Embedded Key Frame Extraction In UGC Scenarios

Alexandro Sentinelli, Luca Celetto
Advanced System Technology - STMicroelectronics
Agrate Brianza (MB) - Italy
Email: alexandro.sentinelli, luca.celetto@st.com

Gustavo Marfia, Marco Roccetti
Computer Science - Università di Bologna
Bologna - Italy
Email: {marfia, roccetti}@cs.unibo.it

***Abstract*—Video summarization is an old problem that offers a wide variety of scenarios and approaches. In this work we investigated the suitability of Key Frame Extraction (KFE) solutions in embedded architectures for mobile platforms. In particular, in our scenario the list of key frames is requested right after a shooting session ends. The most interesting outcome has been the evaluation of performances in User Generated Content (UGC) scenarios through an extensive survey based on Opinion Scores, interviews and other minor metrics. We tested five different solutions and results suggest that pursuing sophisticated algorithms doesn't necessarily enrich the end user experience. We hope that this work will contribute to stimulate the debate on the KFE also in realistic scenarios and to pay attention to the user feedbacks to drive the investigation.**

***Index Terms*—Key Frame Extraction, Video Summarization, MOS, Embedded Architecture**

I. Introduction

In the wide world of Multimedia Information Retrieval (MIR) the browsing tools that help users to navigate in Multimedia (MM) archives have been proved to be very relevant for the usability of any retrieval system. A popular problem in this area is the Video Summarization or Key Frame Extraction (KFE): the aim is to provide a set of frames extracted from the video that represent the content of the video itself. It is proven that a sequence of images is more comfortable and efficient than a bunch of words to read or, an hyperlink to click, open, then stream through a player. Instead a few images can easily describe the content of a (very) rarely tagged video also because, usually, the user who is browsing in the archive is also the author of the video. We point out that UGC are far diverse from TV/professional contents. UGC are unpredictable, short, taken by one camera and content is, not so interesting to watch. All these aspects didn't make UGC scenarios appealing for the research community until a few years ago. In this work we addressed the study of the KFE with the aim to embed a solution in a camera chipset for Smartphones. Since it is important to minimize the computational cost for battery consumption and latency, we tried to exploit, when possible, the available accelerated algorithms on hardware. We will refer to the output of the KFE algorithm as a sequence of KFs (Key Frames) or visual StoryBoard (SB). The added value of STM approach in respect to traditional methods relies on the live computation of the stream: at the end of the shooting session the user has already the list of the KFs. The aleatory behavior of the algorithm and the lack of specific objective metrics suggest being cautious in choosing a definitive solution. However the user must perceive the value of the solution otherwise it is more convenient an arithmetical temporal sampling that brings average performance but it's extremely cheap. We implemented five different architectures and performed experiments on 14 video sequences with the aim to embrace a variety of indoor/outdoor private live scenarios. Our evaluation survey is made by MOS, minor metrics, face to face interviews. Through this work we aimed to contribute to the debate of the KFE addressing the discussion to UGC contents in market scenarios where resources can be limited. Finally we observe the importance of the end user feedbacks to drive the development.

II. Related Works

Lots of different applications in the MM world are linked somehow to the problem of video summarization. It happens that the same algorithms can be used to perform video scene segmentation, semantic understanding, classification, event detection or a mix of tasks that are either overlapping or, at least, correlated to video summarization. In our context, we intend as video summary a limited list of frames picked up from the video that represent the content of the video itself. Such list constitutes the output of the KFE algorithm and is referred as list of KFs (Key Frames) or visual StoryBoard (SB). In the past many techniques have been investigated to extract a list of KFs from a video sequence. In [1] KFE is performed mainly through the motion estimation (ME): the greater the intensity in the center of the frame the more representative is the KF. In [2], instead, the authors point out the steadiness of the camera to measure the relevance of a pose. Other more traditional methods extract color histograms and other low level features. Then the KFs are selected through distance maximization in the feature space [3], or other clustering methods [5]. Such approach has a long tradition in partitioning problems, thus many mathematical tools are available and may be efficaciously reused. However, we also need to take care of the computational cost constraints. In [6] Geraci et al show a cheap and fast clustering method that doesn't affect the performances, but tests are still conducted on TV content. Recently the software R&D community started to embrace UGC scenarios in their tests after the mobile manufacturers understood the great potentials in the CE industry. In [8] Nokia buffers semantic metadata and features to discriminate frame after frame while the camera is streaming. Still on UGC content Kodak [7] uses also face detection engine and

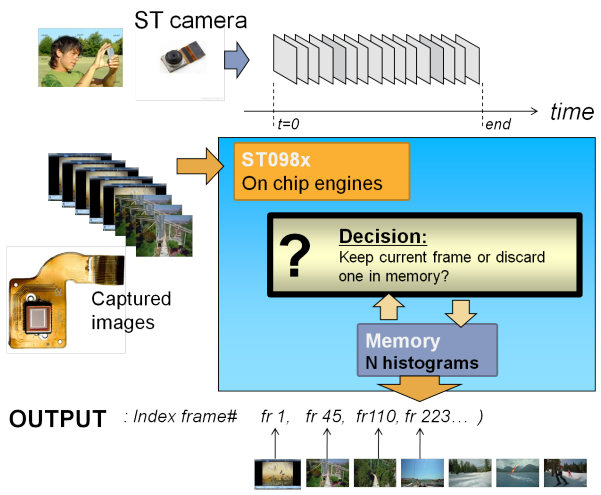

Fig. 1. System overview: the list of KFs is processed while shooting.

audio features. In [9] Intel had focused on the importance of redundancy while choosing KFs. We believe it be an interesting approach although their system was not tested in UGC scenarios.

III. Simulation platform

In figure 1 we show the reference scenario and the basic idea of our architecture. As we said, the purpose of our system architecture is to design a solution where, at the end of the shooting session (up to negligible computation delay) the KFs are available. Therefore the SB has to be stored in a buffer and be always available since we don't know when the video will end. The other development guideline is the integration of the algorithm in the camera chipset. Each raw frame captured by the camera is carried to a decision module that will discriminate if it has to be kept or discarded from the SB buffer. Our simulation model is based on GStreamer open source platform [10]. In the main loop (figure 2) of the decoding process each yuv frame is temporarily available in a buffer just like if it is coming from the camera sensor. Then the frame is passed to the KFE algo, which is the core of our investigation. The majority of input parameters have been decided balancing the computational cost and the performance. Others have been based on heuristics mainly to discriminate similarities. The end of the loop coincides to the end of the shooting session.

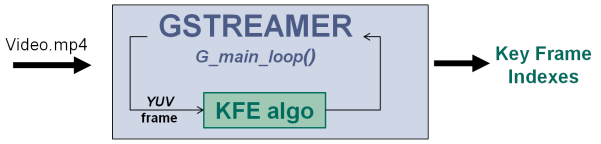

Fig. 2. System overview: the list of KFs is processed while shooting.

An essential brick of the system is the description associated to the frame as frames are compared through their descriptions. Resource constraints obliged us to not test any of the local

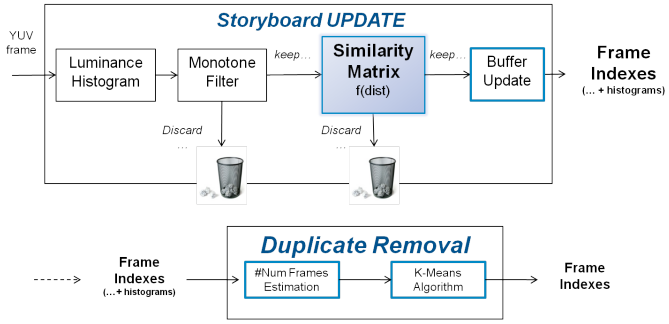

Fig. 3. Full KFE chain: architecture overview.

descriptor family (SIFT, SURF and relatives) but to exploit the descriptions already available on the camera chipset that, from a computational point of view, come for free. As we need to limit the memory stored in the chipset frames are parsed only once, then analyzed and stored in the SB buffer through their available descriptions. This choice has pro and cons effects. On one side we save a lot of memory because the SB buffer do not store images neither raw nor compressed, but only a few integer arrays (a few kB are enough): the final output is represented, therefore, by a pure integer list indicating the absolute position of the KFs in the video. On the other side, the information about the frame is lost so any subsequent analysis can be performed only through their compact histogram description. In the rest of the paper, as they are referred to the same object, we'll refer either to frames or histograms equivalently. Then, in our simulation platform we have allocated additional memory for the KFs also as images, of course, but in a separate buffer and only for evaluation purpose. In the next sub section we describe the main modules of the chain: a monotone frame filter, a buffer where the SB is updated, then a *Duplicate Removal* (DR) at the end of the shooting session(figure 3).

A. Modules

1) Monotone Frame Filter: we will try to introduce more intelligence in the future releases of the algorithm. However, in UGC scenarios, it seemed easier to detect bad quality frames rather than trying to recognize the semantic relevance from a human point of view. Understanding if a frame is "too much monotone", for example a black frame or a fade in/out frame or, when the camera is obscured, seemed to be very simple and efficacious most of the times: if the frame is monotone then it is discarded. The filter works through the evaluation of the traditional luminance cumulative histogram. We found effective representation already when #bins=16; no need to go further. After some tests we opted for a *Zero Forcing*: it is a bit cheaper than other methods based on variance (no root squares to compute). It works with two thresholds, one to select a sufficient number of non null bins, the second one for the height. If the frame is "good", then is passed to the similarity matrix block.

2) Similarity Matrix: if the buffer stores N frames (alias N histograms) there will occur $(N\text{-}1)(N\text{-}2)/2$ distances computa-

tions among the corresponding histograms. All the possible distances are stored in a squared symmetrical matrix M_{diff}. We implemented two options: briefly, maximize the element's sum of the matrix or look for the minimum distance among histograms.

- Maximize Eq 1: it identifies the frame k that, when discarded, maximizes the sum of the elements of M_{diff}:

$$\arg\max_{k} \sum_{i,j} M_{diff}(i,j) \ \forall \ (i,j) \neq k \qquad (1)$$

  We point out that, in this case, the M_{diff} has to compute all similarity values for each new incoming frame. That may be not a big deal for simulations performed by a PC desktop, but can be for the limited resources of an embedded architecture.
- min $M_{diff}(i,j)$: the algorithm looks for the minimum value in the whole M_{diff}. Then the frame i or j is discarded.

3) StoryBoard Buffer: when the frame f_k has been identified through one of the above options the SB buffer is updated by replacing f_k with the new incoming frame from the camera.

4) Duplicate Removal: the last module clusters the histograms through the k-Means algorithm. As the latter needs the #clusters as input, we first need to estimate *K* which, in our case, becomes the very length of the SB. Histograms, in order to keep a natural order of the events, are first ordered temporally and a simple distance among adjacent frames is computed. When the distance is above a certain threshold the #clusters *K* is incremented. Finally the initial set of *K* centroids is chosen by keeping them as equidistant as possible.

B. Distance and similarity

Frames are compared through distances among their histogram representation, which are in fact same dimension vectors. We used the *Generalized Jaccard Distance*(GJD):

$$D(s,z) = 1 - \frac{\sum_i min(s_i, z_i)}{\sum_i max(s_i, z_i)} \qquad (2)$$

Eq 2, in respect to the more traditional Bachattarayya or Euclidean distance, saves around 5% of the computational time over the whole chain. In the next sections we briefly describe the different architectures that we implemented and evaluated.

IV. Methods Description

As we said we implemented 5 chains of KFE. We assumed that an average UGC video length is between ∼15 seconds and ∼2 minutes. Then we set the SB buffer dimension $B_{dim} \cong 30$ which is also *max* size of the SB. For each chain we will provide a brief description.

A. Cumulative Histogram

The *CumulHist* chain relies on a popular histogram representation adopted for images: the cumulative luminance histograms. After many trials we understood that the cumulative histogram, known also as SCD descriptor in the MPEG-7 standard [11], despite its simplicity and ease of computation cannot evaluate more than a macroscopic presence of global light/dark in the scene. The rest of the chain reflects the architecture as shown in figure 3. We tested the two options concerning the M_{diff} update but we didn't observe remarkable differences.

B. Zonal Statistics

The *ZnlStat* architecture is equal to the previous one: only the histogram representation is different. *ZnlStat* frame representation comes from the ST camera chipset and provides information about the spatial distribution of the color inside the image. The image is first segmented by a grid. Therefore statistical information values of red, green, blue, are collected for each area of the grid. The similarity matching seems to perform better; however, this chain doesn't make a notable difference over the whole dataset of video considered.

C. Temporal Sampling (fix)

This solution is nothing but a pure arithmetical division between the length of the video *N* (#frames) and the length *L* of the SB initially set by the user (figure 4).

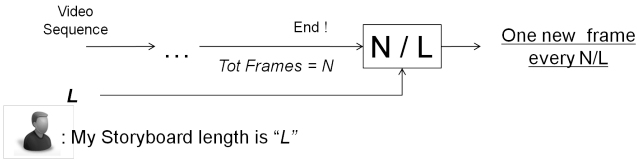

Fig. 4. Pure Temporal Sampling.

After a first benchmarking of the algorithm, when asking to end users, we remarked that the natural temporal sampling of the video sequence had the best ever tradeoff between the output SB quality and the cost of the implementation. The only concern was about the redundancy of the KFs. That's when we thought about a hybrid chain that mixes a pure temporal sampling until the end of the shooting, and then applies the DR to reduce the redundancy.

D. Pseudo Temp ZnlStat

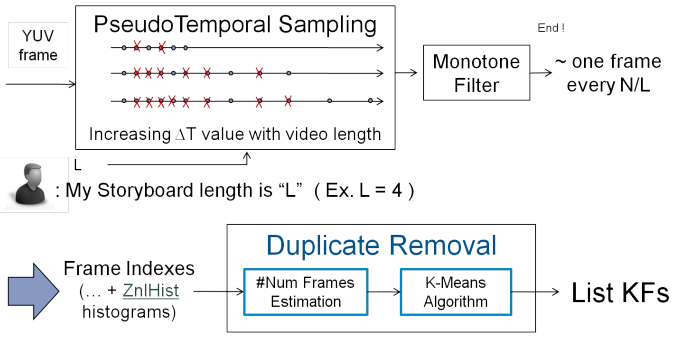

Fig. 5. Pseudo Temporal ZnlStat chain: architecture overview.

As the ratio *N/L* is dependent by *N*, which is unknown while sampling, we need to progressively increase the interval among adjacent frames as shown in figure 5. In particular, by

choosing the buffer dimension L = a power of 2, and doubling ∇T every L samples, we keep a perfect equal distance among sampled frames. While sampling we also store the relative *ZnlStat* histograms in the SB buffer, though they are not used until the end of video shooting. Then the duplicate removal module is applied.

E. Temporal Sampling (var)

The last solution is nothing but the variable version of the temporal sampling algorithm: the possibility to vary L depending by the length of the video through 4 fixed thresholds (figure 6). For example the SB is made by 24 KFs if the video

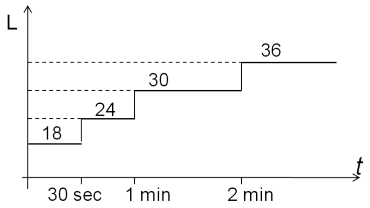

Fig. 6. Variable Temporal Sampling: SB size depends by the video length.

last between 30 and 60 seconds. Although the approach is very simple the trade off cost/performance makes it one of the best KFE candidates for UGC content.

V. Experiments and Results

We benchmarked these 5 solutions on a set of 14 videos taken with a Nokia N900 (res. 848x480). As we didn't find a dedicated dataset of UGC content we shot our own set of video sequences with the aim to cover a variety of indoor/outdoor scenes. We evaluated: Mean Opinion Score (MOS) on 24 users, #Bad Frames, #Redundant Frames, #Key Frames vs an estimated optimal length. Among these metrics we realized that MOS is a reasonable index to estimate so far the performance of the algorithm. After watching the videos all at once evaluators were asked to give a score in [1,10] for each SB (totally 70 SBs). Each SB is a small image composite made by the KFs thumbnails. Since the SB is a tool for browsing in MM archives we encourage the user to look the SB as a whole rather than looking for details in each KF. Then, through one to one interviews we also encouraged users to reveal their subjective perceptions in order to better understand the inner mechanism of the SB summarization. The full test (MOS + interview) lasts $\sim$ 1 hour.

A. Mean Opinion Score

In figure 7 we note that a temporal pure sampling approach (both fix and variable) is often good to meet the needs of an end-user without efforts by the developer and hardware resources. That may depend by both the low quality of the UGC and the length of a typical video sequence shot through a camera phone (rarely more than 2 minutes). It is kind of intuitive that an arithmetical sampling should represent on average the different moments of the story. On the other side, if the story doesn't change KFs can be unpleasantly redundant

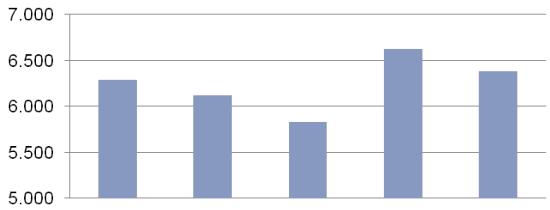

Fig. 7. Average MOS per algorithm.

to a human eye. So the hybrid solution that samples temporally and removes the adjacent duplicates got the highest MOS. We can so far identify trends but not the ideal solution for all videos. For example, the sequence "Elevator_up"shows by far (figure 8) the best MOS with the cumulative histogram chain although the average results tend to point out to the pseudo temporal sampling approach. This is a non trivial aspect when dealing with algorithms that need to be integrated into products for the consumer market. Performances can't be only good on average: they also need to be predictable. It might be tricky otherwise planning the future releases in terms of performance expectations.

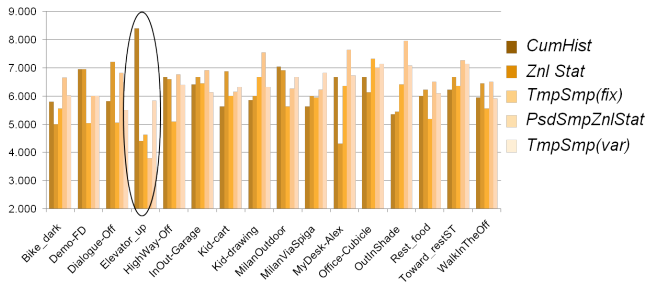

Fig. 8. Average MOS per video sequence.

B. Secondary Metrics

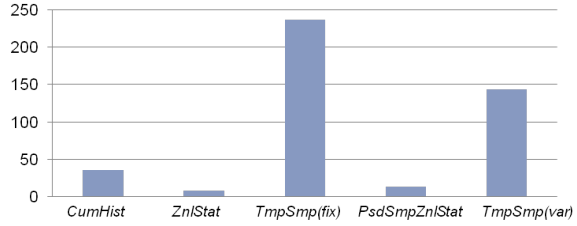

Fig. 9. %error SB length vs optimal estimated length.

In figure 9 we evaluated the error of the SB length (= #KF) for each solution in respect to a ground truth set of (manually) pre-defined SBs. It is pretty expectable that, for this index, the last performing is the temporal sampling as it doesn't discriminate: any frame can be stored. The same effect is witnessed in figure 10 where frames like obscured camera shots are however retrieved.

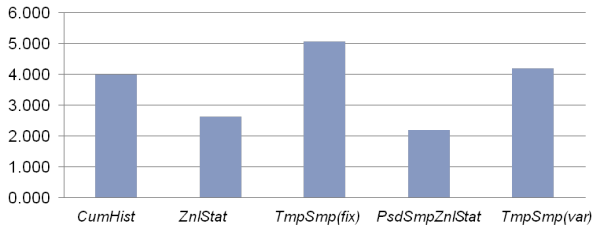

Fig. 10. # bad or non relevant frames.

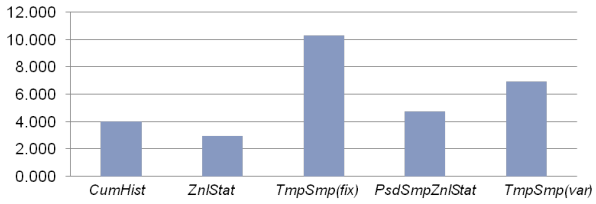

Fig. 11. # redundant frames.

From figure 11, instead, we infer that redundancy raises as with the ratio SB size over the video length. In other words, if the algorithm samples less, KFs have more chance to be different.

C. Issues and Remarks

As expected, we met more issues during the survey in respect to the development. First we had to find a set of videos that represent the content of a typical camera phone user. Second, the evaluation deals with aspects belonging to the variegated world of HMI. The storyboard, in fact, is evaluated through a HMI that can affect dramatically the survey experience. The behavior of the algorithm can be hidden or misunderstood by a wrong evaluation setting and future improvements may take a wrong direction. For example, if thumbnails used for the survey are too small, blurred shapes are less evident, thus any need of sharpness analysis is just ignored. Another aspect to consider is the visual symmetricity of the SB composit. If thumbnails are collated 4 per lines, some user may prefer a SB with 16 KFs rather than a SB with 13 KFs with less redundancy, just because symmetricity is pleasant to the eye. Finally we integrated the MOS collection with one to one interviews asking what they would like to put in the SB and what are the frames that can be considered semantically ”important”. We are confident in saying that CV colleagues were a bit or very much conditioned by their daily activity on images and video analysis, no matter if junior or senior scientists. Because of that the last set of 10 people has been selected from people outside the company or the CV field. Through this benchmark it has been possible to identify the general trends of the five solutions and some highlights for designers who consider to evaluate a KFE algorithm:

- Redundancy is bad, especially among adjacent frames;
- Monotone frames can be part of the story;
- Beauty is somehow important. A colorful and/or symmetrical SB is more pleasant than a representative summary;
- The SB length may depend by the redundancy and relevancy of KF;
- Blurred shapes in key frames are perceived as ”bad”;
- Better a bit of redundancy, but keeping the story as it is;
- The ”story” seems to be a key point to consider: the longer the video, the longer the story, the longer the SB;

Before leaving this paragraph we point out that KFE may serve different taks: browse in a video, browse in a database, recall the content. The developer needs an overview of the whole system and the interaction with the user has to be understood, possibly, prior the development.

VI. Conclusion

Although video summarization is a widely discussed topic in the Computer Vision community, the investigation in mobile use case scenarios did not receive much attention. In this work we focused on a runtime implementation performing the Key Frame Extraction (KFE) for User Generated Contents (UGC) shoot by cellular phones. The computational constraints forced us to look at low cost algorithms trying to exploit, when possible, the information available on the camera chipset. We developed and compared five solutions that have been benchmarked through an extensive evaluation in terms of Mean Opinion Score, interviews and some minor metrics. The most appreciated algorithm was far from embedding complex semantic operations but based on a natural temporal sampling approach with a few more computations to detect low level similarities. This work aims to stimulate the debate on KFE in real market scenarios and point out the importance of the end users feedbacks to drive the investigation to efficient solutions.

References

[1] Narasimha R., Savakis, A., Rao, R., and Queiroz, *Key frame extraction using MPEG-7 motion descriptors*, Proceedings of the Asilomar Conference on Signals, Systems, and Computers (Pacific Grove, CA).

[2] Wolf W. 1996, *Keyframe selection by motion analysis*, In Proceedings of the ICASSP Conference, vol. 2. 1228–1231.

[3] M. M. Yeung and B. Liu, *Efficient matching and clustering of video shots*, Proceedings of the International Conference on Image Processing (Vol. 1)-Volume 1, p.338, October 23-26, 1995

[4] H. Zhang, I. Y. A. Wang, and Y. Alhmbasak, *An integrated system for content-based video retrieval and browsing*, Panem Recognition, vol. 30, pp. 643448,1997.

[5] Y. Zhuang, Y. Rui, T. S. Huang, and S. Mehmtra,*Adaptive Key Frame Extraction Using Unsupervised Clustering*, in Proc. ICIP ’98, Vol. I, pp. 866-870, 1998.

[6] M. Furini, F.Geraci, M.Montangero, M.Pellegrini *VISTO: visual storyboard for web video browsing*, CIVR ’07: Proceedings of the 6th ACM international Conference on Image and Video Retrieval

[7] W. Jiang, C.Cotton, A. Loui, *Automatic Consumer Video Summarization By Audio And Visual Analysis*, ICME 2011, Barcelona, Spain

[8] X. Zeng, X. Xie, K. Wang , *Instant Video Summarization during Shooting with Mobile Phone*, ICMR 2011, Trento, Italy

[9] T. Wang , Y. Gao , P. P. Wang , E. Li , W. Hu , Y. Zhang , J. Yong, *Video summarization by redundancy removing and content ranking*, in MM ’07:Proceedings of the 15th ACM Multimedia, Germany

[10] http://gstreamer.freedesktop.org/, accessed on May the 23th, 2012

[11] Manjunath, B. S., Ohm, J.-R., Vasudevan, V. V., and Yamada, A. 2001. *Color and texture descriptors*. IEEE Trans. Circ. Syst. Video Technol. 11, 6, 703–715.