Rapid Summarisation and Browsing of Video Sequences

Jaco Vermaak Patrick Pérez Michel Gangnet
Andrew Blake
Microsoft Research Cambridge, Cambridge, UK

Abstract

This paper presents a strategy for rapid summarisation and browsing of video sequences. The input video is first transformed into a sequence of representative feature vectors. Using this representation a utility function is designed that assigns high reward to subsequences of keyframes that are maximally distinct and individually carry the most information. For a specified level of detail and endpoints the keyframe sequence that maximises this utility function can be obtained by a non-iterative Dynamic Programming procedure, thus allowing the user to efficiently zoom in on any part or all of the video sequence. For the sake of compactness and clarity the working of the algorithm is illustrated on a television commercial.

1 Introduction

The proliferation of video material, both home and produced, has created a pressing need to develop tools to rapidly and efficiently summarise and browse these sequences. This has traditionally been done by segmenting the video into shots, and to extract within each shot a number of representative keyframes. The sequence of keyframes then serves as a summary for the video sequence. The most serious shortcoming of this approach is that it presents the user with a fixed level of detail. Often more (or less) information may be required for a user to get an adequate overview of a particular video sequence.

The main aim of this paper is to develop a browsing strategy where the level of detail is determined by the user. The proposed method does not require explicit shot detection, and allows the level of detail to be smoothly varied between a small number of keyframes and the entire video sequence. As opposed to clustering approaches it is non-iterative and thus computationally more efficient.

The method represents each frame in the video sequence by a low-dimensional feature vector. Based on this feature representation two functions are defined. The first, a frame utility function, evaluates the goodness of a frame as a keyframe. Good keyframes should be well illuminated and contain lots of information. The second, a frame distance function, computes the similarity between any two frames in the video sequence. Keyframes should be maximally dissimilar. Using these two components a utility function is constructed that measures the goodness of any keyframe sequence. The form of this function is such that the optimal keyframe sequence can easily be obtained by Dynamic Programming. Thus the approach is non-iterative. It can accommodate flexible endpoints, allowing the user to zoom in on any part or all of the video sequence. Any

BMVC 2002 doi:10.5244/C.16.40

number of keyframes can be specified between the endpoints, thus allowing a flexible level of detail.

There have been several previous efforts at generating automatic video summaries at different levels of detail. In [7, 8] a feature representation is used for the video frames, and video summaries are obtained by iterative hierarchical clustering of the frames in feature space under spatial and temporal constraints. In [2] the trajectory of the video in feature space is recursively simplified using a curve splitting algorithm, and represented as a tree structure. Frames at junctions between curve segments at different levels of the tree are then used as keyframes to summarise the video at different levels of detail.

The remainder of the paper is organised as follows. Section 2 presents the feature representation of the video frames. Sections 3 and 4 develop the frame utility and frame distance functions, respectively. The Dynamic Programming algorithm to compute the optimal keyframe sequence is discussed in Section 5. Despite its non-iterative nature this algorithm can still be computationally expensive for very long video sequences. An efficient strategy to deal with such sequences is outlined in Section 6. Some experiments and results are reported in Section 7 before concluding with a summary in Section 8.

2 Video Frame Features

Instead of working directly with the raw images a low-dimensional feature representation is used. The features, once computed, are used in all subsequent processing, thus massively reducing the computational complexity. If the features are chosen carefully the pertinent information for keyframe extraction can be isolated and the redundant information removed.

In the approach here the RGB input images are first converted to a chrominance representation to achieve some degree of invariance to changes in illumination. In the spirit of [3] the chrominance components for each pixel are computed according to $(C_1, C_2) = (R, G)/\sqrt{R^2 + G^2 + B^2}$. The chrominance images are partitioned into four regions by halving them horizontally and vertically. In each region a twenty bin normalised histogram is computed independently for each of the chrominance components. For region r, $r = 1 \cdots 4$, in frame t these are denoted by $\mathbf{h}_{r,t}^{C_1}$ and $\mathbf{h}_{r,t}^{C_2}$, respectively. The full feature vector for frame t is constructed as

$$\mathbf{f}_t = \left(t, \mathbf{h}_{1,t}^{C_1}, \mathbf{h}_{1,t}^{C_2} \cdots \mathbf{h}_{4,t}^{C_1}, \mathbf{h}_{4,t}^{C_2}, \overline{I}_t\right), \tag{1}$$

where $\overline{I}_t \in [0, 1]$ is the average luminance of all the pixels in frame t. Note that the frame index is also included in the feature vector. The region based histogramming achieves invariance to small camera movements while preserving coarse structural information.

3 Frame Utility Function

A frame utility function is required to measure the goodness of any frame in the video sequence as a keyframe. This notion of goodness has a large subjective component, which depends on the nature of the video material and the particular user, and may even change over the duration of a video sequence. However, there are some characteristics common to all good keyframes, and it is these that a general purpose utility function should exploit.

Typically good keyframes should carry the most information and be well illuminated. The utility function used here is designed to capture these simple notions. A crude measure for the amount of information is the entropy of the colour distribution in the image. The higher the entropy, the larger the colour variation, which, in general, corresponds to a higher level of detail. For a B bin normalised histogram $\mathbf{h} = (h_1 \cdots h_B)$ the entropy is defined as

$$H(\mathbf{h}) = -\sum_{b=1}^{B} h_b \log h_b \in [0, \log B].$$

Using the feature representation in (1) and the definition for histogram entropy above the frame utility function is constructed as[1]

$$u(\mathbf{f}) = \begin{cases} u_{\min} & \text{if } \overline{I} < \overline{I}_{th} \\ \eta\left(\frac{1}{8}\sum_{r=1}^{4}(H(\mathbf{h}_r^{C_1}) + H(\mathbf{h}_r^{C_2}))\right) & \text{otherwise,} \end{cases} \tag{2}$$

where $u_{\min} = 0.001$ is the minimum value for the frame utility function, $\overline{I}_{th} = 0.2$ is the threshold above which frames are deemed to be well illuminated, η is a hyperbolic squashing function that limits the utility to lie within $[u_{\min}, 1]$, and $(H(\mathbf{h}_r^{C_1}), H(\mathbf{h}_r^{C_2}))$ are the entropy values for the normalised chrominance histograms in region r. The utility function assigns a low value to frames that are poorly illuminated. For frames that exceed the illumination threshold the utility is proportional to the average entropy over all the regions. This crude and simple utility function was found to work well in practice.

4 Frame Distance Function

For summarisation and browsing purposes the chosen keyframes should be maximally dissimilar. It is thus necessary to define a function to measure the distance between any two frames in a video sequence. Using again the feature representation in (1) an image based distance is defined as

$$d^I(\mathbf{f}_s, \mathbf{f}_t) = \frac{1}{8}\sum_{r=1}^{4}\left(B(\mathbf{h}_{r,s}^{C_1}, \mathbf{h}_{r,t}^{C_1}) + B(\mathbf{h}_{r,s}^{C_2}, \mathbf{h}_{r,t}^{C_2})\right),$$

where $B(\mathbf{h}_s, \mathbf{h}_t)$ is the Bhattacharyya distance between the histograms $\mathbf{h}_s$ and $\mathbf{h}_t$, defined as

$$B(\mathbf{h}_s, \mathbf{h}_t) = \left[1 - \sum_{b=1}^{B}\sqrt{h_{b,s}h_{b,t}}\right]^{1/2} \in [0, 1].$$

However, the image based distance is not enough. Frames during fading transitions in produced video can be highly dissimilar, resulting in a disproportionally large number of keyframes in and around the region of the transition. To eliminate this effect it is necessary to penalise keyframes that are temporally close. This can be achieved by defining a time based distance function of the form

$$d^T(\mathbf{f}_s, \mathbf{f}_t) = 1 - \exp(-\alpha|s - t|),$$

where $\alpha = \log 2/T_h$, with $T_h = 0.25$ the exponential decay constant. Finally, the combined frame distance function is defined as the product of the image and time based distance functions, yielding

$$d(\mathbf{f}_s, \mathbf{f}_t) = d^I(\mathbf{f}_s, \mathbf{f}_t)d^T(\mathbf{f}_s, \mathbf{f}_t). \tag{3}$$

[1] For notational clarity the frame index subscript is suppressed in what follows.

5 Keyframe Computation

Given a video sequence of N frames the objective is to find the K best representative keyframes to summarise the video. These frames should be maximally distinct and individually carry the most information. The approach here is to construct a utility function that captures these criteria, and then to maximise this function to yield the optimal keyframe sequence. Such a utility function is not unique, and should be constructed to be amenable to straightforward optimisation, where possible. One particularly popular form is given by

$$C(s_{1:K}) = u(\mathbf{f}_{s_1}) \prod_{k=2}^{K} \widetilde{d}(\mathbf{f}_{s_{k-1}}, \mathbf{f}_{s_k}) u(\mathbf{f}_{s_k}),$$

where $s_{1:K} = \{(s_1 \cdots s_K) : \ s_k \in \{1 \cdots N\}, \ k = 1 \cdots K; \ s_k < s_{k+1}, \ k = 1 \cdots K - 1\}$ denotes the candidate keyframe sequence, and

$$\widetilde{d}(\mathbf{f}_s, \mathbf{f}_t) = \begin{cases} d(\mathbf{f}_s, \mathbf{f}_t) & \text{if } s < t \\ 0 & \text{otherwise} \end{cases}$$

has been constructed to honour the constraint that the sequence of keyframes should be strictly increasing. To prevent numerical underflow problems the alternative logarithmic representation is used here, *i.e.*

$$\log C(s_{1:K}) = \log u(\mathbf{f}_{s_1}) + \sum_{k=2}^{K} \left[\log \widetilde{d}(\mathbf{f}_{s_{k-1}}, \mathbf{f}_{s_k}) + \log u(\mathbf{f}_{s_k}) \right]. \tag{4}$$

The optimal keyframe sequence is the one that maximises this utility function, *i.e.*

$$s^*_{1:K} = \arg\max_{s_{1:K}} \log C(s_{1:K}).$$

The keyframes in this sequence will have high individual utility and be maximally distinct.

The form of the utility function in (4) facilitates straightforward maximisation using Dynamic Programming. The algorithm is summarised below.

Algorithm 1: Dynamic Programming for Keyframe Computation

Initialisation

- Inputs: number of keyframes K, frame feature sequence $\mathbf{f}_{1:N}$, (optional) endpoints (s^*_1, s^*_K).
- For fixed s^*_1, set $D_{1,s^*_1} = 0$, $D_{1,t} = -\infty$, $t = 1 \cdots N$, $t \neq s^*_1$.
  For flexible s^*_1, set $D_{1,t} = \log u(\mathbf{f}_t)$, $t = 1 \cdots N$.

Forward Iteration

- For $k = 2 \cdots K$, compute

$$D_{k,t} = \max_{s \in \{1 \cdots N\}} \{D_{k-1,s} + \log \widetilde{d}(\mathbf{f}_s, \mathbf{f}_t)\} + \log u(\mathbf{f}_t), \qquad t = 1 \cdots N$$

$$\Phi_{k-1,t} = \arg\max_{s \in \{1 \cdots N\}} \{D_{k-1,s} + \log \widetilde{d}(\mathbf{f}_s, \mathbf{f}_t)\} + \log u(\mathbf{f}_t), \quad t = 1 \cdots N.$$

Traceback

- For fixed s_K^*, set s_K^* to the desired end frame.
  For flexible s_K^*, set $s_K^* = \arg\max_{t \in \{1 \cdots N\}} D_{K,t}$.
- For $k = K-1 \cdots 1$, set $s_k^* = \Phi_{k, s_{k+1}^*}$.

The algorithm requires the individual frame utilities for all the frames in the video sequence, as well as the frame distances between all the possible frame pairs. These computations are $O(N)$ and $O(N^2)$, respectively, but are based on the low-dimensional feature representation, and only need to be performed once. The algorithm itself is non-iterative and of $O(NK)$ complexity. Further computational savings are possible due to the constraint that the sequence of keyframe indices is strictly increasing. The algorithm supports both flexible (automatically selected) and user-specified endpoints, and an arbitrary number of keyframes can be specified, thus allowing the user to zoom in on any portion (or all) of the video sequence to any level of detail.

6 Processing Long Sequences

Despite the non-iterative nature of the algorithm very long video sequences can still incur high computational costs, both in the preprocessing stage and during the computation of the keyframes. A simple approach to reduce these costs is to segment the video into its constituting shots, and apply the algorithm to each shot individually. For digital home video shot detection is trivial, since the time stamps are encoded in the frames. Many algorithms exist for shot detection in produced video, *e.g.* [1, 5]. Here a robust detection algorithm is designed using the frame distance function defined earlier. This algorithm is outlined in Appendix A. Note, however, that the shot detection algorithm is not required to be 100% accurate, since the browsing algorithm itself is robust to transitions.

Denote by N the total number of frames in the video sequence, K the total number of keyframes required, S the number of shots, and N_i, $i = 1 \cdots S$, the number of frames in shot i. The main problem to be resolved is the allocation of the number of keyframes per shot, K_i, $i = 1 \cdots S$, for a given level of detail $K = \sum_{i=1}^{S} K_i$. A naive approach would be to set the number of keyframes proportional to the shot duration, *i.e.* $K_i \propto N_i$, but this takes no account of the possible variation in the shot content. In the approach here the shot content is summarised by computing the absolute minimum number of keyframes required to represent a shot, denoted by K_i^*, in an optimal sense. The number of keyframes for a given level of detail is then set to

$$K_i = \left[K_i^* K / \sum_{j=1}^{S} K_j^* \right], \tag{5}$$

where $[\cdot]$ denotes the rounding operation. This strategy implicitly takes account of both the shot content and duration.

The next section outlines the optimal strategy to compute the minimum number of keyframes per shot, whereas the following summarises the browsing strategy.

Computing the Minimum Number of Keyframes per Shot

Computing the minimum number of keyframes per shot can be viewed as a model order selection problem, which can be solved by a suitable definition of a likelihood function and using this within Bayes factors. For a large number of data points an asymptotic approximation to the Bayes factor is given by the Bayes Information Criterion (BIC) [6]. For the problem considered here this criterion is given by[2]

$$K^* = \arg\min_K \{-2 \log L(\mathbf{f}_{1:N} | \widehat{s}^{ML}_{1:K}) + K \log N\},$$

where L is the likelihood function, to be defined, and $\widehat{s}^{ML}_{1:K}$ is the keyframe sequence of length K that maximises this likelihood. To define the likelihood each keyframe is considered as a cluster centre, with the clusters comprised of the frames temporally closest to the cluster centres, yielding

$$L(\mathbf{f}_{1:N} | s_{1:K}) = \prod_{k=1}^{K} \prod_{t \in \mathcal{I}_k} p(\mathbf{f}_t | s_k),$$

with $\mathcal{I}_k = \{t : |s_k - t| < |s_l - t|,\ l = 1 \cdots K,\ l \neq k\}$. Note that all the data points (frame features) are assumed to be independent. The individual frame likelihoods are defined as

$$p(\mathbf{f}_t | s_k) \propto \exp\left[-\frac{1}{2\sigma^2} d^2(\mathbf{f}_{s_k}, \mathbf{f}_t)\right],$$

thus decreasing with an increase in the distance of the frame from the cluster centre. The parameter σ is set to reflect the width of the clusters, and a value of $\sigma = 0.25$ is used here. Using this definition for the likelihood the BIC solution becomes

$$K^* = \arg\min_K \left\{ \frac{1}{\sigma^2} \sum_{k=1}^{K} \sum_{t \in \mathcal{I}_k} d^2(\mathbf{f}_{\widehat{s}^{ML}_k}, \mathbf{f}_t) + K \log N \right\}. \tag{6}$$

The first term generally decreases with an increase in the number of keyframes, whereas the second increases. The optimisation of (6) is done exhaustively over the range $K \in \{1 \cdots 0.1N\}$. This procdure requires the computation of the maximum likelihood keyframe sequence $\widehat{s}^{ML}_{1:K}$ for every possible value of K. This can be obtained using an iterative clustering algorithm with a suitably chosen cost function, but would incur substantial computational costs. Here the sequence resulting from the Dynamic Programming algorithm in Section 5 is used as an approximation to the maximum likelihood keyframe sequence. As discussed before these computations can be performed efficiently, and the approximation is adequate, since the Dynamic Programming algorithm yields maximally distinct keyframes, which is the implicit requirement for cluster centres.

Browsing Strategy Summary

The complete browsing algorithm is summarised below.

[2] For notational clarity the shot index is suppressed in what follows.

Algorithm 2: Video Browsing

Preprocessing

- Segment the video sequence into shots using the algorithm in Appendix A.
- For each shot, compute
  - the individual frame utilities, using (2),
  - the frame distances for all possible frame pairs, using (3), and
  - the optimal number of keyframes, using (6) and Algorithm 1 to approximate the maximum likelihood keyframe sequence.

Browsing

- The user specifies the endpoints (possibly unconstrained) and the desired level of detail K.
- Compute the number of keyframes per shot according to (5).
- For each shot compute the required number of keyframes for browsing using Algorithm 1.

■

7 Experimentes and Results

This section demonstrates the working of the proposed browsing algorithm. For the sake of compactness and clarity a television commercial is chosen for this purpose. Commercials are almost a feature length film in half a minute, being rich in transitions and variation in shot types.

The particular commerical chosen here is summarised in Figure 1. It is comprised of 20 shots, separated by 18 cuts and one fade. The summary was generated using the browsing algorithm at the coarsest level of representation, *i.e.* only the keyframes suggested by the BIC criterion are retained ($K_i = K_i^*$). This concise summary is useful for indexing and retrieval. Note that the shot detection algorithm successfully located all the transitions. Compare the 24 keyframes obtained in this manner with 24 keyframes evenly spaced over the video sequence, as shown in Figure 2. A lot of detail is lost in parts of the video where there is much activity, whereas prolonged parts with little activity are over-represented.

Figure 3 shows some example results at a higher level of detail where the original video sequence was reduced to 5% of its original length. The 40 resulting keyframes essentially provide a fast-forwarded version of the original video, and capture all the salient features to facilitate rapid browsing to locate areas of interest.

Figure 1: **BIC keyframes generated by the browsing algorithm**. The video sequence is successfully segmented into its constituting shots, shown by the red lines. The keyframes shown are those suggested by the BIC criterion, and provides a concise summary of the video sequence, useful for indexing and retrieval.

Figure 2: **Evenly spaced keyframes**. With evenly spaced keyframes several of the shorter shots are missed, and long shots are over-represented.

Figure 3: **Rapid browsing**. The 40 keyframes generated by reducing the video sequence to 5% of its original length capture all the salient features, and facilitate rapid browsing to locate areas of interest.

8 Conclusions

In this paper an algorithm was developed to facilitate rapid and efficient summarisation and browsing of video sequences. The algorithm is non-iterative and, as illustrated in

Section 7, allows the user to zoom in on any part or all of the video sequence to any level of detail. It uses a feature representation for the video frames and obtains the summarising sequence by using Dynamic Programming to maximise a utility function that assigns high reward to subsequences of keyframes that are maximally distinct and individually carry the most information.

To save computational costs long video sequences are first segmented into shots, after which the algorithm is applied to each shot individually. The number of keyframes for each shot is allocated in proportion to the minimum number of keyframes required to summarise the shot, as determined by the BIC criterion.

Future work will focus on extending the frame utility function to give the user greater flexibility in specifying the desired characteristics of the keyframes chosen for summarisation, *e.g.* degree and type of motion, objects and persons of interest, *etc*.

A Shot Detection

Most traditional methods for shot detection compute some image based inter-frame distance for consecutive frame pairs, and then label shot boundaries as those locations for which this distance exceeds some predefined threshold. This general approach suffers from two drawbacks. First, using only pairs of frames may lead to spurious peaks in the distance computation. Second, a single threshold is often insufficient for detection, especially with more gradual transitions such as fades and wipes. The approach here attempts to address these shortcomings.

For a video sequence of N frames the sequence[3]

$$x_t = \frac{1}{2W}\sum_{s=1}^{W}\left[d^I(\mathbf{f}_t, \mathbf{f}_{t+s}) + d^I(\mathbf{f}_t, \mathbf{f}_{t-s})\right], \quad t = 1 \cdots N, \tag{7}$$

is computed first. Instead of using only frame pairs the inter-frame distance is averaged over a sliding window of size $2W+1$ (typically $W = 2$), thus reducing the effect of spurious peaks. The outliers of the resulting sequence signify the shot boundaries. Instead of using a single threshold, these are detected using a robust statistical procedure [4], outlined below.

Broadly speaking the approach adopted here robustly fits the mean of a Gaussian distribution with known variance to the sequence in (7). The outliers under this model are then deemed candidates for the shot boundaries. Within the context of robust statistics an estimate of the Gaussian mean can be obtained as

$$\mu^* = \arg\min_{\mu} \sum_{t=1}^{N} \phi\left(r(x_t;\mu)\right),$$

where the residual and squashing function are defined as $r(x_t;\mu) = (x_t - \mu)^2$ and $\phi(u) = 1 - \exp(-u/2\sigma^2)$, respectively. Since ϕ is concave and increasing the optimisation problem can be augmented and reformulated as

$$(\mu^*, w^*_{1:N}) = \arg\min_{(\mu, w_{1:N})} \sum_{t=1}^{N} \left[w_t r(x_t;\mu) + \psi(w_t)\right],$$

[3]The summation is suitably truncated and scaled at the endpoints.

where $\psi(w) = \phi \circ \phi^{'-1}(w) - w\phi^{'-1}(w)$. In the above w_t is a weight indicating to which degree x_t contributes towards the estimation of mean of the Gaussian distribution. The desired estimates are easily obtained by the Iterative Reweighted Least Squares (IRLS) procedure [4] that alternatively optimises for the mean and the weights until convergence is achieved. Specifically, conditional on the weights the estimate of the mean becomes

$$\mu^* = \sum_{t=1}^{N} w_t^* x_t / \sum_{t=1}^{N} w_t^*. \tag{8}$$

Thus each sample contributes to the estimate of the mean in proportion to its weight. Conditional on the mean a new estimate for the weights is given by

$$w_t^* = \phi^{'}\left(r(x_t;\mu)\right) \propto \exp\left[-\frac{1}{2\sigma^2}(x_t - \mu^*)^2\right], \quad t = 1 \cdots N. \tag{9}$$

Thus the weight w_t^* is proportional to the probability of the sample x_t under the Gaussian distribution with mean μ^* and variance σ^2. A typical value for the variance used here is $\sigma^2 = 2.5 \times 10^{-3}$. The weights are initialised uniformly, and the two steps in (8) and (9) are iterated until the estimates converge.

After the estimation procedure samples with small weights represent outliers under the Gaussian model, and hence candidates for the shot boundaries. In the final step the shot boundaries are labelled as the peak locations of the sequence $\widetilde{x}_t = x_t(1 - u(\widetilde{w}_t^* - w_{TH}))$, $t = 1 \cdots N$, where $\widetilde{w}_t^* = (\sum_{s=1}^{N} w_s^*)^{-1} w_t^*$ is the normalised weight, u is the unit step function, and w_{TH} is a threshold, typically set to $w_{TH} = 0.1/N$.

References

[1] J.S. Boreczky and L.A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval for Image and Video Databases IV, Proc. SPIE Vol. 2670*, pages 170–179, 1996.

[2] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *ACM Multimedia 98*, pages 211–218, 1998.

[3] M.S. Drew and J. Au. Video keyframe production by efficient clustering of compressed chromaticity signatures. In *ACM Multimedia 2000*, pages 365–368, 2000.

[4] P. J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

[5] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *Image and Video Processing VII, Proc. SPIE Vol. 3656*, pages 290–301, 1999.

[6] G. Schwarz. Estimating the dimension of a model. *Ann. of Statist.*, 6:461–464, 1978.

[7] X. Sun and M.S. Kankanhalli. Video summarisation using R-sequences. *J. Real Time Imaging*, 6:449–459, 2000.

[8] D. Zhong, H.J. Zhang, and S.-F. Chang. Clustering methods for video browsing and annotation. In *Storage and Retrieval for Image and Video Databases IV, Proc. SPIE Vol. 2670*, pages 239–246, 1996.