Video Content Representation and Indexing using Hierarchical Structure

Suh-Yin Lee, Chi-Yi Wu and Duan-Yu Chen
Department of Computer Science and Information Engineering
National Chiao Tung University

{sylee, ciwu, dychen}@csie.nctu.edu.tw

Abstract

An efficient representation and indexing of video content are proposed using hierarchical structure. The hierarchical structure is constructed through shots grouping and scenes clustering. The video representation provides browsing capabilities for digital video databases. The video indexing supports more efficient content-based queries and retrieval capabilities for digital video databases. To implement the video representation and indexing system, hierarchical video structure is constructed. For constructing the hierarchical video structure, some techniques of video processing must be investigated and implemented, including segmenting the video as shots, grouping the shots into scenes and clustering the scenes. The process is designed to work on MPEG-II sequence, where only a partial decoding is required.

Keywords

Video content representation, hierarchical structure, indexing, clustering.

1. INTRODUCTION

With the advance of computer technology and electronic storage, the potential of digital video is growing rapidly. The rapid development of video and multimedia applications has enabled users to handle large amounts of visual information. However, tools and algorithms for effective organization and management of video databases and for content-based search and retrieval are still limited.

Most existing approaches organize the video-content by clustering shots on the basis of the similarity among the visual contents contained in their key frames. Approaches in [1] present an optimal extraction of the most characteristic scenes, which is accomplished by clustering similar scene feature vectors and selecting a limited number of cluster representatives. In addition to key frames, temporal content variations in a shot can be used. Story structures are extracted by using time-constrained clustering and Scene Transition Graph in [2,3] and story units are identified by temporally connecting different shot clusters.

We propose mechanisms of shots grouping and scenes clustering to construct hierarchical video structure. The hierarchical video structure represents and indexes video content efficiently. The primary tasks are depicted in Fig. 1. We first segment a video into shots, which usually mean the duration of a continuous action. Secondly, the similar shots within constrained duration will be grouped into scenes. After that, the similar scenes will be clustered together. Finally, a hierarchical video structure is built.

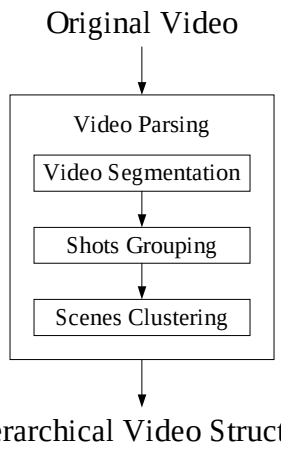


Fig. 1. Video parsing process

2. OVERVIEW OF HIERARCHICAL VIDEO STRUCTURE

We use the hierarchical structure to provide video content representation and indexing. We dissect a video into three levels, including shot, scene and cluster as depicted in Fig. 2.

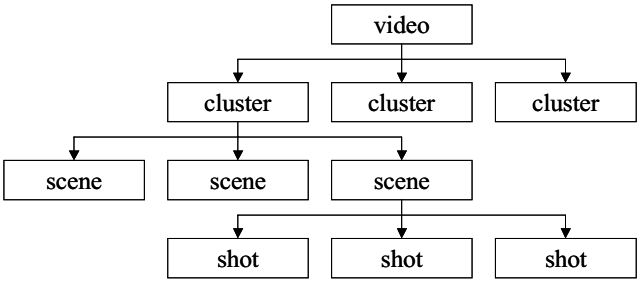


Fig. 2. Hierarchical Video Structure

The basic level is shot, which means a continuous action. It can be extracted by shot segmentation. The next higher level is scene. Similar shots are grouped into the same scene. The highest level is cluster, which is composed of similar scenes.

3. DESIGN METHOD AND PROCEDURE

In this section, we will present the design methods on video content representation and indexing. Section 3.1 describes our proposed scheme on video segmentation. In section 3.2, the process that groups shots into scenes is introduced. In section 3.3 we will present our proposed method on scenes clustering.

3.1 Video Segmentation

Some approaches in scene change detection have been proposed. It can be noted that most of the approaches detect occurrence of scene change frame by frame. However, the scene change does not happen on every frame. Therefore, it is not necessary to detect every frame sequentially for finding scene change. In shot segmentation, we use GOP-based approach of scene change detection [4], which saves more processing time than scene change detection frame by frame.

The workflow of the GOP-based scene change detection approach is shown in Fig. 3.

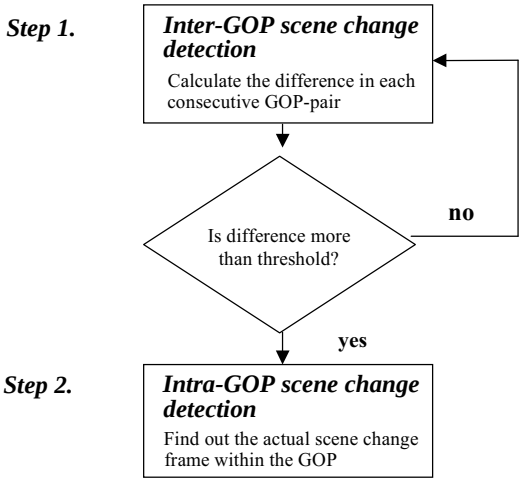


Fig. 3. Workflow of the GOP-based scene change detection approach.

In the first step (Inter-GOP), we detect the possible occurrence of scene change GOP by GOP. If the possible occurrence of scene change is detected, the check process will enter the second step (Intra-GOP), which finds out the actual frame of scene change within the GOP.

3.2 Shots Grouping

We group similar shots into scenes. Logically a scene can be considered as an essence of a video, because a scene means the occurrence of a whole event. For example, scenes of activities in a house, scenes of activities in a car and so on constitute a video. Our goal of shots grouping is to group the semantically related shots as scenes in a video.

In section 3.2.1, we describe the definition in shots grouping. In section 3.2.2, we present the algorithm of shots grouping.

3.2.1 Definition in shots grouping

In this section, we define some terms used in our approach of shots grouping.

1. **Key frame** *Fk* and *Fl:* we set the first I-frame, *Fk* and the last I-frame, *Fl* of a shot as the key frames.
2. **Image feature of a frame** *FTE*: Since video sequences are compressed in MPEG- II, DC images are adopted to present key frames, which are typically 64 times smaller than the original frames (8*8discrete cosine transform blocks are used). We separate the whole DC image to several

regions. By summing all DC values in each region, we can get a frame image as Fig. 4.

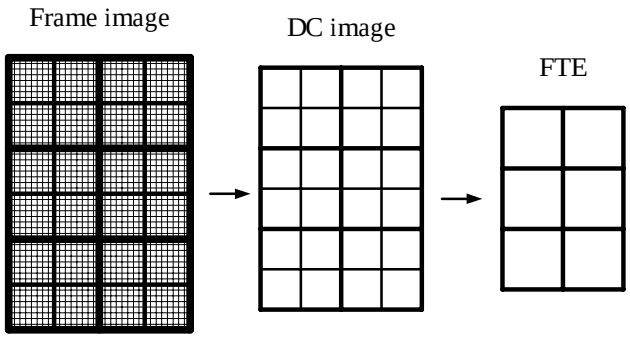


Fig. 4. Frame Image to FTE

3. **Average image feature of the shot** *AFTE*: we sum up the image features of all the I-frames in a shot, and divide by the number of I-frames to get the average image feature of the shot. An $AFTE_n$ of shot *n* is shown as Eq. (1) and Fig. 5.

$$AFTE_n = \frac{1}{K}\sum_{i=0}^{K} FTE_{ni} \qquad (1)$$

K is the number of I-frames in shot *n and* FTE_{ni} is the ith FTE of shot *n*.

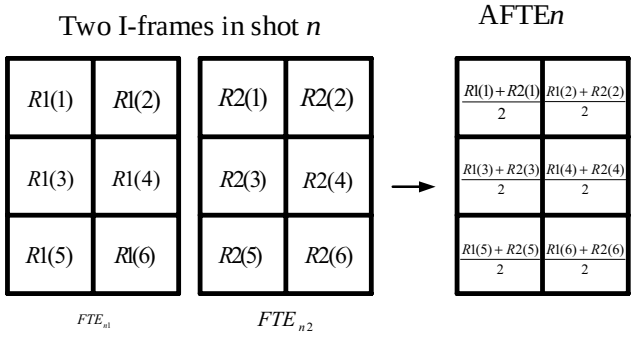


Fig. 5. Average Image feature of the shot n

4. **Image distance of two image features**: the image distance is calculated by summing all distances between regions in two image features [5]. Each region R_{ki} in image feature FTE_k has a unique correspondence to a region R_{nj} in the other image feature FTE_n. The image distance of two image features *iDiff* of FTE_k and FTE_n is shown as Eq. (2) and Fig. 6.

$$iDiff(FTE_k, FTE_n) = \sum_{i,j} \left| R_{ki} - R_{nj} \right| \qquad (2)$$

R_{ki} is the ith region in FTE_k and R_{nj} is the jth region in FTE_n, where the R_{nj} is the matched region to R_{ki} with the smallest distance. In Fig. 6, regions on two sides of arrowhead are matching regions.

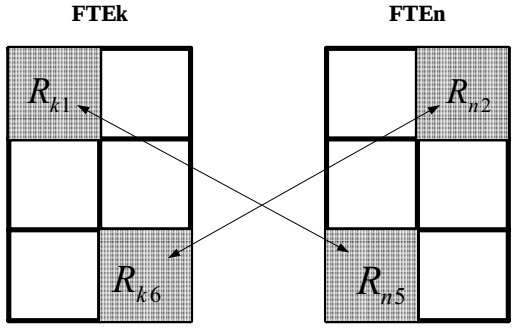


Fig. 6. Comparsion of FTEk with FTEn by matching regions from each region of FTEk with each region of FTEn

5. **Image distance of shots**: the image distance *SIMG* of *shot i* and *shot j* is represented as Eq. (3).

$$SIMG(i,j) = W_k \times DiffK(i,j) + W_a \times DiffA(i,j)$$

$$DiffK(i,j) = \min\begin{Bmatrix} iDiff(Fk(i), Fk(j)) \\ iDiff(Fk(i), Fl(j)) \\ iDiff(Fl(i), Fk(j)) \\ iDiff(Fl(i), Fl(j)) \end{Bmatrix}$$

$$DiffA(i,j) = iDiff(AFTE(i), AFTE(j)) \qquad (3)$$

DiffK in Eq. (3) is the minimum of image distance of key frames between two shots. *DiffA* is image distance of the average image feature between two shots. *Wk* and *Wa* are the weights of *DiffK* and *DiffA*, respectively. We take into consideration both the image distance of the key frame and distance of the average image feature in the determination of shots distance.

6. **Time distance function of shots**: besides small image distance, similar shots also possess the relation of time locality. Thus, we set the time distance function STIM of shot *i* and shot *j* as Eq. (4).

$$STIME(i,j) = Dis(i,j) \qquad (4)$$

$Dis(i, j)$ in Eq. (4) is the frame distance of shot *i* and shot *j*. In other words, it represents the location of the first frame of shot *j* subtracts that of the last frame of shot *i* if shot *j* is behind shot *i* in time order. From this equation we can see, the farther the distance of the shots, the weaker the time relation between these shots is.

7. **Distance function of shots**: After defining the image distance of shots and time distance function of shots, we combine these two distances to evaluate the distance of two shots, which is defined in distance function *DSHOT* of shot *i* and shot *j* as Eq. (5).

$$DSHOT(i, j) = W_i \times SIMG(i, j) + W_t \times STIME(i, j) \quad (5)$$

Wi and *Wt* in Eq. (5) are the weights of *SIMG* and *STIME*, respectively. The smaller the *DSHOT(i,j)* is, the more similar shot *i* and shot *j* are.

3.2.2 Algorithm of shots grouping

In a movie, there is a situation that the same shot may appear again after some other shots. In this situation, the shots of repeated occurrence and other shots in between them usually have a semantic relation. Thus we consider this sequence of shots may belong to the same scene. For example, if the shot of global view of a park is first presented, the shots of trees, flowers, people's walking and the blue sky are followed, and then the shot of global view of the park is presented again, we can treat this sequence of shots as the scene of a park in a video. According to these characteristics of scenes, we can detect scenes by finding the shots of repeated occurrence within a short range, which can be called similar shots, because similar shots usually have a time locality. For each intermediate shot in between the similar shots, we consider they have semantic relation with the similar shots, thus they belong to the same scene as the similar shots. For example, suppose we have a video sequence with 12 shots, which have been assigned group id according to their similarity:

A, B, C, A, D, E, F, G, H, I, D, J

The first shot and fourth shot are similar, thus they have the same group id, A. And the fifth shot and 11th shot are similar, thus they have the same group id, D. According to the above discussion that any intermediate shot in between the similar shots belongs to the same scene as similar shots, the first scene is the sequence of ABCA, because shot B and C are intermediate shots in between the similar shots A. Analogously, shot E, F, G, H and I are intermediate shots in between the similar shots D, thus the second scene is DEFGEHD. Based on the above discussion, we can build a scene structure through the following algorithm.

- **Alg. 1. Shots Grouping Algorithm**

Input: all shots of a video:*{shot 0, shot 1,..., shot i}*

Output: the scene structure of the video

1 Assign 0 to the group id and story-unit id of *shot 0*. Initialize the number of group, *NumGrp* and the number of scene, *NumScn* both to 1.

2 If all the shots have been processed, then quit; otherwise get the next shot and denote it as shot *i*.

3 Compute the distances between shot *i* and the shots before shot *i*. But not every shot before shot *i* is compared, because similar shots usually have time locality. Thus we define a comparison range, *N = 10,* which is determined through the statistics in the experiment that if the shot distance of two shots is over *N*, the probability of similarity between two shots is very small. Only the shots in the comparison range will be compared with shot *i*. The comparison process is called subroutine **[ComputeDistance]**.

4 Find the minimum distance value from *N* compared-pairs and test if the minimum distance value is less than the predefined threshold. We set *Tshot* as the shot with the minimum distance value compared with shot *i*.

<If yes>

4.1 Merge group id of shot *i* to the group id of *Tshot*.

4.2 Merge the scene id of shot *i* to the scene id of *Tshot*

4.3 Update the video structure: call subroutine **[UpdateStructure]**.

4.4 Go to step 2.

<Otherwise>

4.5 Assign new group id, *NumGrp* to shot *i*.

4.6 Assign new scene id , *NumScn* to shot *i*.

4.7 Increase the number of group, *NumGrp* and the number of scene, *NumScn* both with 1.

4.8 Go to step 2.

[**ComputeDistance**]

Input: two shots, named shot *s1* and *s2*, respectively.

Output: the distance value *DSHOT* of the two shots

$$DSHOT(s1, s2) = W_i \times SIMG(s1, s2) + W_t \times STIME(s1, s2)$$

[**UpdateStructure**]

Input: Current shot, group and scene id of the video

Output: The updated shot, group and scene structure of the video

1. Denote the group of the current shot as group *A*.

2. For any shot whose group id is not equal to group *A* and in between two shots of group *A*, merge scene id of the shot to that of the shot of group *A*.

In this algorithm, we first compare each shot with its previous shots within a comparison range. If two shots are dissimilar, we assign the current shot a new group id and scene id. In the contrast, if two shots are similar, then the shots in between these two similar shots will be assigned the scene id. The example is illustrated in the following.

Before update:

Shot *i*	Shot *i+1*	Shot *i+2*	Shot *i+3*	Shot *i+4*
Group *1*	Group *2*	Group *3*	Group *4*	Group *1*
Scene *9*	Scene *10*	Scene *11*	Scene *12*	Scene *9*

After update:

Shot *i*	Shot *i+1*	Shot *i+2*	Shot *i+3*	Shot *i+4*
Group *1*	Group *2*	Group *3*	Group *4*	Group *1*
Scene *9*	Scene *9*	Scene *9*	Scene *9*	Scene *9*

An example of update process is: before update, shot *i+1*, shot *i+2* and shot *i+3* are in between shot *i* and shot *i+4* whose group id are both 1. After update, story-unit id of any intermediate shot in between shot *i* and shot *i+4* is replaced by 9.

After the shots grouping algorithm, each shot will have a scene id. One scene id represents a scene, thus the shots with the same scene id will belong to the same scene.

3.3 Scenes Clustering

In scenes clustering, our criterion is to make the number of scenes in each cluster approximately the same and the variation of the size is not so big. We employ concept of modification of Self-Organizing Map (SOM) [6,7] based on scene feature vectors to achieve the goal.

3.3.1 Definition in scenes clustering

In this section, we define some terms used in our approach of scenes clustering.

1. **Scene Feature** SF*:* we use the scene image as the feature of scene. The scene image of an example scene *n* is shown in Fig. 7.

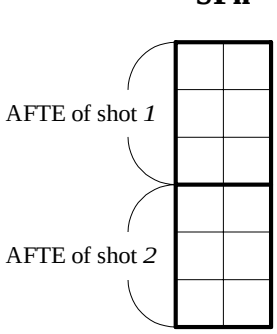


Fig. 7. Scene Image n example
(Two shots in a scene *n*)

The scene feature includes average image feature AFTE of all shots in the scene.

2. **Average scene feature of a scene** ASF: we sum up the average image features AFTE of all the shots in a scene, and divide by the number of the shots to get the average scene feature of the scene. An example scene *n* with two shots is shown in Fig. 8. We denote ASF as scene center.

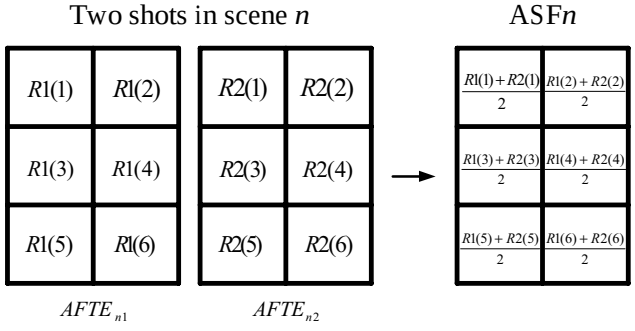


Fig. 8. Average scene feature of the scene n

3. **Cluster Feature** CF*:* we use the cluster image as the feature of a cluster. A cluster image *n* consisting of scene SF*1* and SF*2* is shown in Fig. 9.

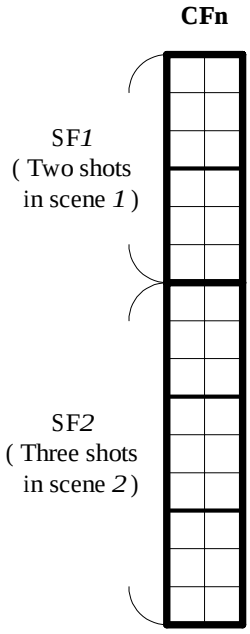


Fig. 9. Cluster Image n
(Two scenes in a cluster *n*)

The cluster feature includes scene feature SF of all scenes in the cluster.

4. **Average cluster feature of a cluster** *ACF*: we sum up the average scene features ASF of all the scenes in a cluster, and divide by the number of the scenes to get the average cluster feature of the cluster as Fig. 10. We denote ACF as cluster center.

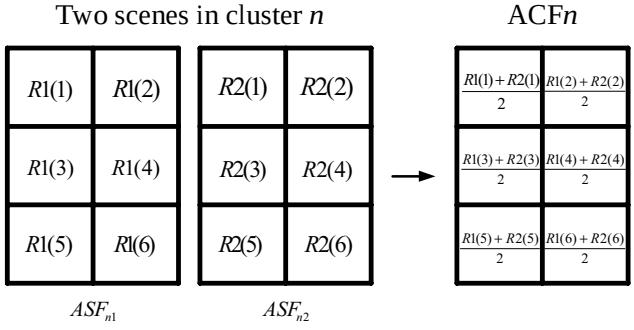


Fig. 10. Average cluster feature of the cluster n

5. **Image distance between scene and cluster**: The distance between scene and cluster is analogous to image distance of the image feature. The difference is that unit of image distance is region but unit of distance between scene and cluster is average image feature AFTE.

   The distance between $scene_k$ and $cluster_n$ is calculated by summing all distances between average image features AFTE in scene *k* and those in cluster *n* and dividing by the number of shots in $scene_k$ or $cluster_n$. If the number of shots in $scene_k$ is smaller than that in $cluster_n$, each $AFTE_{ki}$ of $shot_{ki}$ in $scene_k$ has a unique correspondence to an $AFTE_{nj}$ of $shot_{nj}$ in $cluster_n$. Oppositely if the number of shots in $cluster_n$ is smaller than that in $scene_k$, each $AFTE_{nj}$ of $shot_{nj}$ in $cluster_n$ has a unique correspondence to an $AFTE_{ki}$ of $shot_{ki}$ in $scene_k$. In other words, this is a one-to-one matching. The detail is shown as Eq. (6) and Fig. 11.

$$scDiff(k,n) = \frac{1}{L}\sum_{i,j} iDiff(AFTE_{ki}, AFTE_{nj}) \quad (6)$$

L in Eq. (6) is the smaller one between the number of shots in $scene_k$ and that in $cluster_n$. $AFTE_{ki}$ is the ith AFTE in $scene_k$ and $AFTE_{nj}$ is the jth AFTE in $cluster_n$, where $AFTE_{nj}$ is the matched AFTE to $AFTE_{ki}$ with the smallest distance.

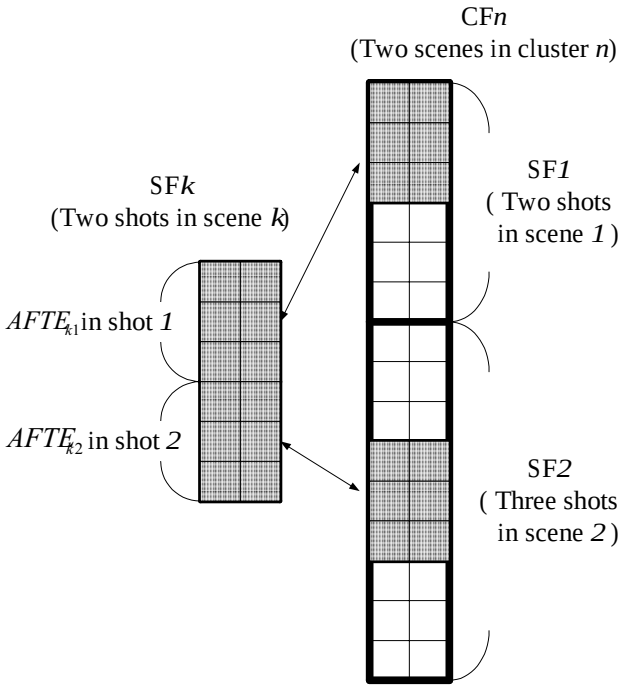


Fig. 11. Distance between scene k and cluster n

6. **Image distance between scene center and cluster center**: The distance between scene center and cluster center is analogous to the distance between scene and cluster. The difference is that distance between scene and cluster is calculated by average image feature AFTE but distance between scene center and cluster center is calculated by average scene feature ASF and average cluster feature ACF. The distance between scene center *k* and cluster center *n* is shown as Eq. (7).

$$AscDiff(k,n) = iDiff(ASF_k, ASF_n) \quad (7)$$

7. **Number distance function***:* We set the number distance function SNumI of cluster *n* as Eq. (8).

$$SNumI(n) = \frac{C}{SceneNum(n)} \quad (8)$$

C in Eq. (8) is constant. From this equation we can see, the larger the number of scenes in a cluster, the smaller the SNumI is.

8. **Distance function between scene and cluster**: Our criterion of scenes clustering is to make the variation of cluster size not so big. Thus, we take into consideration the number of scenes in a cluster in the definition of distance function between scene and cluster as shown in Eq. (9).

$$DScene(k,n) = scDiff(k,n) - SNumI(n) \quad (9)$$

The larger the number of scenes in a cluster, the smaller the SnumI is, and the larger the Distance is.

9. **Distance function between scene center and cluster center**: The distance function between scene center and cluster center is similar to the distance function between scene and cluster. It is shown as Eq. (10).

$$ADScene(k,n) = AscDiff(k,n) - SNumI(n) \quad (10)$$

10. **Inaccuracy of cluster**: For getting inaccuracy of cluster *n*, we sum all the distance function between all scene centers in the cluster *n* and the cluster center *n* as shown in Eq. (11).

$$CInaccuracy(n) = \sum_{i=0}^{K} ADScene(i,n) \quad (11)$$

K is the number of scenes in cluster *n*.

3.3.2 Algorithm of scenes clustering

Considering time complexity, we use a simple one pass clustering method for scenes clustering. The scenes clustering algorithm is shown as Alg. 2.

- **Alg. 2. Scenes Clustering Algorithm**

Input: all scenes of a video:*{scene 0, scene 1,..., scene i}*

Output: the cluster structure of the video

1 Create a new cluster and assign 0 to the cluster id of scene *0*. Initialize the number of clusters *ClusterCount* to 1. Call subroutine **[NewCluster]**.

2 If all scenes have been processed, then quit; otherwise get the next scene and denote it as scene *i*.

3 Compute the value of distance function between scene *i* and all clusters. The computation process is called subroutine **[ComputeSCDistance]**

4 Find the minimum distance value and test if the minimum distance value is less than the predefined threshold. We set *Tcluster* as the cluster with the minimum distance value computed with scene *i*.

<If yes>

4.1 Assign *Tcluster* to cluster id of scene *i*.

4.2 Update the cluster information: call subroutine **[UpdateCluster]**

4.3 Go to step 2

<Otherwise>

4.4 Create a new cluster and assign *ClusterCount* to cluster id of scene *i*. Call subroutine **[NewCluster]**.

4.5 Assign *ClusterCount* to cluster id of scene *i*.

4.6 Increase the number of clusters, *ClusterCount.*

4.7 Go to step 2

[ComputeSCDistance]

Input: scene *s1* and cluster *c1*

Output: the distance value, *DScene* of the scene and cluster

$$DScene(s1,c1) = scDiff(s1,c1) - SNumI(c1)$$

[NewCluster]

Input: Current scene *s1* and cluster *c1*

Output: The new cluster information

1. Assign 1 to the number of scenes in cluster *c1*.

2. Assign the number of shots in scene *s1* to the number of shots in cluster *c1*.
3. Copy SF of all shots in scene *s1* to CF of cluster *c1*
4. Assign ASF of scene *s1* to ACF of cluster *c1*.

[UpdateCluster]

Input: Current scene *s1* and cluster *c1*

Output: The updated cluster information

1. Increase the number of scenes in cluster *c1*.
2. Update the number of shots in cluster *c1*.

$$ShotNum(c1) = ShotNum(s1) + ShotNum(c1)$$

3. Add SF of scene *s1* to CF of cluster *c1*
4. Update ACF of cluster *c1*.

$$ACF_{c1} = \frac{ACF_{c1} \times (SceneNum(c1) - 1) + ASF_{s1}}{SceneNum(c1)}$$

After the simple one pass clustering, we get approximate scenes clustering. For getting better scenes clustering result, we adjust each clustering iteratively as Alg. 3.

- **Alg. 3. Scenes Clustering Adjusting Algorithm**

Input: the cluster structure of the video constructed by Alg. 2

Output: the adjusted cluster structure of the video

1. Initialize variable *count* to 0 and variable *iteration* to 0.
2. If *count* is large than the number of clusters or *iteration* is larger than ten, then quit; otherwise get the next cluster and denote it as cluster *i*.
3. Compute the Inaccuracy of the cluster *i*. The computation process is called subroutine **[ClusterInaccuracy]**.
4. If the distance is larger than the predefined threshold, then go to step 5; otherwise increase the value of *count and go to step 2*.
5. If all scenes in cluster *i* have been processed, then set *count* to 1 and go to step 2; otherwise get the next scene and denote it as scene j.
6. Compute the distance function between scene *j* and all clusters. The compute process is called subroutine **[ComputeSCADistance]**.
7. Find the cluster with the minimum distance value computed with scene *i* and denote it as cluster *Mcluster*. If *Mcluster* is not equal to i, call subroutine **[AdjustScene]**. Go to step 5.

[ClusterInaccuracy]

Input: cluster *c1*

Output: the inaccuracy of the cluster *c1*

$$CInaccuracy(n) = \sum_{i=0}^{K} ADScene(i,n)$$

[ComputeSCADistance]

Input: scene *s1* and cluster *c1*

Output: the distance value, *ADScene* of the scene and cluster

$$ADScene(k,n) = AscDiff(k,n) - SNumI(n)$$

[AdjustScene]

Input: scene *s1* and old cluster *c1* and new cluster *c2*

Output: The updated scene information and cluster information

1. Decrease the number of scenes in cluster *c1*.
2. Increase the number of scenes in cluster *c2*.
3. Update the number of shots in cluster *c1*.

$$ShotNum(c1) = ShotNum(s1) - ShotNum(c1)$$

4. Update the number of shots in cluster c2.

$$ShotNum(c2) = ShotNum(s1) + ShotNum(c2)$$

5. Remove SF of scene *s1* from CF of cluster *c1*
6. Add SF of scene *s1* to CF of cluster *c2*
7. Update CAF of cluster *c1*.

$$CAF_{c1} = \frac{CAF_{c1} \times (SceneNum(c1) + 1) - SAF_{s1}}{SceneNum(c1)}$$

8. Update CAF of cluster *c2*.

$$CAF_{c2} = \frac{CAF_{c2} \times (SceneNum(c2) - 1) + SAF_{s1}}{SceneNum(c2)}$$

9. Change cluster id of scene *s1* to *c2*

4. EXPERIMENTAL RESULTS

The proposed algorithms were tested using several video sequences from video database. Some results obtained from a video sequence of total duration 3 minutes (3050 frames) are presented in the following figures. The video sequence was first segmented into 24 shots and then shots were grouped into 8 scenes.

Finally, scenes were clustered into 4 clusters. Fig. 12 illustrates all scenes in the video sequence. The clustering results are shown in Fig. 13. Let us focus on the scene 7 in Fig. 12. The two shots in scene 7 do not seem to belong to the same scene by looking at the pictures on the paper. They are grouped into the same scene in our system because the two shots are similar in colors. The two shots are both out-of-doors and their backgrounds are sky and sea, which are both blue. From Fig. 12 and Fig. 13, the results of our system are satisfactory. The shots in the same scene and scenes in the same clustering are similar in content.

It is somewhat subjective how good the video structures are. The way of performance testing is to ask users to evaluate the system. We have requested several users to construct video structures after viewing the test video sequences. The grouping and clustering results of our system are very close to the results of users.

5. CONCLUSION

The proposed hierarchical video structure provides efficient video content representation and indexing. Users can obtain general overall view of contents through browsing the hierarchical structure. Because the hierarchical structure is already gained, video indexing can be established thereafter. Through the video indexing, users can retrieve any scene in any cluster quickly.

For constructing the hierarchical video structure, some works of video processing have been done, including segmenting a video as shots, grouping the shots into scenes and clustering the scenes. We propose an efficient method of shot segmentation, which only examines frames on possible scene changes. We design an algorithm to group shots into scenes, which form logical building units of a video. Finally, the two algorithms for scenes clustering are proposed. The experimental results validate the effectiveness of the system. More testing and refinement are worthy of further investigation.

6. REFERENCES

[1] Nikolaos D. Doulamis, Anastasios D. Doulamis, Yannis S. Avrithis and Stefanos D. Kollias, “Video Content Representation Using Optimal Extraction of Frames and Scenes,” IEEE conference on Image processing, Vol. 1, pp. 875-879, 1998.

[2] M. Yeung, B. L. Yeo, and B. Liu, “Extracting Story Units from Long Programs for Video Browsing and Navigation,” Proc. IEEE Conf. on Multimedia Computing and Systems, pp. 296-305, 1996.

[3] M. Yeung, B. L. Yeo, and B. Liu, “Video Browsing using clustering and scene transitions on compressed sequences,” Proc. IEEE Conf. on Multimedia Computing and Systems, 1996.

[4] J. L. Lan, “Video Summary and Browsing Based on Story-Unit for Video-on-Demand Service,” Master thesis, National Chiao Tung University, Dept. of CSIE, June 1999.

[5] Alan Hanjalic, Reginald L. Lagendijk, Jan Biemond, “Automated High-Level Movie Segmentation for Advanced Video-Retrieval Systems,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No 4, 1999.

[6] Duane DeSieno, “Adding a Conscience to Competitive Learning”, IEEE International Conference, Vol. 1, pp. 117-124, 1988.

[7] J. Michael Rozmus, “The Density-Tracking Self-Organizing Map”, IEEE conference on Neural Network, Vol. 1, pp. 44-49, 1996.

[8] Kohonen, T., “Self-Organizing Maps”, Proceedings of the IEEE, Vol. 78, No 9, September 1990.

[9] Marc Cavazza,Roger Green and Ian Palmer, "Multimedia Semantic Features and Image Content Description", IEEE Multimedia modeling, pp. 39-46, 1998.

[10] A.W Wardhani and R Gonzalez, "Automatic Image Structure Analysis", IEEE Multimedia computing and System, pp.180-188, 1998.

[11] Jeongnam Youn, Ming-Ting Sun, Fellow, IEEE, and Chia-Wen Lin, "Motion Vector Refinement for High-Performance Transcoding", IEEE Transactions on Multimedia, Vol. 11, pp. 30 –40, 1999.

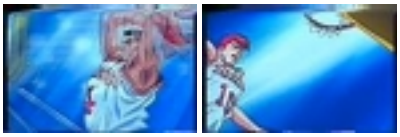
[12] A. Mufit Ferman and A. Murat Tekalp, "Efficient Filtering and Clustering Methods for Temporal Video Segmentation and Visual Summarization", Journal of Visual Communication and Image Representation, 1998.

[13] Adnan M. Alattar, "Wipe Scene Change Detector for Use with Video Compression Algorithms and MPEG-7", IEEE Transactions on Consumer Electronics, Vol. 441, pp. 43-51, 1998.

Scene 1

Scene 2

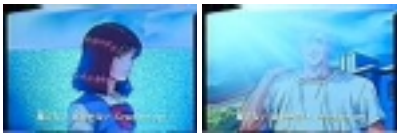
Scene 3



Scene 4

Scene 5

Scene 6



Scene 7

Scene 8

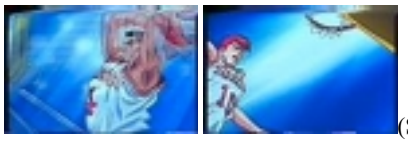
Fig. 12. The 8 scenes of the test video sequence

Cluster 1


(Scene 1)

Cluster 2

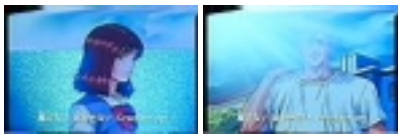

(Scene 2)


(Scene 3)


(Scene 7)

Cluster 3


(Scene 4)


(Scene 6)


(Scene 8)

Cluster 4

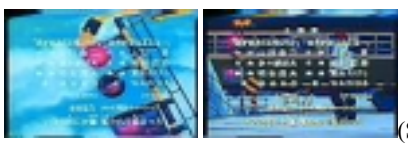

(Scene 5)

Fig. 13. The four clusters generated by the clustering algorithms