VIDEO SUMMARIZATION BY EFFICIENT CLUSTERING OF COMPRESSED CHROMATICITY SIGNATURES

Mark S. Drew and James Au
School of Computing Science, Simon Fraser University,
Vancouver, B.C. Canada V5A 1S6
{mark,ksau}@cs.sfu.ca

ABSTRACT

Motivated by problems in video summarization caused by change in lighting, we develop a new low-dimensional feature video frame feature that is more insensitive to such changes than previous methods. Because we effectively reduce any video to the same lighting conditions, we can produce a *universal basis* on which to project video frame features. We set out a new multi-stage hierarchical clustering method that merges clusters based on variance ratio, the ratio of intra- and total variance, and merges only adjacent clusters. A second stage merges clusters incorrectly split in the first round by the greedy hierarchical algorithm, and finally merges non-adacent clusters to fuse near-repeat shots. The new summarization method produces a very succinct set of keyframes for videos, and results are very encouraging.

1. INTRODUCTION

Keyframes remain the most natural and immediate summarization expressions for video. While keyframes are established, "inverse Hollywood" techniques, such as graph-based pruning [1] or other hierarchical representations of video sequences [2, 3, 4] can be brought to bear.

Ferman and Tekalp [5] applied a clustering approach to intra-shot keyframe selection, based on average color histogram [3] and intersection histogram [6]. The average histogram is indeed simply the average of all histograms in the shot. The intersection histogram is a generalization of the intersection of two image histograms, a rephrasing of histogram difference in an L_1 norm [7]. Other algorithms advanced have used comparison to the first frame in a shot using color and motion [8], accumulating frame differences [9], algorithms based on geometric metaphors, e.g. [10], and other means but here we mean to set out a more effective algorithm that is also simpler.

Ferman and Tekalp [5] extract keyframes by clustering, and then go on to prune as well as augment the number of frames identified as keyframes, using a number of criteria. Firstly, since lighting changes can bias the average histogram adversely, outliers with respect to luminance are added to the list of keyframes. Secondly, clusters that contain only a few frames are eliminated. Thirdly, clusters with centers closer in color distance than a threshold T_C are merged. Finally, within each cluster, if the cluster radius exceeds T_C two representative frames are chosen that are (1) closest to the center, and (2) farthest from the center. The authors use the Y-Cb-Cr color space.

We have found that applying some of the techniques above yield poor results for keyframe extraction in quite common situations. E.g., we considered a sequence taken from the movie *Ant*, (see http://www.cs.sfu.ca/∼mark/ftp/Ismip00/) and found that the moving background and subtle lighting changes seen there confound the algorithm in [5] because of the change in pixel values under different synthetic lighting. In the sequence shown in Fig. 1 the method in [5] applied to real outdoor lighting change from shadowing produces many keyframes, when in fact we wish to see only one.

In this paper we set out a method that performs correctly under this common lighting difficulty and moreover performs better overall on any video. Here, we utilize knowledge about color formation for lighting and surfaces, from physics-based vision, to cast the clustering problem into the form of projection of frames onto a basis derived offline. This basis is precomputed from a large training set and then applies equally to all subsequent videos.

Lighting is first discounted by normalization of color-channel bands [11]. This step approximately but effectively removes dependence on both luminance and lighting *color*. Then image frames are moved into a chromaticity color space. As well as reducing the dimensionality of color to 2, since chromaticity involves a ratio of color bands this also has the effect of removing shading. In order to make the method fairly robust to camera and object motion, and displacements, rotations, and scaling, we go over to a 2D histogram derived from DC components of frames. In this paper we use 128×128 histograms. Chromaticity histograms are then *compressed* — i.e., we treat the histograms as *images* (see [12, 11]). Here, we use a wavelet-based compression because this tends to strike a balance between simple low-pass filtering and retaining some important details. Using a 3-level wavelet compression we arrive at 16×16 histograms.

However, we found that compression of histograms could be improved if the histograms are first *binarized*, i.e., entries are replaced with 1 or 0. The rationale for this step is that chromaticity histograms are a kind of color signature for an image, similar to a palette. In work involving recovering the plausible illuminant from pixel values in an image [13] it was found beneficial to utilize this kind of color signature. Here, the step of binarizing the histogram not only reduces the computational burden, since true chromaticity histograms need not be computed, but also has the effect of producing far fewer negatives in the compressed histogram. Finally, we found that one further step could substantially improve the energy compaction of the representation: we carry out a 16×16 Discrete Cosine Transform (DCT) on the compressed 16×16 histogram. After zigzag ordering, we keep 21 DCT coefficients.

The main utility of color-channel normalization and then chromaticity space is that every image approximately *goes over into the same lighting conditions*. That is, we perform a kind of universal normalization of images. In fact, it was shown that going further in this direction and *iterating* the steps (1) color-channel normalization, and (2) forming the chromaticity, which is itself a kind of color normalization, produced excellent results for image indexing [14]. There, it was determined that about five iterations produced convergence, using such an iterated "comprehensive" normalization. Thus, here, we are going about one-fifth of the way towards the optimum, but still with good results and far less computation.

Since every image lives in approximately the same lighting, after our normalization steps, we can in fact precompute a *basis* for the DCT 21-vectors, offline, that can then be reused for any new image or video. Here we find a basis set by the Singular Value Decomposition (SVD) of the DCT 21-vectors (cf. [15], in which a basis set for DCT-transformed log color-ratios was found using an SVD and [16] which used SVD on motion vectors and was considerably more complex than the method presented here). We found here that 12 components in the new basis represent the entire DCT vector very well: 12 eigenvectors account for 99.6% of the variance. We found that energy compaction worked better using a spherical chromaticity, rather than the usual linear one.

Thus the method we set out here is to precompute a set of 12 basis vectors, once and for all, and then form the 12-vector coefficients for any video frame with respect to this basis. Then keyframe extraction by clustering can be carried out very efficiently, using only 12-component vectors, and need not be performed only intra-shot, as in [5]. In fact, we can present an entire video sequence to a clustering method, or just a small sequence, and still find keyframes. Abrupt cuts certainly form boundaries between shots, but gradual transitions can also be simply identified as regions with a fast derivative of keyframe number – if many keyframes are produced over a short time period then, while this *could* be due to very fast scene changes, the most likely explanation is that a gradual transition has taken place. Thus, rather than keeping these transitions, we discard them.

Each keyframe is associated with a length-12 feature vector of basis coefficients. Thus video search, retrieval, and browsing are a fast and simple matter once keyframe production has occurred.

The clustering method we adopt is a variant of hierarchical clustering [17], adapted to video. Firstly, only adjacent frames are merged, as we wish to maintain the temporal order of keyframes in a video summary (as opposed to discarding temporal order, as in [18]). This permits an efficient data structure for the clustering tree and distance matrix. Also, an initial clustering is followed by an additional step on clusters found, marking some keyframes as representing near-repeated shots.

This efficiently provides a further pruning step that has the effect of merging similar shots. The result is a set of frames that provide both an intra- and inter-shot representation of the video. Further, the second, cluster-merging step, automatically provides the structure for a graph-based representation of the video. Details are given in §3.

The threshold we use to control the number of clusters is the *variance ratio*; this is efficient to compute and is adaptive to different videos. We can efficiently calculate variance for all subclusters (maintained in the hierarchical tree structure) and threshold on the ratio of intra-cluster variance to total variance. We find that a universal threshold of $T_{VR} = 0.25$ variance ratio performs quite well.

While boundary detection is simply quantified, testing performance for keyframe extraction involves a comparison of keyframes generated to those that *should* be generated, and for ground truth we must here rely on human opinion. With that proviso, we find that the method presented here does agree well with human notions of what frames most expressively summarize a video sequence.

2. ILLUMINATION CHANGE AND NORMALIZATION

2.1. Pixel values under illumination change

The motivation for color-channel normalization lies in making images approximately independent of illumination; however such a normalization followed by moving to 2D chromaticity space has the concomitant effect of transforming every image into the same illumination domain.

To understand how color normalization effects illuminant independence, firstly we note that several justifications can be made [19] for a *diagonal model* for illumination change: color values for pixels approximately undergo independent multiplicative changes in each color channel R, G, and B, that is independent of retinal coordinates.

For consider pixel values $\boldsymbol{\rho}^x$ at camera retinal position $\boldsymbol{x}$. Under lighting $E(\lambda)$, if the light impinges on a Lambertian surface with reflectance function $S(\lambda)$ from direction $\boldsymbol{a}$, the RGB camera value produced is

$$\boldsymbol{\rho}^x = \boldsymbol{a} \cdot \boldsymbol{n}^x \int E(\lambda) S^x(\lambda) \boldsymbol{q}(\lambda) d\lambda \qquad (1)$$

where $\boldsymbol{q}(\lambda)$ are three camera sensitivity functions, and $\boldsymbol{n}^x$ is the surface normal at position x. Now, according to Borges [20], the usual *factor model* approximation used in computer graphics can be employed:

$$\rho_k^x \simeq \boldsymbol{a} \cdot \boldsymbol{n}^x s_k^x e_k / \sigma_k \ , \ k = 1..3 \quad \text{with} \qquad (2)$$

$$s_k^x = \int S^x(\lambda) q_k(\lambda) d\lambda \, , e_k = \int E(\lambda) q_k(\lambda) d\lambda \, , \sigma_k = \int q_k(\lambda) d\lambda$$

providing the light is fairly white. Therefore under a lighting change $E \to E'$, $\boldsymbol{a} \to \boldsymbol{a}'$ pixel values transform according to

$$\rho_k^x \ \to \ \rho_k^{x\prime} = ({e_k}'/e_k) \rho_k^x \qquad (3)$$

Hence normalizing each color channel, in L_1 or L_2 norm, effectively removes the lighting. In fact, in the context of eq. (2), division of an N-pixel image by the mean amounts to seeing the image as it would appear in lighting $e_k = N\sigma_k / \sum_x \boldsymbol{a} \cdot \boldsymbol{n}^x s_k^x$. Thus under the common "gray-world" assumption, every image goes into the same lighting.

Going over to a 2D chromaticity $(r, g) = (R, G)/(R+G+B)$ removes the shading term $\boldsymbol{a} \cdot \boldsymbol{n}^x$ as well, and thus removes the effect of lighting direction. Here, for reasons discussed in §3, we actually use a spherical chromaticity $(r, g) = (R, G)/\sqrt{R^2 + G^2 + B^2}$.

2.2. Universal basis

Because we reduce all images to approximately the same color domain, we can calculate offline a basis on which to project each video frame. In [11] we used a wavelet-based compression method, applied to histograms as if they were images.

A further low-pass filter was applied by forming the DCT image of the resulting histogram, and truncating. However, that method can be improved in several substantive ways. Firstly, consider the 128×128 2D chromaticity histogram in Fig.3(b) that arises from the video frame in Fig.3(a). The original image essentially consists of only a few uniformly-colored patches and so produces a histogram comprising a few sharp spikes. As a consequence, the wavelet-based compression gives rise to some negatives, as in Fig.3(c).

A more well-behaved compression results if, instead, we replace the histogram with a *binarized* one, i.e., with entries either 0 or 1. In a sense, this shows a kind of color *signature* for the image, like a color palette. Now the wavelet-compressed histogram appears as in Fig.3(d): the percentage of negative counts in the histogram have gone from 8% in Fig.3(c) to only 1% in Fig.3(d).

As well, the DCT produces better energy compaction if we go over to a nonlinear definition of chromaticity, based on squared (or higher) powers of pixel values: $(r, g) = (R, G)/(R^p + G^p + B^p)^{1/p}$. The reason is that, since linear chromaticity obeys $r + g \leq 1$, there can exist a straight diagonal edge in a chromaticity histogram, as in Fig.4(b), which arises from the image Fig.4(a). The DCT image for this histogram shows a ringing effect. In terms of a spherical definition of chromaticity (with $p = 2$), the straight edge is replaced by a circle, in Fig.4(c), and the ringing disappears.

Figs.5(a,b) show the spherical chromaticity color signature for the image in Fig.3(a), and the resulting DCT image for the compressed histogram: the DCT is quite concentrated.

As a final step, we form the SVD basis for the first 21 DCT coefficients, in zigzag order, for a large training set of images. We found that 12 basis vectors suffice quite well for DCT images of videos we examined.

3. CLUSTERING INTRA- AND INTER-SHOTS

3.1. Clustering strategy

Fig.2 shows the clustering scheme we use. We cluster using the 12-vector features outlined above. Although these vectors are longer than those used in [5], they carry much more information about the video than do the 2-vectors used there since they are grounded in a physics-based vision approach that arises from the underlying physics of color formation.

To begin with, we compute the 12-vector features for each frame, projecting onto a pre-determined basis. Because the feature space is relatively low-dimensional, and especially because we merge only adjacent frames, frame clustering is very fast. With buffering, this means that we could effectively carry out clustering on data from a temporal window of streaming video, presenting only keyframes to the user.

Only adjacent clusters are merged. The threshold used is the Variance Ratio = (intra-cluster variance)/(total variance). As well, we apply a minimum variance threshold. The reason for this is that we found that if the entire video consists of only one scene, the ratio becomes meaningless. To fix this, we introduce a minimum variance threshold that would identify such uniform videos. Fortunately, these kinds of videos contain total variance that is usually much smaller than those of complex videos (by about a factor of 20). Thus we are able to select a global threshold that can work on most types of videos.

Finally, we carried out a second clustering step, on the clusters found. This has the effect of dropping whole shots that are near-repeats, as in the sequence ABCADEF.

3.2. Clustering algorithm

In detail, the steps used are as follows:

1. Perform bottom-up hierarchical clustering using minimum L_2 distance between cluster means. Only adjacent clusters are merged. Spherical clusters containing contiguous frames are produced.
2. Extract top-level clusters that satisfy both the variance ratio threshold (intra-cluster variance)*T_{VR} $<$ (total variance) and the minimum variance threshold. This gives us an initial set of clusters.
3. Merge adjacent initial clusters that satisfy thresholds. This step is needed to correct errors in the greedy hierarchical clustering algorithm in step 1: we merge similar adjacent clusters that were split in the early rounds.
4. Detect transitions and mark them: all contiguous clusters of size ≤ 3 are merged and marked as a transition.
5. Hierarchically merge non-adjacent non-transition clusters that result in lower variance: if cluster AB's variance is lower than either cluster A or B, then the two are merged.
6. Finally, one keyframe closest to its cluster centroid is extracted from each resulting cluster.

4. RESULTS

Table 1 shows results for the algorithm compared to the algorithm in [5], as specified there. Not only are the summarizations much more succinct, they are correctly unresponsive to changes in illumination, and also agree very well with human summarizations. In general, even given perfect transition detection, the method [5] generates too many keyframes whereas the present method does not; without perfect transition detection, [5] misses many more keyframes than the present method. Results may be viewed at
http://www.cs.sfu.ca/∼mark/ftp/Ismip00/ .

5. CONCLUSIONS

Results produced are excellent, yielding a very succinct summarization that agrees well with human expectations, with almost no redundancy and very few misses. The success of the method depends on both the new approach to color histograms and the multi-stage hierarchical clustering algorithm.

However, we did find that wipe transitions becomes clusters on their own and thus create keyframes which are redundant or “half-wipe” frames. We intend to pursue this problem, applying the wipe detection in [21]. Misses mostly occur when the new scene's colors do not vary enough from the previous scene.

6. REFERENCES

[1] B.-L. Yeo and M.M. Yeung. Classification, simplification and dynamic visualization of scene transition graphs for video browsing. In *SPIE Storage and Retrieval for Image and Video Databases VI*, 1998.

[2] M.M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *ICIP '95*, pages 338–341, 1995.

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

[4] J. R. Kender and B. L. Yeo. Video scene segmentation via continuous video coherence. In *CVPR '98*, pages 367–373, 1998.

[5] A.M. Ferman and A.M. Tekalp. Efficient filtering and clustering methods for temporal video segmentation and visual summarization. *J. Vis. Commun. & Image Rep.*, 9:336–351, 1998.

[6] A.M. Ferman and A.M. Tekalp. Multiscale content extraction and representation for video indexing. In *SPIE Multimedia Storage and Archiving Systems II*, 1997.

[7] M.J. Swain and D.H. Ballard. Color indexing. *Int. J. Comput. Vision*, 7(1):11–32, 1991.

[8] H.J. Zhang, S.Y. Tan, S.W. Smoliar, and Y. Gong. Video parsing, retrieval and browsing: An integrated and content-based solution. In *ACM Multimedia '95*, pages 15–24, 1995.

[9] A. Hanjalic, M. Ceccarelli, R.L. Lagendijk, and J. Biemond. Automation of systems enabling search on stored video data. In *SPIE Storage and Retrieval for Image and Video Databases V*, pages 427–438, 1997.

[10] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *ACM MM98*, 1998.

[11] M.S. Drew, J. Wei, and Z.N. Li. Illumination–invariant color object recognition via compressed chromaticity histograms

of color–channel–normalized images. In *ICCV98*, pages 533–540. IEEE, 1998.
[12] J. Wei, M.S. Drew, and Z.N. Li. Illumination invariant video segmentation by hierarchical robust thresholding. In *Electronic Imaging '98: Storage and Retrieval for Image and Video Databases VI*, pages 188–201. SPIE Vol. 3312, 1998.
[13] G.D. Finlayson, P.M. Hubel, and S. Hordley. Colour by correlation. In *Fifth Color Imaging Conf.*, pages 6–11, 1997.
[14] G.D. Finlayson, B. Schiele, and J.L. Crowley. Comprehensive colour image normalization. In *ECCV98*, 1998.
[15] J. Berens and G.D. Finlayson. Log-opponent chromaticity coding of colour space. In *ICPR '00*, 2000. To appear.
[16] E. Sahouria and A. Zakhor. Content analysis of video using principal components. *IEEE Trans. Circ. Sys. Vid. Tech.*, 9:1290–1298, 1999.
[17] B.S. Everitt. *Cluster analysis*. Halsted Press, 3rd edition, 1993.
[18] A. Girgensohn and J. Boreczky. Time-constrained keyframe selection technique. In *IEEE MM Sys.*, pages 756–761, 1999.
[19] M. S. Drew, J. Wei, and Z.N. Li. Illumination–invariant image retrieval and video segmentation. *Pattern Recognition*, 32:1369–1388, 1999.
[20] C.F. Borges. Trichromatic approximation method for surface illumination. *J. Opt. Soc. Am. A*, 8:1319–1323, 1991.
[21] Mark S. Drew, Ze-Nian Li, , and Xiang Zhong. Video dissolve and wipe detection via spatio-temporal images of chromatic histogram differences. In *ICIP'00*, 2000. To appear.

Video(#Frames)				
Method	Correct	#Keyframes	Redundant	Missed
6 Short videos(1455)				
Signatures	17	17	2	2
HistInt	17	75	58	0
Long Video(2004) (Simpsons)				
Signatures	20	16	0	4
HistInt	20	81	63	2
Long video(897) (Basketball)				
Signatures	10	12	4	2
HistInt	10	132	122	0

Table 1. Keyframes generated, compared to algorithm in "Signatures" indicates present algorithm. "HistInt" indicates method in [5], with $k = 3$ and $T_c = 3000$.

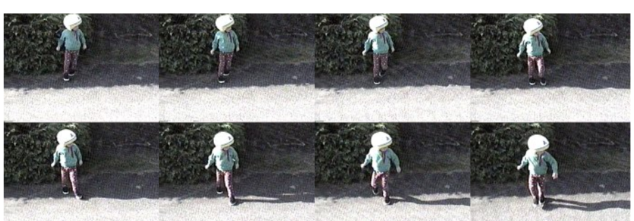


Fig. 1. Video with lighting change.

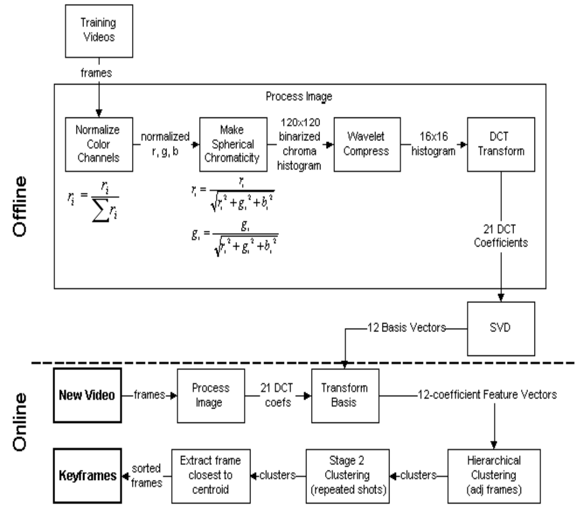
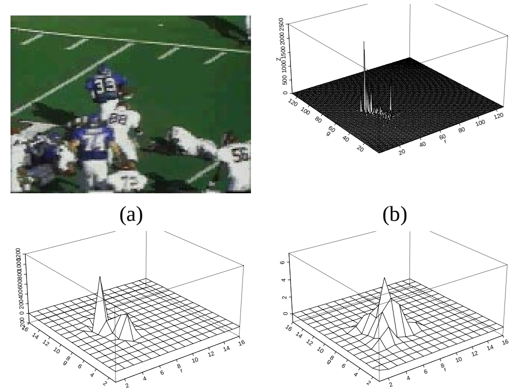


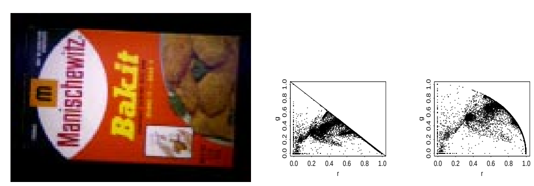
Fig. 2. Clustering algorithm.



(a) (b)

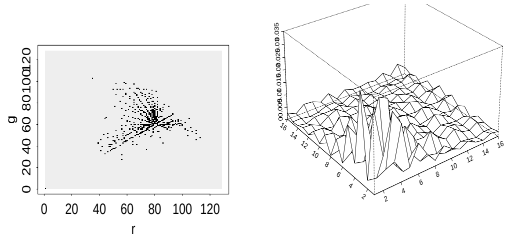
(c) (d)

Fig. 3. (a): Video frame. (b): Chromaticity histogram. (c): Compressed histogram. (d): Binarized histogram, compressed.



(a) (b) (c)

Fig. 4. (a): Image. (b): Linear chromaticity. (c): Spherical chromaticity.



(a) (b)

Fig. 5. (a): Spherical chromaticity signature. (b): Absolute value of DCT of compressed binarized histogram.