Compressed Domain Summarization of Digital Video

C.M. Chew[1] and M.S. Kankanhalli[1]

[1] School of Computing, National University of Singapore
{chewchor, mohan}@comp.nus.edu.sg

Abstract. Video data is usually voluminous and it is desirable that one be able to get a quick idea of the content before actually watching a video or downloading it from the web. In this paper, we present a video summarization algorithm that works in the compressed domain, in particular MPEG videos. The algorithm is based on an existing one used in the uncompressed domain. To adapt it for MPEG videos, we make use of a feature known as DC histogram that can be extracted from MPEG videos without full decompression. A video summarizer based on our algorithm is implemented and experiments are conducted to examine its effectiveness. Results show that the summarizer performs quite well to user expectations.

1 Introduction

With the recent and rapid advances in multimedia technology and computing power, digital video is increasingly becoming a popular and valuable information resource. However, the sheer amount of data in a video when compared with other forms of media such as text and audio makes management and manipulation very difficult. To reduce the amount of storage needed, most video clips are compressed into a smaller size using a compression standard such as MPEG [1, 2].

However, even after going through compression, compressed videos are still too big to transfer over general-purpose networks such as the Internet. Even if a user has a complete video clip, he or she may just want to view a summary of the video instead of watching it from the beginning till the end. Thus browsing and summarization tools that would allow the user to quickly get an idea of the overall content of the video footage are very useful. This functionality will become very important in the coming years when TV broadcasts worldwide will be done in a fully digital format.

Much of the research work done on video summarization methods has only focused on the uncompressed domain [3]-[7]. They cannot be used directly on compressed videos such as MPEG files. An MPEG video will have to be decompressed back to its original uncompressed form before the summarization can be performed. This is very time consuming and requires a huge amount of space, which defeats the original purpose of compressing videos. Moreover recompression back to compressed form after summarization may result in loss of picture quality due to re-quantization.

In this paper, we propose a video summarization algorithm that operates directly in the compressed domain. The algorithm uses an adaptive clustering method that has been employed in the uncompressed domain. This clustering method does not rely on shot detection techniques which most of the previous work, e.g. [8]-[10], have been based on. We also show how to extract a feature from MPEG videos without full frame decompression such that the feature still captures the essential information of the video frames.

2 Summarization Algorithm

2.1 Clustering Method

The main aim of our summarizer is to analyze a given video clip and extract the important frames that reflect the content changes of the video. We call these important frames *representative frames (R-frames)*. Since it is still not possible today for computers to understand the semantics of video images, we will base our summarizer on low-level features such as color, motion, etc. However, we do not assume that all low-level features change somewhat abruptly at shot boundaries. Instead we will use a clustering method to remove redundant frames and retain the important frames. In particular, we will extend an earlier work [7] and apply the technique described in that paper to compressed domain videos.

The nature of the spatial distribution of the points corresponding to video frames can be described as clusters connected by abrupt or gradual changes. So it is possible to divide a video V into N' clusters. We call these clusters *units*. If we represent frame 1 of all the units' representative frames using R_{p1} to $R_{pN'}$, V can then be described as:

$$V = U_1 \, O \, U_2 \, O \ldots O \, U_{N'}$$

where

- $U_i = \{ R_{pi}, R_{pi1}, \ldots, R_{p(i+1)} \}, \; i \in [1, N']$
- O is the temporal concatenation operation

Since the units are in temporal order, we can define the unit change as the difference between two consecutive representative frames, i.e.

$$Change(U_i) = D(R_{pi}, R_{p(i+1)})$$

In order to extract good representative frames, we should aim to divide the video into units that have very similar unit changes. This can be described as to minimize:

$$\sum_{i=1}^{N'-1} \sum_{j=i+1}^{N'-1} \left| Change(U_i) - Change(U_j) \right|$$

By having units that have very similar unit changes, we are actually dividing the videos into individual sections whose length corresponds to the amount of content changes within the particular section. For example, a high action scene is usually divided into more units than a relatively static scene since the content of the high action scene changes much quicker. This concept is what we will base our clustering algorithm on.

Our clustering algorithm works in an iterative fashion. We start initially with all the frames of the video and iteratively drop frames until the desired result is obtained. For a given video V with length N, suppose we want to extract N' representative frames. The feature of each frame in V is computed first. Then the video is partitioned in small units whose lengths are all L. All the units are temporally contiguous. For example, Figure 1 shows the partitioning with L=2 and L=3. So the units for L=3 are {(0,1,2), (2,3,4), (4,5,6), (6,7,8)}. In each unit, the unit change is computed, which is the distance between the first frame and the last frame of the unit.

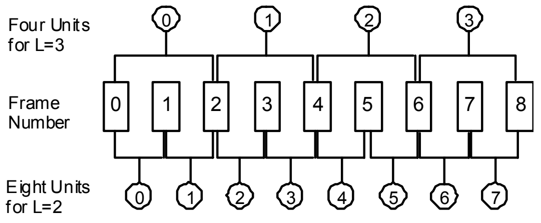


Figure 1: Partitioning a Video into Units

After all the unit changes are computed, these values will form an array of length $K=\lceil N/(L-1)\rceil$. Because our objective is to extract representative frames according to frame content changes, the values in the array do reflect the actual degree of content change in all the units. The values are then sorted in ascending order. After sorting, the elements that are located at the beginning represent the frames where there are small changes, while the units in the later part consist of frames having large changes.

By selecting a ratio $0<r<1$, we cluster the array into two clusters according to the value of unit change. The first cluster comprises of the smallest elements of the array and its length is K*r. We call this cluster the *small-change cluster*. The rest of the elements comprise the *large-change cluster*.

If the change of a unit belongs to the large-change cluster, we take all of its frames as part of the current extracted representative frames. If the change of a unit belongs to the small-change cluster, then we delete all the frames except the first and the last from the unit. The first and the last frames are retained as part of the current extracted representative frames. After the deletion process, K*r*(L-2) frames will be deleted.

Suppose the number of frames left is N". If N' is greater than or equal to N" then we have achieved the desired result and we can stop the algorithm. If not, we regroup all the retained frames as a new video and repeat the last procedure. With the decrease in the number of frames after each iteration, small units are consequently clustered together. A unit will physically span across more frames in the original video. So it will represent a larger range of frame changes. Frames are deleted from the sequence after each iterative process, so the overall number of frames left will decrease each time. Therefore, no matter how small a number may be required, the algorithm will converge to the desired requirement.

Another characteristic of the clustering method is that it is general, i.e. it can use any low-level content as a feature to calculate the difference between two frames in a video. For example, the feature could be color, motion, shape or texture. The feature used in our summarizer is discussed in the next section.

After extraction of representative frames from a video, users can make use of a browsing tool that provides interactive functions for traversal between these frames. However this process requires a lot of feedback between the user and the computer, since thousands of representative frames can be generated for an hour of video. This may be too time consuming for some users and therefore, a grazing view is preferred in this case, i.e. a video summary is generated from the representative frames which the user will view in order to get a general idea about the original video.

One way of generating the summary is to output the representative frames sequentially to a new video. However, this approach is not really useful because when the summary is played at the normal frame rate, users will find it very difficult to grasp information from it. This is because the pace of the summary will be too fast and jerky. To solve this problem, *representative sequences (R-sequences)* are used. An R-sequence consists of a representative frame plus its several successive frames. The length of the following frames is called the *smoothing factor S*. From experiments, it has been found that the smoothing factor has to be greater than or equal to 5 in order to obtain a visually pleasing result.

2.2 Feature Extraction

In order to work directly on an MPEG video, we need to find a feature with these properties: 1) it can be extracted efficiently from the video; and 2) it can capture the essence of each video frame such that analysis can still be carried out without much error. In this paper, we will utilize a feature known as DC histogram that is based on the DC image proposed by B.L. Yeo and Bede Liu [11].

The compression of MPEG video is carried out by dividing each frame of the video into 8x8 pixel blocks. The pixels in the blocks are transformed into 64 coefficients using Discrete Cosine Transform. The DC term $c(0,0)$ is related to the pixel values *f(i,j)* via the following equation:

$$c(0,0) = \frac{1}{8}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)$$

In other words, the value of the DC term is 8 times the average intensity of the pixel block. If we extract the DC term and subsequently the average intensity of all the blocks in an image, we can use the average values to form a reduced version of the original image. This smaller image is known as the DC image.

Although the size of the DC image is only 1/64 of that of the original image, it still retains significant amount of information. This suggests that scene operations of a global nature originally performed on the original image can also be applied on the DC image.

For the sake of efficiency, we will only consider luminance blocks when forming the DC image. This is because the eye is sensitive to small changes in luminance, but not in chrominance. Thus we can discard the chrominance information without affecting the quality of the extracted DC image much.

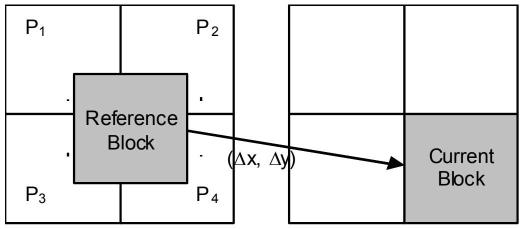
It is trivial to extract a DC image from an I-frame in MPEG since all blocks are intra-coded. The average intensity of each block is 1/8 the DC-coefficient of that block. However, extracting DC images from P and B frames involves more effort as motion compensation and differential coding are used in these frames.

To obtain the DC coefficients of a P frame, we need to use the coefficients in the reference I or P frame. This is illustrated in Figure 2. Consider the current block in a new P frame. This block can be reconstructed from the information in the reference block in a previous I or P frame referred to by the motion vector. Thus, by using the DC coefficients of P1, P2, P3, and P4, the DC coefficient of the current block, D_c, can be reconstructed by

$$D_c = \frac{1}{64}\sum_{i=1}^{4}[DC(P_i)]_{00} w_i h_i + e$$

where $[DC(P_i)]_{00}$ refers to the DC coefficient of the block Pi and e is an error term. In practice, e is found to be small and thus the first term is a good enough approximation of the desired DC coefficient.

Figure 2: Determining DC coefficient of a P frame



The same technique can also be applied to B frames which are forward-only or backward-only predicted. For bi-directional frames, two DC coefficients are first obtained in each prediction direction and the final coefficient value is the average of the two.

Although DC images are 1/64 the size of the original video frame size, they are uncompressed and thus takes up a substantial amount of space. For example, a DC image for a 320 x 240 video frame will occupy 40 x 30 = 1200 bytes. Assuming a frame rate of 30 fps, the total amount of space required for a 2 hour video is 1200 x 30 x 60 x 60 x 2 = 247.2 MB. This number will be even larger for higher resolution videos such as DVD and Digital TV.

To reduce the amount of data to be processed, we will use a DC histogram as the feature for each frame instead of the DC image. The histogram will be divided into 64 bins, i.e. each bin will account for 4 luminance values. Since luminance values ranges from 0 (black) to 255 (white), we can assume that this range is linear and thus values close together are similar. The cost of computing the histogram is very low and each frame now only takes up 64 x 2 = 128 bytes (assuming a 16-bit integer is used for the value of each histogram bin) regardless of the frame size. So a 2 hour video will only occupy 128 x 30 x 60 x 60 x 2 = 26.4 MB, a figure that average computers nowadays can handle quite easily. To calculate the difference between two histograms, the sum of the absolute bin-to-bin difference is taken.

3 Experimental Results

Evaluating the quality of a video summary is difficult as the factors to consider are highly complex and difficult to quantify computationally. Some methods based on shot detection have used the number of shots detected as their metric. However, as our algorithm is not based on shot detection, this metric cannot be used for our summarizer. Since there is no absolute measure of summarization quality available today, we decide to measure the quality of our summaries by user questioning.

For the experiment, we used 10 persons as our test subjects. Three video summaries generated using our summarizer were used as the test videos. Information about the summaries is shown in Table 1. P refers to the summarization percentage, L the unit length, r the clustering ratio and S the smoothing factor applied.

Video	**Genre**	**Original Length**	**Summarization Parameters**	**Summary Length**
A	Movie	1h 59m 34s	P=1.5%, L=5, r=0.3, S=5	5m 10s
B	News	21m 20s	P=2%, L=12, r=0.3, S=20	2m 17s
C	Movie	2h 28m 32s	P=0.2%, L=11, r=0.3, S=20	1m 51s

Table 1: Details of Video Clips used for User Evaluation

Before viewing the video summaries, the test subjects were given some information about each summary such as the genre and the aim of the summary. After viewing each summary, each person was then asked to rate the summary in four categories (Clarity, Conciseness, Coherence and Overall Quality) on a scale of 1 to 7, corresponding to worst and best respectively. At the end of the questionnaire, the person is then requested to rate the automatically generated summaries against their opinions of human-generated ones. Table 2 shows the average scores of the evaluation exercise.

Video	Clarity	Conciseness	Coherence	Overall Quality
A	5.2	5.7	4.8	5.3
B	6.0	6.2	5.8	6.0
C	5.2	5.2	5.2	5.2
Average	5.5	5.7	5.3	5.5

Table 2: Results of User Evaluation

From the table, we can see that on the whole, the summaries performed quite well with scores of over 5 in all but one category. Looking at the performance of each individual summary, Video A scored highly in conciseness but got the lowest coherence score among the three videos. This is expected since the aim of the summary is to portray the complete story of the original video as much as possible to the viewer within the 5 minutes. The smoothing factor has to be set to a relatively low value of 5 in order to keep the length of the summary within the desired length. This results in a certain degree of choppiness when viewing the video, thus affecting the coherence score.

Video B scored the highest in all categories. A news programme usually cycles between the news presenter and the video footage of the story currently presented. The luminance difference between these two kinds of scenes is usually quite large, so our video summarizer was able to extract all the segments in the programme. Coupled with the fact that a large smoothing factor of 20 was used, the resultant summary was very clear and smooth flowing.

Video C's score was the same for all the categories. Since the aim of this summary is to give a general idea of the movie's content rather than portraying the whole plot (which is the case for the summary of Video A), the parameters used for summarization (smaller percentage, larger unit size and smoothing factor) were different from those used for Video A. Therefore, the amount of content covered is less (accounting for the lower conciseness score) but the summary is more fluid (accounting for the higher coherence score).

The last question of the survey asked the viewer to rate the automatically generated summaries against their opinions of human-generated ones. The average score for this question is 5.00 (out of a maximum of 7), which means that the users agree to a limited extent that the quality of the summaries is comparable to that of human-generated ones. This is quite a good score for an automated summarizer. Since our summarizer is still based on a low-level feature, it is expected that the automatically generated summaries are still inferior to human-generated summaries that are based on semantics.

4 Summary

We have presented a video summarization algorithm that operates directly in the compressed domain (MPEG videos) without employing shot detection techniques. We extract the DC histogram feature from each frame of the MPEG video and use it together with an adaptive clustering method to extract representative frames to form the summary. User surveys conducted have shown encouraging results.

An area for further investigation is the use of other features such as motion vectors. We have explained that our clustering method is general, i.e. it can use any low-level content of a video as the feature for summarization. Therefore it would be interesting to study how different features affect the quality of summarization and the effectiveness of these features in the various genres of video.

References

[1] ISO/IEC 11172-2, Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s, Part 2: Video.

[2] ISO/IEC 13818-2, Generic Coding of Moving Pictures and Associated Audio Information, Part 2: Video.

[3] JungHwan Oh, Kien A. Hua, "An Efficient Technique for Summarizing Videos Using Visual Contents", *Multimedia and Expo, 2000. ICME 2000. IEEE International Conference, Vol. 2, pg 1167 – 1170, Jul 2000.*

[4] Yihong Gong, Xin Liu, "Generating Optimal Video Summaries", *Multimedia and Expo, 2000. ICME 2000. IEEE International Conference, Vol. 3, pg 1559 – 1562, Jul 2000.*

[5] D. DeMenthon, V. Kobla, D.Doermann, "Video Summarization by Curve Simplification", *Technical Report LAMP-TR-018, CS-TR-3916, University of Maryland, College Park, 1998.*

[6] Rainer Leinhart, Silvia Pfeiffer, Wolfgang Effelsberg, "Video Abstracting," *Communications of the ACM, Vol. 40, No. 12, Dec 1997.*

[7] X. Sun, M. Kankanhalli, "Video Summarization Using R-Sequences", *Journal of Real-Time Imaging, Vol. 6, No. 6, pp. 449-459, Dec 2000.*

[8] N. Gamaz, X. Huang, S. Panchanathan, "Scene Change Detection in MPEG Domain", *Image Analysis and Interpretation, IEEE Southwest Symposium, pg 12 – 17, Apr 1998.*

[9] Ali M Dawood, Mohammed Ghanbari, "Clear Scene Cut Detection Directly from MPEG Bit Streams", *IEEE Image Processing and its Applications, No. 465, Vol. 1, pg 285 – 289, Jul 1999.*

[10] Jongho Nang, Seungwook Hong, Youngin Ihm, "An Efficient Video Segmentation Scheme for MPEG Video Stream using Macroblock Information", *7th ACM international conference on Multimedia, pg 23 – 26, Oct 1999.*

[11] B. L. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video," *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 6, Dec 1995.*

[12] H.S. Chang, S. Sull, and S.U. Lee, "Efficient Video Indexing Scheme for Content-based Retrieval", *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 8, pp. 1269-1279, Dec 1999.*

[13] A. Hanjalic and H.J. Zhang, "An Integrated Scheme for Automatic Video Abstraction Scheme Based on Unsupervised Cluster-validity Analysis", *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 8, pp. 1280-1289, Dec 1999.*