ROBUST CLUSTERING-BASED VIDEO-SUMMARIZATION WITH INTEGRATION OF DOMAIN-KNOWLEDGE

Dirk Farin[1], *Wolfgang Effelsberg*[1], *and Peter H. N. de With*[2]

[1] Dept. of Computer Science IV
University of Mannheim
68131 Mannheim, Germany
`farin@informatik.uni-mannheim.de`

[2] CMG / Univ. of Technol. Eindhoven
5600 MB Eindhoven, Netherlands
`P.H.N.de.With@tue.nl`

ABSTRACT

Clustering techniques have been widely used in automatic video-summarization applications to group shots with comparable content. We enhance the popular k-means clustering algorithm to integrate user-supplied domain-knowledge into the cluster generation step. This provides a convenient way to exclude scenes from the summary which are *a-priori* known to be irrelevant. Furthermore, we added an additional, time-constrained clustering step preceding the scene clustering step to exclude short ranges with transitional content. This makes the algorithm robust to fading and wipe-effects in the input without requiring explicit cut detection.

1. INTRODUCTION

Video sequences are time-varying data streams which cannot be viewed easily at a glance. Hence, searching through video databases requires a large amount of time. Automatically generated video summaries are a popular aid for the user to get an impression of the video content. These can be a short version of the video (a video abstract) or a set of key-frames.

Many techniques for the automatic extraction of key-frames have been proposed. These techniques can be classified into two classes. The first class uses cut detection to separate the video into a large number of shots. Key-frames are obtained by choosing a representative frame of each shot (see [1]). The disadvantage of this approach is that errors in the cut detection are propagated into the key-frame selection process. Furthermore, the number of key-frames is directly coupled to the number of shots and cannot be adjusted by the user. Even if the content changes much within a single shot (consider a camera pan), only a single key-frame is extracted for the shot.

The second class consists of algorithms based on clustering [2, 3]. Usually, feature vectors are derived for all video frames, which are clustered subsequently. For each cluster, a single, representative key-frame is selected.

Applying conventional clustering algorithms to common video material still show the disadvantage of generating key-frames for irrelevant scenes. Examples are completely black images from pauses between two scenes or white pictures from fades to white or from flashlights. When the summary is generated in a well-defined application area, additional classes of images may be of no interest. For example, in a summarization of a news broadcast without audio, the user may want to exclude shots with the news-speaker, the program intro or the weather chart.

A further problem occurs if the video contains slow transitions between shots. Frames within these transitions show no stable content and should therefore not appear in the summary.

This paper presents a new clustering-based algorithm, providing a solution to these disadvantages. The problem of frames selected out of transitions is solved by a two-stage clustering process. The first stage provides a soft form of shot separation, determining periods of stable image content with good key-frame candidates. These candidates are then used in the second clustering stage to select the final set of key-frames. Image content which is not desired in the summary can be explicitly excluded by providing domain-knowledge in form of sample images of shots to exclude. This aspect is solved by modifying the clustering step to circumvent the building of clusters for these shots.

2. SUMMARIZATION ALGORITHM

Our algorithm is composed of three steps which operate from low-level features to semantically more meaningful data structures. As a first step, a small set of features is extracted from each input frame. These feature vectors are subsequently used to determine similarity between frames. The second step then groups time consecutive feature vectors to small *segments*. We define a segment as a small period in the video sequence (usually even smaller than a shot) such that the content in the segment is as static as possible. More specifically, no cut should be present in a segment. The third step combines the segments to clusters such that as much as possible of the input video content is covered by the clusters. Domain-knowledge is integrated by inhibiting the building of clusters in areas of the feature-space which are known to be irrelevant.

2.1. Feature Extraction

For each input frame n, a feature vector f_n is extracted. The subsequent steps of our algorithm work on arbitrary feature vectors. Thus, a variety of features can be used, provided that an appropriate distance measure $||f_a; f_b||$ can be defined which corresponds to visual similarity.

For our implementation, we have chosen to use quantized luminance histograms as feature vectors $f_n = (h_1, ..., h_m)$. We are using two different distance measures for the segment positioning and clustering steps. In segment positioning, the *sum of absolute difference* measure (SAD) is used, which is defined as

$$||f_a; f_b||_{SAD} = \sum_{i=1}^{m} |f_a(i) - f_b(i)|.$$

For the clustering step, the *Earth-Mover's Distance* (EMD) is used. This measure has been used by several authors in the context of image retrieval from large databases; see [4] for an in-depth description. In the one-dimensional case, the EMD can be determined efficiently as

$$||f_a; f_b||_{EMD} = \sum_{x=1}^{m} \left| \sum_{i=1}^{x} f_a(i) - f_b(i) \right|.$$

The reason for using two different distance measures is that the two steps operate on different time-scales. In short periods of time, the image contents varies less. Hence, the more sensitive SAD measure is used to accurately segment slow transitions. From a global perspective, shots with large distances in time can show large differences even when the semantic content is comparable. Therefore, the more liberal EMD is more appropriate for the high-level clustering step.

2.2. Determining Segment-Boundaries

Video sequences may contain gradual transition effects like fades and wipes between shots. It is desired that video frames from these transitions are not present in the final video summary. However, when two subsequent shots are grouped into the same cluster, clustering algorithms usually tend to select the transition frames as cluster centers because these features are mixtures of the features from both shots. To avoid the selection of those frames, we split the input video into short segments of about 4 seconds length. The objective of the first clustering step is to position the segment boundaries such that they are favourably positioned at cuts and within transitions. The intention is to obtain small segments of video with almost homogeneous content (see Fig. 1). Consequently, the frame in the middle of each segment will be a good key-frame candidate.

Our algorithm for positioning the segment boundaries is motivated by the time-constrained clustering technique described in [5]. However, that paper used the clustering technique to

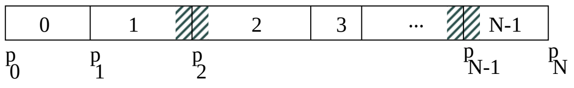


Fig. 1. The input video is divided into a large number of segments (N). Segment boundaries are positioned such that they coincide with cuts or that they are placed in the middle of transitions between shots (shown as shaded areas).

generate hierarchical summaries of existing key-frames, whereas we are using it for the low-level placement of segment boundaries.

Let p_i be the first frame of segment i. To determine the positions p_i, the total sum of inhomogeneity over all N segments is minimized. The inhomogeneity of a segment i is determined by summing up the distances between all frames in the segment and the mean feature vector s_i of the segment with

$$s_i = \frac{1}{p_{i+1} - p_i} \sum_{n \in [p_i\ ;\ p_{i+1})} f_n.$$

The positions of the segment boundaries p_i are chosen to minimize the total segment inhomogeneities:

$$\min_{p_1, \ldots, p_{N-1}} \underbrace{\sum_{i \in [0;N)} \underbrace{\sum_{n \in [p_i\ ;\ p_{i+1})} ||s_i; f_n||_{SAD}}_{\text{inhomogeneity of segment } i}}_{\text{minimized over all segments}}.$$

If there are enough segments available, the above optimization will place segment boundaries into transitions between shots. In the usual case in which many more segments are available than shots, long shots will be split into several segments. According to the optimization criterion, the segment length will depend on the amount of change in the video. Static parts will be assigned longer segments, while fast changing parts will be split into shorter segments.

Since the computational complexity of an exact optimization would be too high for practical implementations, we are using a time-continuous variant of the k-means algorithm for optimization. The algorithm approaches the global optimum by performing many local optimizations as follows (see Fig. 2):

1. Distribute p_i equally spaced over the full length of the video;
2. for all pairs of adjacent segments $[p_{i-1}\ ;\ p_i)$ and $[p_i\ ;\ p_{i+1})$ set p_i to

$$\underset{p_i}{\text{minarg}} \underbrace{\sum_{n \in [p_{i-1}\ ;\ p_i)} ||s_{i-1}; f_n||_{SAD}}_{\text{left segment inhomogeneity}} + \underbrace{\sum_{n \in [p_i\ ;\ p_{i+1})} ||s_i; f_n||_{SAD}}_{\text{right segment inhomogeneity}};$$

3. repeat step 2 until convergence is reached.

Usually, the solution found does not exactly correspond to the global optimum, however at cuts, the segment boundaries are positioned reliably between shots. This property makes the solution sufficient for later steps of the algorithm. From each segment i, the frame at the middle of the segment (at position $m_i = (p_i + p_{i+1})/2$) is taken as the representative frame of that segment and as a later key-frame candidate. Further processing steps operate only on the feature vectors $K = \{k_i = f_{m_i}\}$, leading to a significant reduction of computation time compared to clustering algorithms using all input frames.

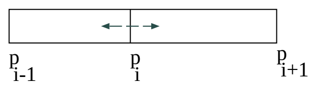


Fig. 2. Local optimization step. The boundary between two adjacent segments is moved to a position such that the homogeneity of the segments on both sides is maximized.

2.3. Clustering

Our clustering algorithm is based on the approach described in [2, 3], which is outlined in the following. Differing from the algorithm described in the literature, we are using the EMD distance as clustering criterion which results in perceptually more reasonable clusters. Moreover, we observed that an arbitrary initialization of cluster centers sometimes results in bad convergence. Hence, we perform a gradual increase of the number of cluster centers and initiate new centers into areas where new clusters are most likely to be found.

The basic principle is to find a predetermined number of clusters (corresponding to the number of key-frames) such that the dissimilarity of the frames in each cluster is minimized. To define this more formally, let c_i be the set of M cluster centers and let $K = \{k_i\}$ be the set of key-frame candidates extracted in the last step of the algorithm. For each cluster center, we define its neighbourhood N_{c_i} as:

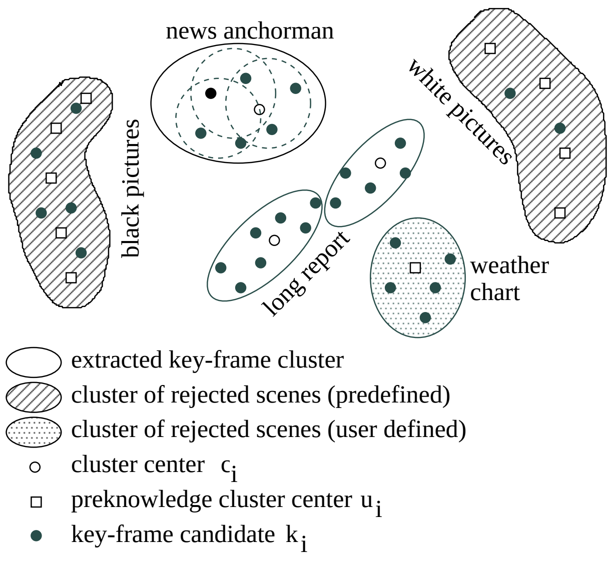
$$N_{c_i} = \left\{ k \in K \;\middle|\; \forall j : \; ||k; c_i||_{EMD} \leq ||k; c_j||_{EMD} \right\},$$

meaning that each feature vector is assigned to the neighbourhood of the nearest cluster-center (according to the EMD-distance). We say that a set of cluster centers is optimal iff they fulfill

$$min_{c_0,\ldots,c_{M-1}} \underbrace{\sum_{i\in[0;M)} \underbrace{\sum_{s\in N_{c_i}} ||s\;;\;c_i||_{EMD}}_{\text{dissimilarity of cluster } i}}_{\text{summed over all clusters}}.$$

The clustering is carried out by a k-means algorithm without the time-consecutiveness constraint used in the last step. Our experiments have shown that the k-means algorithm works best when the initial cluster centers are not chosen randomly, but rather added one at a time. Each new cluster center is initialized to the k_i with the largest distance to any existing cluster center. The overall clustering algorithm can be summarized as:

1. Set c_1 to a random k_i (e.g. k_1), set $n = 1$;
2. determine the neighbourhood N_{c_i} for all clusters;
3. reassign c_i to $c_i := \frac{1}{|N_{c_i}|} \sum_{k\in N_{c_i}} k$;
4. continue at step 2 until convergence is reached;
5. if $n = M$ stop the algorithm, else set $n := n + 1$, set $c_n = \text{maxarg}_{k_i} \min_{c_j} ||k_i; c_j||_{EMD}$ and continue at step 2.



news anchorman
white pictures
black pictures
long report
weather chart
extracted key-frame cluster
cluster of rejected scenes (predefined)
cluster of rejected scenes (user defined)
cluster center c_i
preknowledge cluster center u_i
key-frame candidate k_i

Fig. 3. Schematic example of the clustering process with integrated domain-knowledge. The lengthy report scene is divided into two separate clusters while the repeated anchorman scenes are combined into a single cluster. Black and white frames are removed by the predefined clustering centers. User defined domain-knowledge has been applied by adding a cluster-center to remove the weather chart scene from the summary.

For each cluster obtained from the last step, the k_i which is nearest to the cluster center is selected as a key-frame. The selected k_i are sorted to the correct temporal order, and the input frames corresponding to the k_i are composed to the final summary.

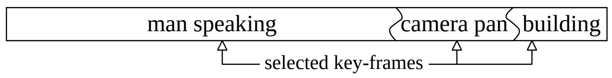
2.4. Integration of Domain-Knowledge

In this section, we modify the clustering step from the last section to insert domain-knowledge about irrelevant video scenes. To prevent these scenes from occuring in the summary, we compute the feature vectors for all scenes to be excluded. These feature vectors can also be provided by the user if he detects an image in the summary that he wants to exclude. After feeding this information back into the algorithm, a new summary can be computed without the undesired scenes. After a period of interactivity with the user, the classes of scenes to be excluded are known to the system, and summaries will only contain the desired scenes.

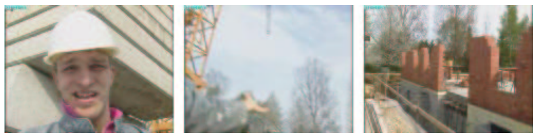
Exclusion of the scenes is accomplished by introducing the feature vectors u_i of uninteresting scenes as additional cluster centers (see Figure 3). They are treated the same as the c_i with the exception that the position of u_i is fixed and that no key-frames will be generated for their clusters. The consequence in the clustering process is that the u_i centers grab the feature vectors of scenes near the u_i vectors. These vectors will have no influence on the clustering because they are contained in the neighbourhood of a vector u_i. The total number of generated key-frames will remain the same.

3. EVALUATION

We demonstrate the behaviour of our algorithm with two test sequences. The first is the well-known *Foreman* sequence. This sequence is 400 frames long and contains no cuts. Its plot is depicted in Figure 4a. After showing the speaking man for over half of the sequence, the camera pans to the right and shows a building. Note that even though there are no cuts, our algorithm finds the three most important parts in the video. Algorithms that are based on cut detection fail on this sequence.



(a) test sequence plot



(b) summary (three key-frames)

Fig. 4. Summary of the Foreman sequence.

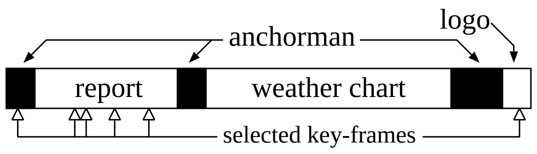
The second video contains the last two minutes of a news broadcast. Again, the video plot is shown in Figure 5a. First, we started our algorithm without any domain-knowledge (Figure 5b). Note that the news anchorman only appears once in the summary even though he appears three times in the input sequence. Since the weather chart contains very different image content, the summary contains three pictures of it but only one picture of the preceding report. Let us now suppose that we are not interested in the weather chart. So we provided some pictures of the chart as domain-knowledge in a succeeding experiment and restarted the algorithm (Fig. 5c). All key-frames of the weather-chart were removed, and additionally, more meaningful key-frames of the news report were generated.

4. CONCLUSIONS

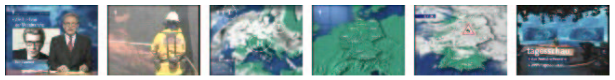
We have described a new algorithm for automatic generation of video summaries. User domain-knowledge about the video-content can be provided to improve the quality of the generated summary. We consider the approach of a modified clustering step superior to specialized filters for excluding undesired frames because our generic approach can be adapted to new application areas by simple user interaction. A topic of further research may be to integrate an algorithm for automatic detection of irrelevant feature-vectors.

Finally, the fact that our algorithm does not depend on an accurate cut detection algorithm (known to have difficulties with soft cuts) increases robustness and enables the summarization of video material without scene changes.

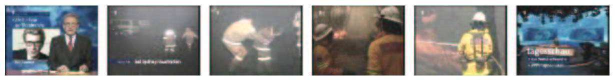
Our algorithm has been integrated into the video-database of the L^3 project (lifelong learning) for learning-videos and the ECHO project (European CHronicles On-line) to generate abstracts for four major national audio-visual archives (Italy, France, the Netherlands, Switzerland).



(a) test sequence plot



(b) without domain-knowledge: 1× anchorman, 1× report, 3× weather chart, 1× logo



(c) with knowledge to ignore weather chart: 1× anchorman, 4× report, 1× logo

Fig. 5. Summary of the last two minutes of a news broadcast.

5. REFERENCES

[1] Amit Chakraborty, "Video structuring for multimedia applications," in *SPIE Proc. of Visual Communication and Image Processing*, 2000, pp. 496–507.

[2] Chong-Wah Ngo, Ting-Chuen Pong, and Hong-Jiang Zhang, "On clustering and retrieval of video shots," in *ACM Multimedia*, 2001, pp. 51–60.

[3] Nikolaos D. Doulamis, Anastasios D. Doulamis, Yannis S. Avrithis, and Stefanos D. Kollias, "Video content representation using optimal extraction of frames and scenes," in *Proceedings of ICIP 98*, 1998, pp. 875–879.

[4] Yossi Rubner, *Perceptual Metrics for Image Database Navigation*, Ph.D. thesis, Stanford University, 1999.

[5] Krishna Ratakonda, M. Ibrahim Sezan, and Regis Crinon, "Hierarchical video summarization," in *SPIE Proc. Visual Communications and Image Processing*, 1999, pp. 1531–1541.

[6] Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg, "Video abstracting," in *Communications of the ACM*, 1997, vol. 40, pp. 55–62.

[7] Michael A. Smith and Takeo Kanade, "Video skimming and characterization through the combination of image and language understanding techniques," in *CVPR*, 1997, pp. 775–781.

[8] Yihong Gong and Xin Liu, "Video summarization using singular value decomposition," in *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2000.