

# Unsupervised Scene Change Detection Techniques In Feature Domain Via Clustering And Elimination

Serkan Kiranyaz, Kerem Caglar\*, Bogdan Cramariuc and Moncef Gabbouj.

Signal Processing Laboratory, Tampere University of Technology, Tampere, Finland.

\*Nokia Mobile Phones, PO Box 88, FIN-33721 Tampere, Finland

E-mail: [serkan@cs.tut.fi](mailto:serkan@cs.tut.fi), [kerem.caglar@nokia.com](mailto:kerem.caglar@nokia.com)

## ABSTRACT

*The last generation video compression techniques such as MPEG-4 and H.263+, allow presence of shot-cut detection methods that are usually built into the encoders. Such methods are used to extract key-frames to be encoded as intra frames. This paper presents automated scene change detection techniques, which mainly work over these key-frames to extract scene frames. Proposed techniques aim at unsupervised or semi-automatic scene detection schemes. Preliminary results show that the proposed framework provides near optimal scene frames suitable for video summarization by choosing semantically important key-frames as the scene frames.*

## 1. INTRODUCTION

As generation of video and video databases becomes more popular in the recent years, an increasing need has emerged for research on how to represent and summarize video as efficiently as possible. Similarly last generation video compression standards and techniques such as MPEG-4 [1] and H.263+ [2] are supplying solutions for video recording and transmission over several bandwidth requirements and therefore, used in wide range of applications. So it is essential to design video representation algorithms while using the features of the underlying compression schemes in order to achieve an effective unified solution.

Both MPEG-4 and H.263+ supports key-frame extraction methods that can be built into the encoders for intra frame encoding. Even though there is no mandatory requirement on how to choose the intra frames, today in most of the MPEG-4 and H.263+ encoders, intra frames are shot-cut frames with some possible forced intra frames. So the idea is to encode any frame as an intra frame whenever considerable shot changes are detected at the encoder side. In this way bit-rate efficiency is considerably increased. In forced intra mode, without any shot-detection encoders might encode frames as intra frames at some fixed rate. This of course gives some key-frame redundancy but maintains the overall quality of the video.

Video representation and summarization relies on the video structure, which can be hierarchically described in three segmentation layers: Shots, scenes and whole video [3]. In shots layer, one or more frames

can be chosen to represent the segments that are so called key-frames. For video summarization purposes key-frames might include quite a large amount of redundancy such as repetition of the similar shots during the run-time of the video. Therefore, by eliminating such redundancy, scene frames should be extracted out of the key-frames to achieve efficient video summarization.

In this paper, we propose unsupervised (adaptive parameter or threshold values setting) scene detection techniques, which mainly work over key-frames of H.263+ or MPEG-4 encoded video but can also be applied to any compressed or uncompressed video sequences. The proposed framework removes the redundant key-frames and thus extracts semantically important scene frames out of key-frames. The rest of the paper is organized as follows. In section 2, the proposed framework is presented. In section 3, the comparison and effectiveness of the proposed scene detection techniques are validated via real-world video clips. Concluding remarks and discussions are in section 4.

## 2. PROPOSED SCENE DETECTION TECHNIQUES

In order to extract scene frames out of key-frames, we developed two distinct techniques: Scene Detection (SD) by Minimum Spanning Tree (*MST*) clustering [4] and SD by Nearest Neighborhood Elimination (*NNE*). Once the compression scheme (MPEG-4 or H.263+) segments the video sequence into shots and extracts the key-frames as the first frame of each shot, both techniques apply feature extraction to the key-frames in order get an inter-similarity measure between them. Any feature extraction method (based on color, motion, DCT coefficients, etc.) can be used for the proposed techniques. In this work we used HSV color histogram BINS as the feature vectors. These feature vectors are then used for inter-similarity measure between key-frame for both techniques.

### 2.1. SD by MST clustering

*MST* is an efficient and widely used clustering tool. We use HSV bin vectors of the key-frames representing the nodes in *MST*. In order to find the weight (similarity measure) of the branch between two nodes, we use normalized Euclidean Distance (*D*) between associated feature vectors  $\bar{x}$  and  $\bar{y}$ , such as:

$$D = \frac{|\bar{x} - \bar{y}|}{1 + |\bar{x}| + |\bar{y}|}$$

By using the distance between two vectors as the weight of the *MST* branches, the tree is formed. Then the tree branches are sorted from the largest branch weight to the smallest one. At the beginning, the whole tree is a single cluster. In order to create a new cluster, it is sufficient to break the (next) largest branch. Figure 1 represents a sample *MST* example with three clusters.

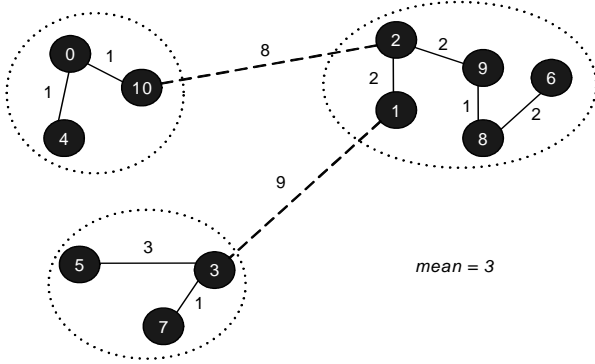


Figure 1

Every time a branch is broken and thus a new cluster is created, the nodes inside the cluster can be assumed to show high similarity than the nodes outside the cluster. Knowing that those nodes represent the key-frames, in this way we grouped all the key-frames that are similar to each other. Once a sufficient number of clusters are achieved, it is straightforward to choose one or more key-frame(s) as the scene frame(s), which can represent the scene (cluster). Therefore, for efficient SD, it is extremely important to know where to stop clustering. Usually in supervised methods, a preset threshold value for branch weight comparison is experimentally found and thereof applied for the decision whether or not to break the branch. This might work on some video sequences but obviously not guaranteed to work on every video sequence.

In our unsupervised technique, this threshold value is found adaptively. In a sample case as shown in Figure 1, the optimal threshold value should be in between 3 and 8 since there are two comparably long (8 and 9) inter-cluster branches and short intra-cluster branches. So the aim is to break the longest two branches and stop clustering. For this the threshold value should be somewhere in between the smallest inter-cluster (broken) branch and the largest intra-cluster (unbroken) branch. This might justify the usage of the *mean* of all the branch weights ( $mean=3$ ) as the threshold value since it satisfies the condition. There might however be circumstances that this method leads to some redundant cluster creation and hence redundant scene frame detections. The probability of redundant scene detection is especially increased when there are significantly more intra-cluster branches than the inter-cluster branches. For example in the sample *MST* in Figure 1, if there happen to be one or more intra-cluster branches that have weights below 3, the *mean* value would be less than 3 and as a result, there would indeed be 4 clusters and node 5 is to be a new but redundant

scene frame. In order to avoid such a case, we propose a 2-step weighted root mean square algorithm to determine the threshold value as follows:

- Step 1: Calculate mean ( $\mu$ ) and variance ( $\sigma$ ) of the branch weights ( $w_i$ ).
- Step 2: Calculate the threshold value as the weighted root mean square of the branch weights as follows:

$$Threshold = \sqrt{\frac{\sum_i^N (k_i * w_i^2)}{N'}}$$

where  $N' = \sum_i^N k_i$  and  $N = \text{no. of branches}$ .

$$k_i = \begin{cases} (\text{int}) \left( \frac{(w_i - \mu)}{\sigma} \right) & \text{if } w_i \geq \mu \\ 1 & \text{if } w_i < \mu \end{cases}$$

Equation 1

As clearly seen in this equation, big weight values affect the threshold value more significantly than the smaller ones. Furthermore, the weight values around the mean (in a window of variance) have no effect at all on the threshold value. As a result we can summarize the algorithm as follows:

- I. Extract key-frames and their feature vectors.
- II. Use feature vectors as the nodes of the *MST* and a similarity distance measure (i.e. Euclidean) as the branch weights (distance between nodes).
- III. Form *MST*.
- IV. Sort branches from biggest to smallest weights.
- V. Find the mean and variance of the branch weights.
- VI. Find the *Threshold* value according to Equation 1.
- VII. Break the next largest branch if it is bigger than *Threshold* value. If not, stop clustering and proceed to VIII
- VIII. For each cluster choose appropriate one or more key-frame(s) to represent the cluster (scene) and thus extract scene frames.

The total number of clusters gives the optimal number of scenes. Finally, scene frame(s) extracted from each cluster can be used for the summarization of the video sequence.

## 2.2. SD by NNE

In this technique, the idea is to keep only one key-frame as a scene frame and eliminate all the other similar ones. This is basically achieved without any pre-clustering or even any kind of tree forming. Initially the first key-frame of the video sequence is chosen as the first scene frame. By using the same normalized similarity measure as in the previous section, similar key-frames to the current scene frame are eliminated from the list of key-frames. In a supervised mode, a fixed threshold value is used to find which key-frames are to be eliminated. Once all the similar key-frames are eliminated, the next “present” key-frame in the time-line is chosen as the

second scene frame since it is not eliminated and therefore, it is a new scene frame. Then all the existing key-frames are compared with this new scene frame and similar ones are again eliminated. The algorithm proceeds up to the elimination of the last key-frame while always honoring next surviving key-frame as the scene frame.

In order to achieve unsupervised SD by *NNE*, the optimal threshold value used for key-frame elimination should be found beforehand. Similar to *MST* method, this value should yield the algorithm to come up with the optimal number of scenes to represent (summarize) the video clip. Since the similarity measure between two key-frames is found by normalized Euclidean distance, this threshold value is somewhere in between  $0$  and  $1.0$ . The threshold value  $0$  will not eliminate any key-frames so that all the key-frames will be chosen as scene frames. The threshold value  $1.0$  will eliminate all the key-frames during the first step of the iteration and yield to only one scene frame, which is first key-frame. So starting from  $0$  threshold, increasing threshold values yields decreasing number of scene frames. As a result, for a fixed threshold we can summarize the *NNE* algorithm as follows:

- I. Choose the first/next key-frame as the first/next scene frame.
- II. Compare the normalized similarity measure between the current scene frame and next key-frame with the threshold value: if it is less than the threshold, eliminate that key-frame otherwise keep it.
- III. Once all the key-frames in the list are compared with the current scene frame, proceed to the next un-eliminated key-frame. If such key frame exists, go to I, otherwise, terminate iterations.

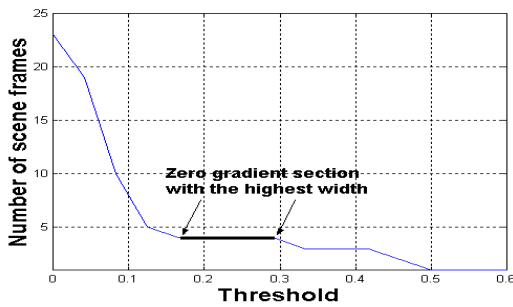


Figure 2

In ideal case the key-frames, which belong to the same scene should be similar to each other, so gives close distance values but rather different than the key-frames of the other scenes, thus results in large distance values. In this way once the elimination of the close (intra-scene) key-frames are completed, one can expect a region that no elimination is possible until the inter-scene frame eliminations occur. Since the number of scene frames is the difference between total number of key-frames and number of eliminated key-frames, such a region can be directly detected from the number of scene frames vs. threshold curve. In ideal case we can expect a zero-gradient plateau in such a curve but to have a robust algorithm we try to detect the lowest gradient section with the largest width as shown in

Figure 2. Once this region is found, the scene frame number yields the optimal number of scenes with which this video clip can be represented (summarized) and thus the scene frames are accordingly extracted.

### 3. EXPERIMENTAL RESULTS

For the experiments in this section, the video streams are captured in QCIF size at 25 fps and H.263+ encoded at 128 Kb/s. A built-in YUV histogram based shot-detection method in H.263+ encoder extracts key-frames with some additional forced intra key-frames (period of 1000). HSV 8x4x4 color histogram bins are used as the feature vectors. As mentioned in the previous section, the similarity measure is the normalized Euclidean distance between two feature vectors.

In order to find the optimal number of scene frames for SD by *NNE* algorithm, linearly distributed threshold values are generated and by detecting the maximum number of threshold values which yield the same scene number gives us the optimal scene number. For SD by *MST* method, in order to have comparable results with SD by *NNE* method, only one key-frame with minimum time-line index in each cluster is chosen as the scene frame for that cluster.

Figure 3 and 4 show the results of two simulations. The top sections of these figures show the extracted key-frames by the encoder. For the comparison of the proposed methods, the bottom section of the figures are divided into 2 parts: the bottom-up parts show scene frames by *NNE* method and bottom-down parts show scene frames by *MST* clustering method.

The experimental results on scene frames detected by a group of users (i.e. semantically suitable for human perception system) over 10 different video clips are shown in Table 1.

### 4. CONCLUSIONS AND FUTURE WORK

This paper proposes unsupervised scene detection techniques in which scene frames are extracted among the key-frames that are detected by shot-detectors built-in the encoders. In fact both of the proposed techniques can eventually be applied to any video sequence; compressed or uncompressed but we use the advantage that the underlying compression technique (such as H.263+ or MPEG-4 video codec) removes most of the frame redundancy for scene detection by choosing the key-frames automatically for compression purposes. This indeed saves considerable amount of processing time on the behalf of clustering and elimination parts of each technique.

Although we use (HSV) color histogram for similarity measure, any useful semantic or visual feature can be integrated into the proposed techniques. Experimental results over several real-world video clips show that both techniques have high accuracy and efficiency in the extraction of the scene frames that are semantically and visually meaningful for human perception and therefore, they can effectively be used for video representation and summarization purposes.

For the future work, we are planning to experiment with different features, such as shape, color, texture or audio. Extensive experimental studies will be mainly on combining those in order to get better semantic features.

### 5. ACKNOWLEDGEMENT

The authors would like to thank Mr. Faouzi Alaya Cheikh for fruitful discussion related to the paper.

### 6. REFERENCES

[1] ISO/IEC 14496-2:2001 JTC 1/SC 29/WG 11 N5350, "Information Technology-Coding of Audio-

Visual Objects - Part 2: Visual ", International Organization for Standardization, Sydney, July-2001.

[2] ITU-T Draft, " Recommendation H.263 - Video Coding for low bitrate communication", International Telecommunication Union, November 1995.

[3] Y. Rui, T. S. Huang & S. Mehrotra, "*Exploring Video Structure Beyond The Shots*," Proc. of IEEE Intel. Conf. on Multimedia Comp. and Systems (ICMCS), June 28, 1998.

[4] Graham, R.L., and O. Hell, "On the history of the minimum spanning tree problem," *Ann. Hist. Comput.* 7, pp. 43-57. 1985.

NNE MST	Seq. 1 23 KFs	Seq. 2 21 KFs	Seq. 3 30 KFs	Seq. 4 27 KFs	Seq. 5 31 KFs	Seq. 6 63 KFs	Seq. 7 38 KFs	Seq. 8 39 KFs	Seq. 9 17 KFs	Seq. 10 26 KFs
<b>Number of detected scenes.</b>	4	9	5	4	13		5		15	9
<b>False Alarm</b>	0	4	1	0	4	3	1	4	2	0
<b>Missed.</b>	0	0	0	2	0	3	1	2	0	8
<b>Expected number of scenes</b>	4	5	4	6	9	12	5	10	13	17

Table 1: Comparison of the performance of MST and NNE methods



Figure 3: Key-Frames (top) and Scene Frames by *NNE* (bottom-up) and *MST* (bottom-down).



Figure 4: Key-Frames (top) and Scene Frames by *NNE* (bottom-up) and *MST* (bottom-down).