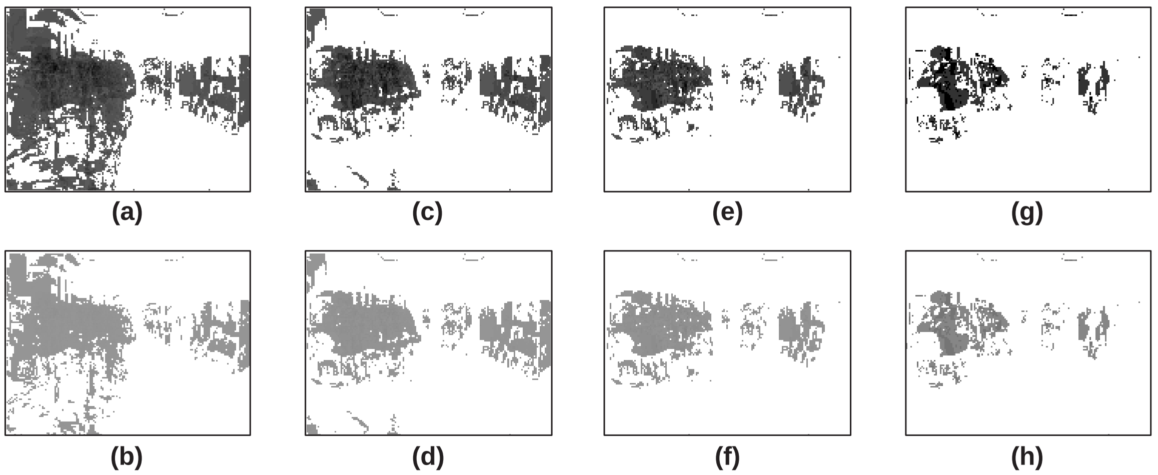

Fig. 7: (a) and (b) : TMMI and AMMI of Segment #1, (c) and (d) : TMMI and AMMI of Segment #2, (e) and (f) : TMMI and AMMI of Segment #3, and (g) and (h) : TMMI and AMMI of Segment #4

spatio-temporal relations of objects, modeling and detection of normal and abnormal (interesting) events, event pattern recognition. For this raw video data mining, in this paper, we propose a general framework to perform the fundamental tasks which are temporal segmentation of video sequences, feature (motion in our case) extraction, and clustering of segments. Although our experimental data set are limited, the results are showing that the proposed framework is performing the fundamental tasks effective and efficiently. In the future study, we will consider the other features (objects, colors) extracted from segments for more sophisticated clustering and indexing. Also, a suitability and availability of various video compression techniques including MPEG will be investigated to store these video data in database physically.

7. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] P.K. Singh and A.K. Majumdar. Semantic content-based retrieval in a cideo database. In *Proc. of International Workshop on Multimedia Data Mining (MDM/KDD'2001)*, pages 50–57, San Francisco, CA, August 2001.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. Coen. The future of human-computer interaction or how i learned to stop worrying and love my intelligent room. *IEEE Intelligent Systems*, 14(2):8–10, March 1999.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] E. Ardizzone and M. Cascia. Automatic video database indexing and retrieval. *Multimedia Tools and Applications*, 4:29–56, 1997.

[22] H. Yu and W. Wolf. A visual search system for video and image databases. In *Proc. IEEE Int'l Conf. on Multimedia Computing and Systems*, pages 517–524, Ottawa, Canada, June 1997.

[23] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying scene breaks. In *Proc. of ACM Multimedia '95*, pages 189–200, San Francisco, CA, 1995.

[24] R. Lienhart and S. Pfeiffer. Video abstracting. *Communications of the ACM*, 40(12):55–62, December 1997.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] P.J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987.

[31] A. K. Jain. *Algorithm for Clustering Data*. Prentice Hall, 1988.