Efficient Generation of Pleasant Video Summaries

Johannes Sasongko
Faculty of Information Technology
Queensland University of Technology
j.sasongko@student.qut.edu.au

Cyril Rohr
Faculty of Information Technology
Queensland University of Technology
c.rohr@qut.edu.au

Dian Tjondronegoro
Faculty of Information Technology
Queensland University of Technology
dian@qut.edu.au

ABSTRACT
This paper presents an efficient video summarization technique with the focus of generating video summaries that are pleasant to watch. The validity of the technique was tested in the TRECVID 2008 evaluation event. The results show the effectiveness of this technique to produce pleasant video summaries in a short time.

Categories and Subject Descriptors
I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding – Video Analysis.

General Terms
Algorithms, Experimentation, Performance.

Keywords
BBC Rushes Summarization, Video Summarization, TRECVID.

1. INTRODUCTION
As the amount of video that humans store increases, the time that is needed to understand these videos also becomes longer. For example, while searching for a particular video, a user is presented with a list of possible matches. Without having prior knowledge about these videos, the user might have to spend a long time viewing all the search results in order to find the exact video that he is looking for. If the user is presented with *summaries* of these videos, however, this time can be significantly cut down, making the task much faster and less tedious to perform.

The ultimate goal of a video summary is to give users a way to quickly understand the video. To achieve this, a video summary needs to include important segments from the original video and show them in a pleasant way. The problem of selecting important / interesting segments has been discussed many times in the past [1]. Recent techniques are generally based on the concept of *user attention* [2], which we have loosely adapted in this work. However, the "pleasantness" of video summaries has not been the focus of the existing literature. The TRECVID 2007 Rushes Summarization task [3] introduced important measures that represent the pleasantness of video summaries. Namely, these are: ease of understanding, amount of junk, and redundancy. In TRECVID 2008 [4] the "ease of understanding" measure has been replaced with "pleasantness of tempo".

The goal of our video summarization system is to score high in these three pleasantness measures, without sacrificing other measures such as ground truth inclusion and creation time. We do this by removing duplicate shots in the video, ranking the shots by their importance, and building a video summary containing important shots while keeping a constant shot change tempo. The efficiency of our system comes from the fact that it only performs simple analyses on each shot (e.g. length of shot, number of faces, and magnitude of motion).

TVS'08, October 31, 2008, Vancouver, BC, Canada.

2. DESCRIPTION OF THE ALGORITHM
Our video summarization algorithm consists of 5 main steps, as shown in Figure 1. After the last step, all the necessary information has been obtained, and the video summary is ready to be built.

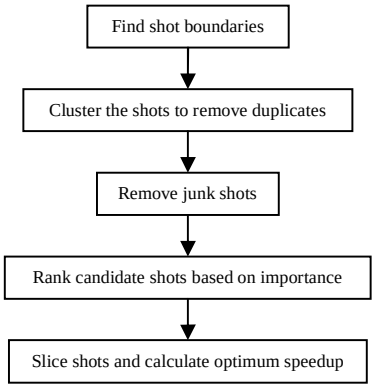

Figure 1. Diagram of the algorithm.

2.1 Shot Segmentation
The first step of our summarization algorithm is to divide the video into shots, where a shot is a video segment taken continuously from one camera. Our shot boundary detection method is based on the color histograms of frames that are chosen with coarse time sampling (only one in ten frames is processed) to significantly cut down the number of frames to be processed. The histogram comparison method is the chi-square test [5]:

$$\chi^2 = \sum_{i=1}^{n} \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} \quad (1)$$

where H_1 and H_2 are the histograms of the two frames, and n is the number of histogram bins. This chi-square value is calculated for every Hue-Saturation-Value color component and weighted by H:S:V = 4:2:1. The use of global color histogram, the chi-square test, and the emphasis on Hue have been shown to be effective for finding shot boundaries [6].

2.2 Shot Clustering

The next phase in the summary generation process is the shot clustering phase. First we create a graph whose nodes represent all shots in the video. An undirected edge is added between each pair of nodes, and the weight of this edge is the distance between the key frames of the two shots, calculated using the chi-square test as mentioned in Equation 1.

After the graph is built, the Minimum Spanning Tree of the graph is computed. All remaining edges that are longer than a threshold is removed, leaving only clusters of nodes (shots). This threshold can be either selected manually or automatically, but for this experiment we simply set it to a good value based on trials.

From each cluster, the longest shot is taken as a candidate shot. This is based on the observation that the longer a shot is, the more likely it is to be important. Although not always accurate, this approach is chosen because it is computationally inexpensive.

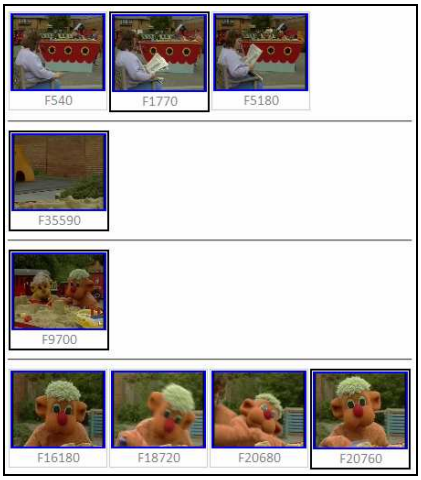

Figure 2. Sample clustering result.
The shots with solid borders are the candidate shots (i.e. the longest shots in their respective clusters).

2.3 Junk Filtering

From the set of candidate shots, we need to remove obvious junk shots, such as blank frames. The dataset that we used for the evaluation also contained another type of junk shot, namely color bar shots.

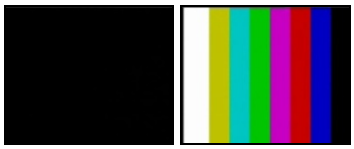

Figure 3. Example junk shots found in the dataset.

Since these blank and color bar frames have distinct histogram characteristics, we calculate the histograms of a known blank frame and a known color bar frame, and compare them with each candidate shot's key frame histogram, as per the chi-square test in Equation 1. If any of them matches, we reject the candidate shot.

2.4 Scoring and Ranking

At the beginning of this step, we calculate the following statistics for each candidate shot of the video:

1. A set of numbers of faces (F) detected at 20-frame intervals along the shot;
2. A set of magnitude values of the motion (M), calculated on 20-frame segments of the shot;
3. The length of the segment (L), in number of frames;
4. The number of shots that are in the same cluster as the shot, which corresponds to the retake frequency (R).

For *F* and *M*, based on our observations, the shots that have the greatest number of faces and the greatest motion magnitude should score high. This is what intuition suggests since shots are interesting when there are people and/or when there is a significant amount of motion. However, instead of just taking into account the mean of each of these measures, we choose to favor segments in which the number of faces is not highly deviated. A low deviation means that the number of faces for each frame does not vary a lot within the shot. We do the same for the amount of motion. The drawback of this approach is the presence of short segments with very high motion or number of faces (e.g. when the crew calibrate their camera or lights) that could potentially bias the results. This issue is of limited impact since we introduce the length of the segment as a parameter of the score. Finally, the number of retakes is taken into account since it is likely that a scene which has required a lot of retakes is important. We do not want to give excessive priority based on the number of retakes (*R*), so a limit of 10 is applied to this measure and, as with the *L* measure, the logarithmic value is added to the overall score.

The score for each segment is the result of the following formula:

$$Score = \frac{mean(F)}{stddev(F)+0.1} + \frac{mean(M)}{stddev(M)+0.1} + \log(L+1) + \log(\min(R,10)) \quad (2)$$

Once a score has been computed for each shot, we filter out every shot which has a score below a certain threshold obtained from experimental observations. We then sort the resulting set of shots by their descending score.

2.5 Satisfying Time Constraints

Most applications of video summarization require each summary video to be significantly shorter than the original video. Considering this fact, it is clear that choices have to be made to satisfy the time constraint: either we speed up the video a lot, with the risk that the resulting summary will be hard to follow, or we remove some shots, with the risk that we will lose potentially interesting content, or a combination of both. To reduce those risks, a third solution is to reduce the size of the shots by taking small portions of them. Thus this technique allows including most of the content without requiring a high speed-up.

The solution we implemented is to allow, a speed-up of at most *MaxSU* times, but only take at most *MaxLength* frames (before speed-up) from each shot. We found that a speed-up of more than 3 times results in the video being too fast to understand, and a length of less than 60 frames results in an unpleasant video due to rapid shot changes. We therefore set *MaxSU* to 3 and *MaxLength* to 60. This results in a bit more than 2 seconds of viewing time for each shot, which is enough to detect the subject of the scene. In addition to reducing the length of the shots, this technique had another positive effect on the summary: since each segment has around the same duration, it results in a very good rhythm when watching the video.

We experimented taking several "slices" of each shot to obtain the *MaxLength*-bounded summary. Finally we settled for taking only

one slice from the middle of the shot. Thus we avoid the beginning and the end of the shots, which could contain visual junk and are likely to not be the most interesting parts; for example, some shots begin by showing a clapper board or end with the crew talking to each other.

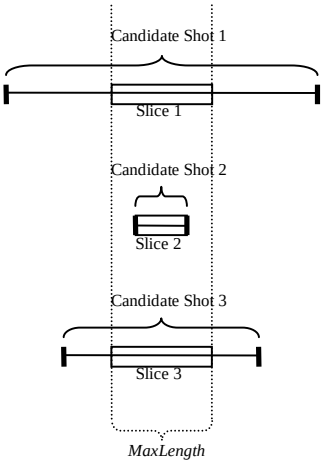

Figure 4. Examples of shot slicing.

After slicing each candidate shot, we calculate the total length of these slices. We then apply the final algorithm, which consists of three cases providing a gradual response to the problem of satisfying the time constraints:

function FixTime(*S*, *T*, *MaxSU*):
  // S *is an array of slices, sorted by descending score.*
  // T *is the target video length.*
  // MaxSU *is the maximum speedup allowed.*
  loop until *S* is empty:
    L = total length of *S*
    // *Case 1: The slices fit into* T.
    if *L* ≤ *T*, **return** *S*
    // *Case 2: The slices fit into* T *after limited*
    // *speedup.*
    // su *is the speedup required for the slices to fit*
    // *into* T.
    su = *T* / *L*
    if *su* ≤ *MaxSU*, **return** SpeedUp(*S*, *su*)
    // *Case 3: Cannot fit all slices.*
    Remove last element of *S*

This algorithm assumes that the maximum allowable length is long enough to accommodate at least one slice; otherwise it results in an empty summary.

Finally, the returned slices are written to the summary video file, in their order of appearance in the original video.

3. EVALUATION

The proposed video summarization method was submitted to the TRECVID 2008 evaluation event together with 42 submissions from other teams. In this evaluation, the target video length is 2% of the original video. This means that the summary of a 30-minute video should last at most 36 seconds.

While observing the evaluation results, we identified three major patterns in the objectives of the different submissions:

1. Pattern 1: Short length, high pleasantness
2. Pattern 2: Medium length, high pleasantness, medium ground truth inclusion
3. Pattern 3: High ground truth inclusion

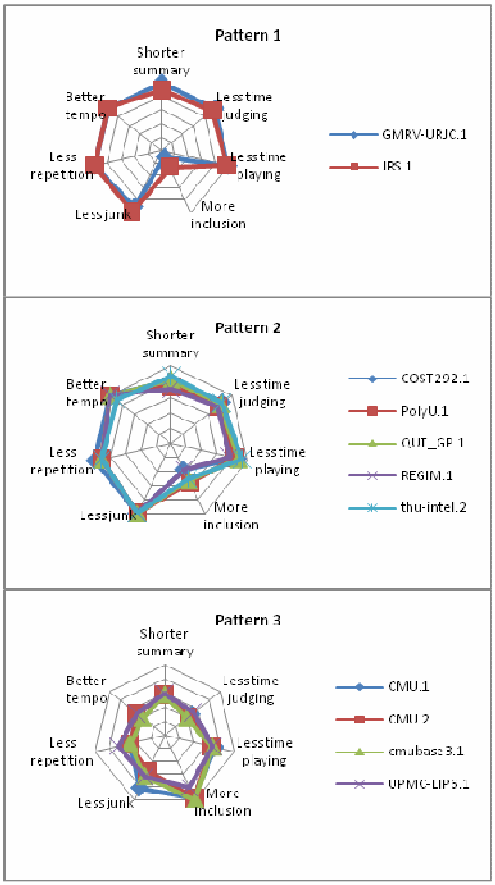

Figure 5. Three patterns in the evaluation results. Note that the axes do not scale in the same way; they are only meant to show participants' scores relative to each other.

As shown on Figure 5, our algorithm falls into the second pattern, which maximizes the three pleasantness measures—better tempo (TE), less repetition (RE), less junk (JU)—without sacrificing too much ground truth inclusion. In line with our aims of creating pleasant video summaries, our system succeeds in obtaining high scores in these three measures that we consider representing the pleasantness of the summaries, as shown on Table 1.

Table 1. Systems with top three pleasantness scores

Rank	Systems	Pleasantness (TE+RE+JU)/3
1	COST292.1, JRS.1	3.6667
2	PolyU.1, **QUT_GP.1**, REGIM.1	3.5567
3	GMRV-URJC.1	3.5533

The "shorter summary" and "more inclusion" measures seem to be opposites of each other; short summaries yield less ground truth inclusion, while more ground truth inclusion is possible given longer summaries. Figure 5 shows this relationship: systems producing short summaries tend to neglect ground truth inclusion (Pattern 1), while systems that focus on inclusion produce long summaries and are less pleasant (Pattern 3). As with other algorithms in the second category, we position ourselves in the middle of both measures, producing short summaries with reasonable ground truth inclusion (see Figures 6 and 7).

Parameters in our algorithm can be modified in order to achieve results more similar to the first and third patterns. The maximum video speed-up (*MaxSU*) can be increased to increase the ground truth inclusion at the cost of pleasantness (tempo). The maximum slice length (*MaxLength*) can also be decreased to obtain the same effect. If ground truth inclusion is not important, the maximum summary length (*T*) can be reduced, and the results will be closer to pattern 1. This shows the flexibility of our algorithm, as these different parameters can be tweaked depending on preference.

In terms of efficiency, our system ranked the eighth in the average summary creation time (see Figure 8), which is the best among the 6 systems with highest pleasantness scores mentioned in Table 1. The machines running our code consist of an Intel Core 2 Duo 1.83 GHz with 2 GB RAM running Windows Vista (for shot segmentation, clustering, filtering, and scoring) and an Intel Core 2 Duo 2.16 GHz with 2 GB RAM running Mac OSX (for shot ranking, time fitting, and video writing). The two machines were *not* running in parallel, and we do not take into account the post-processing time to compress the video files.

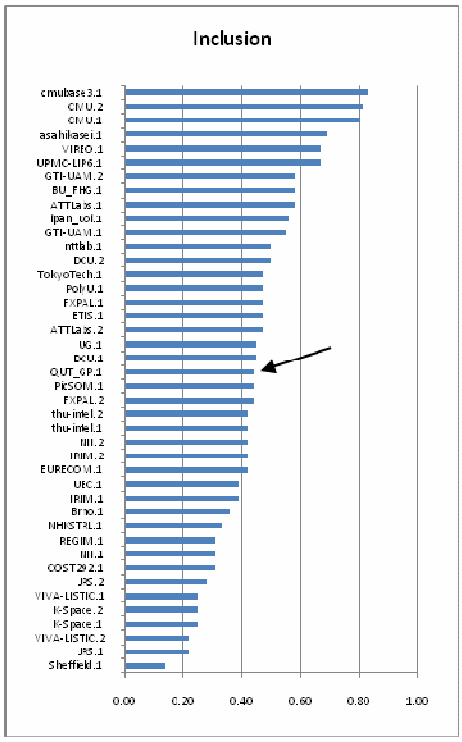

Figure 7. TRECVID 2008 average ground truth inclusion.
Our system (QUT-GP.1) is the 21st from the top.

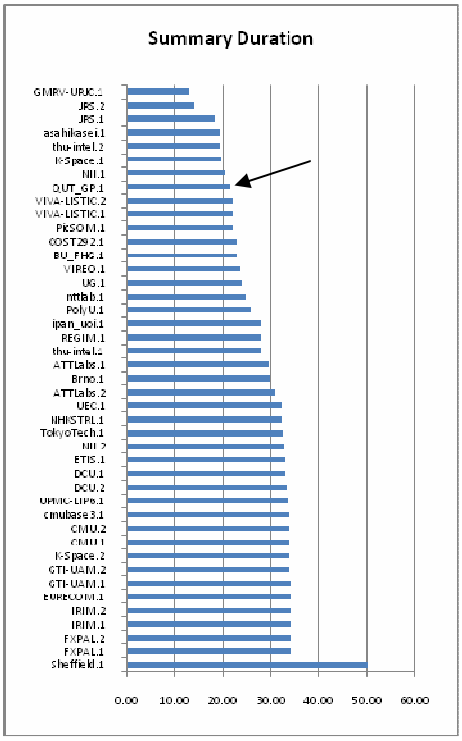

Figure 6. TRECVID 2008 average summary duration.
Our system (QUT-GP.1) is the eighth from the top.

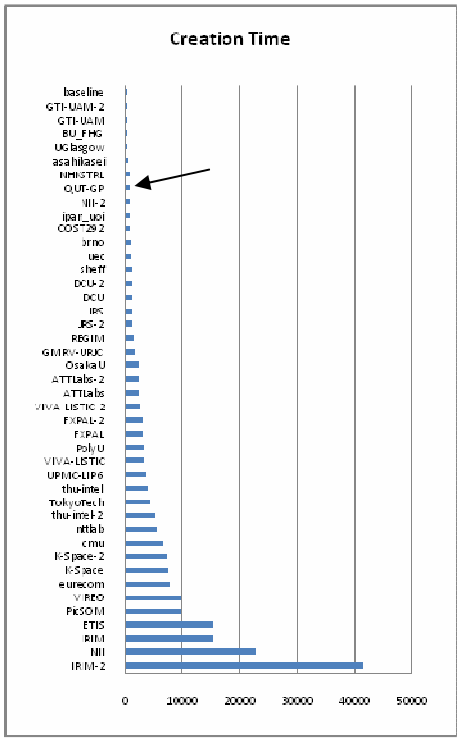

Figure 8. TRECVID 2008 average summary creation time.
Our system (QUT-GP) is the eighth from the top.

4. CONCLUSION AND FURTHER WORK

In this paper we have presented an effective and efficient video summarization algorithm. A fully working system was implemented and tested on the TRECVID 2008 Rushes dataset. The evaluation results show that the system is able to create video summaries that are pleasant to watch and relatively accurate, while being very fast in terms of processing speed. Important factors in our success are the duplicate removal process, and the shot ranking and time fitting algorithms.

In the future, other shot scoring measures will be considered, for example audio intensity, pitch, etc. The clustering algorithm will also be improved to reduce the number of shot duplicates that still occurred in some of the video summaries.

5. REFERENCES

[1] Truong, B. T. and Venkatesh, S. 2007. Video abstraction: A systematic review and classification. ACM Trans. Multimedia Comput. Commun. Appl. 3, 1 (Feb. 2001), 1-37. DOI= http://doi.acm.org/10.1145/1198302.1198305.

[2] Ma, Y. F., Lu, L., Zhang, H. J., and Li, M. 2002. A user attention model for video summarization. In Proceedings of the Tenth ACM International Conference on Multimedia (Juan-les-Pins, France). ACM, New York, NY, 533-542. DOI= http://doi.acm.org/10.1145/641007.641116.

[3] Over, P., Smeaton, A. F., and Philip, K. 2007. The TRECVID 2007 BBC rushes summarization evaluation pilot. In Proceedings of the International Workshop on TRECVID Video Summarization (Augsburg, Bavaria, Germany). ACM, New York, NY, 1-15. DOI= http://doi.acm.org/10.1145/1290031.1290032.

[4] Over, P., Smeaton, A. F., and Awad, G. 2008. The TRECVid 2007 BBC rushes summarization evaluation. In Proceedings of the International Workshop on TRECVID Video Summarization (Vancouver, British Columbia, Canada). ACM, New York, NY, 1-20.

[5] Patel, N. V. and Sethi, I. K. 1996. Compressed video processing for cut detection. IEE Proc.-Vis. Image Signal Process. 153, 6 (Oct. 1996), 315-323.

[6] Lupatini, G., Saraceno, C., and Leonardi, R. 1998. Scene break detection: A comparison. In Proceedings of the Eighth International Workshop on Research Issues In Data Engineering: Continuous-Media Databases and Applications (Orlando, FL). DOI= http://dx.doi.org/10.1109/RIDE.1998.658276.