Summarization scheme based on near-duplicate analysis

David Gorisse, Frederic Precioso and Sylvie Philipp-Foliguet
ETIS, CNRS, ENSEA, Univ Cergy-Pontoise
F-95000
Cergy Pontoise, France
{gorisse, precioso, philipp}@ensea.fr

Matthieu Cord
LIP6, UPMC-P6
104 av Kennedy 75006
Paris, France
matthieu.cord@lip6.fr

ABSTRACT

This paper presents our approach to select relevant sequences from raw videos in order to generate summaries to TRECVID 2008 BBC Rush Task. Our system is composed of two major steps: First, the system detects "semantic" shot boundaries and keeps only non-redundant shots; then, the system estimates average motion for each shot, as a criterion of amount of information, to better share out the duration of the summary between remaining shots. The first step is based on a fast near-duplicate retrieval using Locality Sensitive Hashing (LSH) which provides results in few seconds (if we do not take into account decoding and encoding processes). The evaluation of TRECVID shows very promising results, since we ranked 17^{th} over 43 runs, regarding redundancy measure (RE), and 18^{th} for object and event inclusion (IN). These balanced results (most of best teams for the first criterion are among the latest for the second one) show that our method offers a quite good trade-off between false negatives (IN) and false positives (RE).

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia information systems: video

General Terms

Algorithms, Experimentation

Keywords

LSH, Near-duplicate, Video summarization

1. INTRODUCTION

This paper describes in detail our system producing summaries for Trecvid 2008 BBC Rush task. The aim of this task is to automatically generate mpeg-1 summary clips of 2% of the original duration of raw videos. These videos are coming directly from movie production without any post-production process. In order to achieve this objective, the system must remove as many repetitive shots as possible while preserving the most significant shots. Resulting summaries must be pleasant to see and most contain relevant parts or events of the videos. Frame clustering is one of the early approaches to address video summarization task [6, 9, 7, 8]. Following this direction, many other methods have been recently proposed: in [11], Truong et al. proposed a method of hierarchical shot clustering based on hierarchical Sift description of each frame. This approach intends to avoid complex implementation in terms of concept detection and excerpt assembly (i.e, no picture-in-picture, split screen and special transitions). In [3], Chen et al. use clustering to remove redundant shots and the shot segmentation is based on kernel correlation of pairwise inter-frame similarity features (color and motion). In our method, we try to combine several good properties of the aforementioned methods: avoiding complex implementation in terms of concept detection and excerpt assembly, hence we consider only global features; an approximate clustering to remove redundant shots but also to segment videos into semantic shots based on a near-duplicate approach.

Our system uses one simple global descriptor, HSV color histogram, as input of the LSH algorithm to efficiently compute near-duplicates of each frame. We exploit the analysis of the sets of near-duplicate frames LSH provides us with in order to segment the videos into semantic fragments and remove redundant parts. We compute a classic skimming process, to adapt the frame sampling rate in each relevant fragment, using average motion estimation based on our phase correlation descriptor. The computational power of LSH approach allow us to further consider larger or more complex features which should increase the precision of semantic fragment extraction and consequently improve the visual quality of our summaries while facilitating the skimming process

2. SYSTEM OVERVIEW

Due to acting mistakes or film making process, each scene is usually shot several times, which results in many repetitive sequences of frames in rush videos. Our system selects short excerpts from videos, identifying non-redundant segments which contain actions. In this paper, we relate action to motion, moving objects and camera motion. The input video is decoded and only 1 frame over 4 are kept. This decimation allows to decrease the computational complexity without loss of information. Two kinds of features are computed from these decoded frames : HSV color histogram and the entropy of phase correlation. Color histograms are involved in several modules of our system: to remove junk

frames, to segment the video into "semantic" shots and to reduce redundant information. The entropy of phase correlation, defined on frame blocks, describes coarse local motion. This feature is used as an action detector and a duration factor to rule the final skimming process. Figure 1 is an overview of our video summary system and we detail each module in next sections.

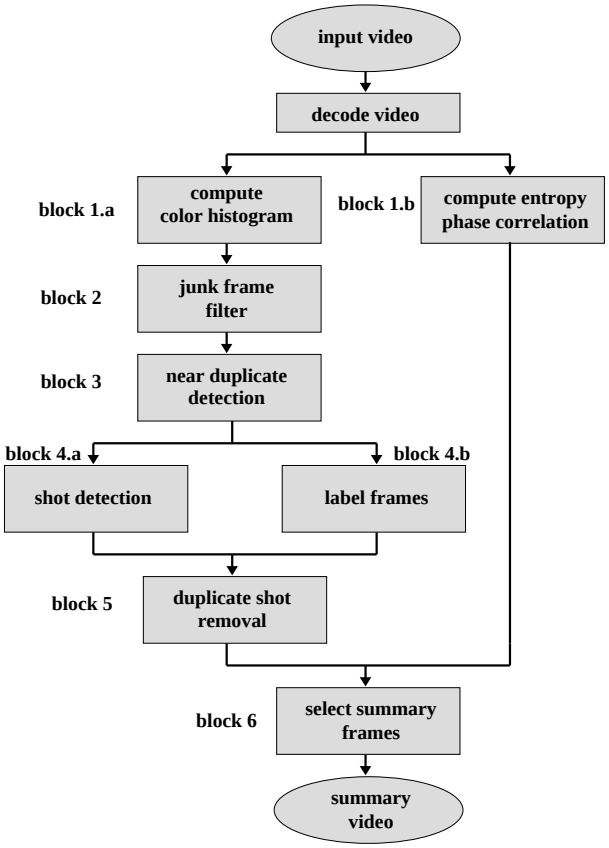


Figure 1: Block diagram of system

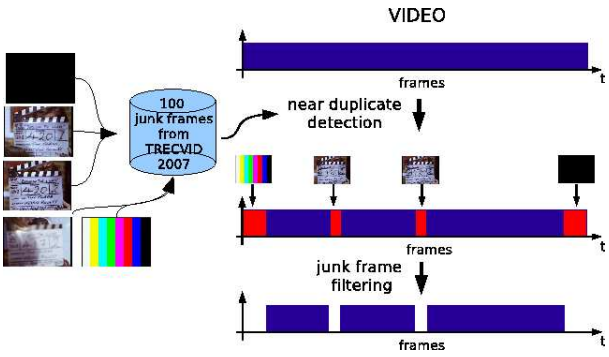


Figure 2: junk frame filtering

To briefly describe the system (fig.1):
In *block 2*, junk frames are removed. Junk frames include rainbow color bars, clap-boards, black frames (that can be due to objects accidentally covering up the camera lens) or white frames (due to dazzling), etc., and are classically considered as useless. We provide a dataset of about one hundred junk frame samples, extracted from TRECVID 2007 videos. Then, all the frames, from raw video sequences, detected as near-duplicate to one of these samples are removed (fig.2).
In *block 3*, a data set containing each remaining frames of the video is formed. Iteratively, each frame is used as query and a near-duplicate detection is carried out. The result of this process is stored in a square binary matrix of size number of frames by number of frames. If the j^{th} frame of the video is a near-duplicate to the i^{th} one, a 1 is stored at i^{th} row and j^{th} column, otherwise this value is 0. The matrix is quite sparse (fig.3).

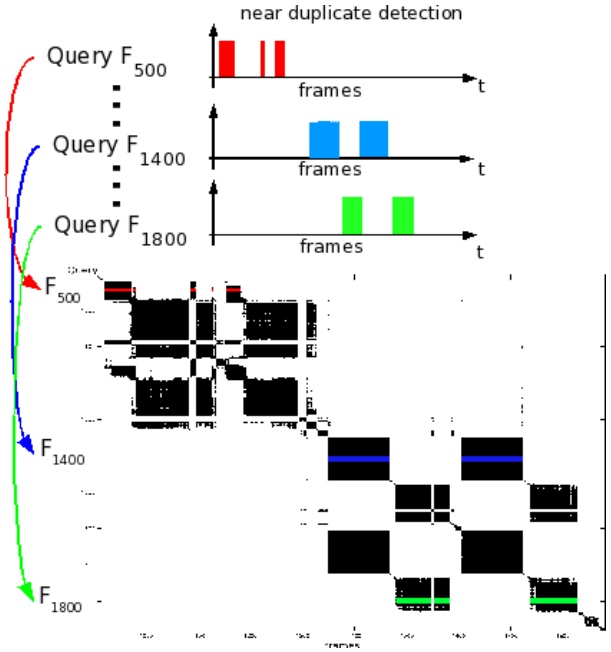


Figure 3: Near-duplicate detection

In *block 4*, this matrix is analyzed to detect shots and to label frames by clustering similar frames.
In *block 5*, duplicate shots are then removed by keeping the longest one among all the shots sharing the same label.
Finally, in *block 6*, a variable amount of frames in each remaining shot is selected by considering that shots containing more motion require more time to be summarized.

3. FAST FRAME MATCHING

In this part, we will dive into details on how we obtain the near-duplicate matrix. This matrix is supposed to provide a clustering of all the frames having the same sementic information in order to detect redundant shots. This latter process will be discussed in the next part. We start by explaining the choice of features for the near-duplicate detection. Then we will present LSH algorithm which provided us with a fast approximation of near-duplicate detection able to process the large amount of frames of the BBC Rush dataset.

3.1 Content representation for near-duplicate detection

The role of the features used to represent each frame content is double: bring together the frames of a same shot and

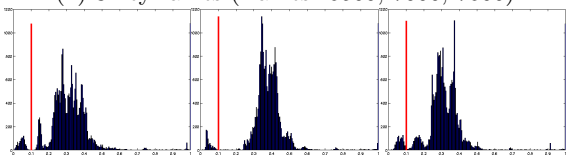
locate duplicate shots. Since during a shot objets can move, we only considered global features. Several features were tested [2]: HSV color histogram of 64 bins, vertical projection accumulative histogram, horizontal projection accumulative histogram... The protocol of the test is as follows:

- Extraction of the keyframes of a video (tab. 1.1)
- For each keyframe, computation of the distance distribution to all frames of the video (tab. 1.2).
- Estimation of the distance between modes and selection of a threshold R.
- Visualisation of selected frame on a temporal scale (tab. 1.3).

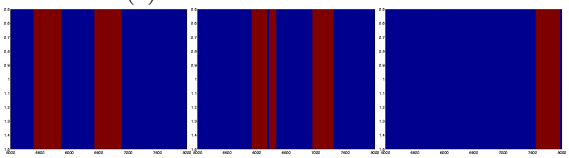
As we can see from table 1, the distribution of HSV histogram inter-frame distance reveals several distinct modes with a fairly clear separation around 0.1 for all the keyframes. Moreover, by locating the frames where HSV histogram distance is less than 0.1 (in red in Table (1.3)), there are many dense blocks corresponding to shots interspersed with transitions or other shots (in blue in Table (1.3)). HSV color histogram were selected to describe frames because this feature, compared to the others we tested, maximizes the distance between modes. Each frame of the rush videos is considered as a query and a near-duplicate search is carried out to find all frames whose distance to the query is lower than 0.1. We store the result of these successive searches into the near-duplicate matrix (fig.3).

Table 1: near-duplicate detection with HSV histogram (MS210470)

(1) 3 keyframes (frames: 6500, 7000, 7600)



(2) HSV distance distribution



(3) time position of image selected with $R = 0.1$

As the frame number of a video is quite big, about 10,000 (after our decimation, mentionned at the beginning of this paper), brute force search is a complete non-sense, given the high dimensionnality and size of the data. Hence as proposed by Chum et al. [4], we use the Locality Sensitive Hashing (LSH) scheme, briefly described in the following section, to efficiently find the frames within a given distance to the query.

3.2 Locality Sensitive Hashing scheme

We shortly report in this section the basic LSH functionalities to explain how we use it in our context.

LSH solves the $(R, 1+\epsilon)$-NN problem: find at least one vector $\mathbf{b}'$ in the ball $B(\mathbf{q}, (1+\epsilon)R)$ if there is a vector $\mathbf{b}$ in the ball $B(\mathbf{q}, R)$. $\mathbf{b} \in B(\mathbf{q}, R)$ if $||\mathbf{b} - \mathbf{q}|| \leq R$. Indyk and Motwani [10] solved this problem for the Hamming metric with a complexity of $O(n^{1/(1+\epsilon)})$ where n is the number of vectors of the database. Datar and al. [5] proposed an extension of this method that solves this problem with the Euclidean metric and with similar time performances. The method generates some hash tables of points, where the hashing function works on tuples of random projections of the form:

$$h_{\mathbf{a},c}(\mathbf{b}) = \lfloor \frac{\mathbf{a}.\mathbf{b} + c}{w} \rfloor$$

where $\mathbf{a}$ is a random vector whose each entry is chosen independently from a Gaussian distribution, c is a real number chosen uniformly in the range $[0, w]$ and w specifies a bin width (which is set to be constant for all projections).

A tuple of projections specifies a partition of the space where all points inside the same part have the same key. All points with the same key are stored in the same bucket C. Clearly, if the number of projections is carefully chosen, then two points which hash into the same bucket C will be nearby in the feature space. To avoid boundary effects, many hash tables are generated, each using a different tuple of projections. In practice, a proportion of these points (called "false matches") will be at a distance greater than R from the query point $\mathbf{q}$. That is why, a check (computation of the Euclidean distance between all points $\mathbf{b}$ of bucket C and $\mathbf{q}$) is carried out to remove "false matches".

In our case, we do not want to find only one vector $\mathbf{b} \in B(\mathbf{q}, (1+\epsilon)R)$ but all of them. For that, we use a method from E^2LSH [1] which is a modified version of [5] to solve the $(R, 1-\delta)$-near-neighbor problem: each vector $\mathbf{b}$ satisfying $||\mathbf{b} - \mathbf{q}|| \leq R$ has to be found with a probability $1 - \delta$. Thus, δ is the probability that a near-neighbor $\mathbf{b}$ is not reported.

For our experiments, we used 10 random projections for 50 hash tables, and $R = 0.1$. With this parameter, we reached 9.75 sec to compute a near-duplicate matrix of 10.000 frames on a 3.2GHz Pentium IV PC with 8Go of RAM memory.

4. REDUNDANT SHOT DETECTION

In this part, we detail the three stages of redundant shot detection from the near-duplicate matrix. The first two stages: shot segmentation and frame labelling, consist in clustering the frames. The last stage: shot labelling, unifies the two previous decisions to detect redundant shots. It may seem useless to use two separate clusterings but as we will see, these two stages have complementary behaviours.

4.1 Shot segmentation

The aim of this module is to cluster adjacent near-duplicate frames along the diagonal of the near-duplicate matrix.

To achieve good performance in computing time, we process data sequentially. In the ideal case, all frames of a shot are near-duplicate of the others, one decision by shot is sufficient. We could process the near-duplicate search of a frame l. Initializing two pointers on the column l, one looking for the first 0 before l and the other one, looking for the first

0 after l, we could search the longest interval of continuous near-duplicate frames containing the frame l and thus detect the shot S_l. Hence, it could be sufficient to detect the next shot to repeat the process with the frame next the end of the shot S_l. But as we can see on figure 4, transitions

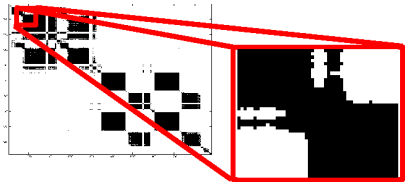


Figure 4: Transition between shots are not sharp

between shots are rarely abrupt and shot detection must be more robust. For instance, claps which caracterize shot transition appear progressively and as we use 64 bits color histogram, the descriptor is gradually deteriorated. Instead of taking one detection for each shot, we made the previously described detection for each frame and we merged decisions.

Figure 5 shows a result of our shot detection scheme (in red) on the near-duplicate matrix.

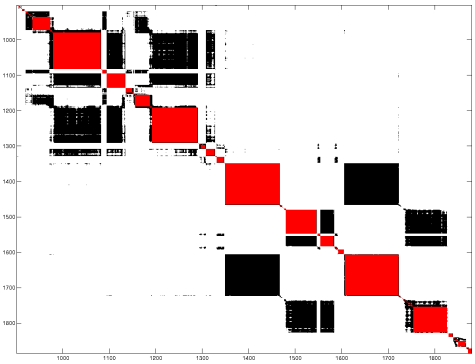


Figure 5: Shot detection from the near-duplicate matrix of fig.3

4.2 Frame labelling

The aim of this module is to cluster all frames that contain the same semantic information. As the shot detection, frame clustering is carried out using near-duplicate detection and to speed up the process, the clustering is iteratively performed after each query. The idea is to decide which group the current frame belongs to, then to propagate the decision to the near-duplicate frames. As a current decision does not affect a previous decision, we only need to consider the upper triangular of the near-duplicate matrix.

To do that, a table $TLabel$ of size number of frames is initialized to 0. For each query frame q, the label contained at bin q of $TLabel$ is selected. If the label is 0, a new label is affected. For each near-duplicate frame b of query q, we modify the value $TLabel(b)$ into $TLabel(q)$.

The noise in the near-duplicate matrix (fig.3) may significantly degrade the clustering. Indeed, a false detection can unify two groups of frames that do not share the same information. To limit this problem, we have filtered the matrix by removing all groups of near-duplicate frames of less than 25 ms.

As we can see on figure (fig.6), we managed to split the matrix into 3 groups.

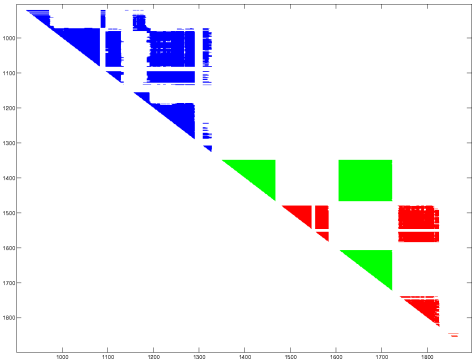


Figure 6: label frames

4.3 Shot labelling

The unification of the two previous stages of clustering is quite simple, it consists in scanning each shot obtained by shot detection step and in assigning a label provided by the stage frame labelling. As the clustering provided by the frame labelling step is not perfect, noise of the near-duplicate matrix can induce false detection, shots may contain several labels. For this reason, we affect the predominant label (fig.7). It may happen that some shots (those which last less than 25 sec) do not contain labelled frames. In this case, we assume these shots are too short to represent significant information and thus are removed.

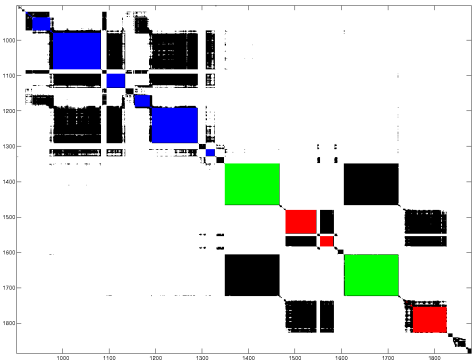


Figure 7: shot labelling

5. FRAME SKIMMING

From now on, we have detected shots and set them a label that allow to identify redundant shots. To build the summary, we have just to remove redundant shots and to select relevant frames of each remaining shot respecting the duration assigned.

5.1 Redundant shot removal

It is difficult to know what shots have to be kept to make the final summary. If we consider that a shot is replayed when an actor makes a mistake, common sense would like to keep the last detected duplicate but sometimes, only a part of the shot is replayed. Thus we decided to keep the longest shot (fig.8). As we can see on figure (fig.7), some shots are replayed after other ones, the shot in green is replayed after the shot in red. If we decide to delete the first green shot and to keep the first red shot, the two shots order will be reversed. Hence we sort shots by order of label, which helps

keeping the order in which shots were played for the first time.

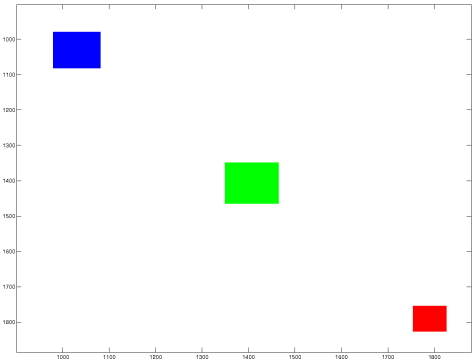


Figure 8: duplicate shot removal

5.2 Summary frame selection

We now have the main shots of rush, we must then select some frames of each shot to respect the limited time of the summary. The duration of the summary is limited to 2% of the duration of the video. To prevent automatic scoring from misleading our video summary systems to favor extremely short clips, we imposed a constraint that all shots of a summary must last at least 1 second. 1 second is close to the lower limit that humans can comfortably reconize non-trivial visual content on a screen.

After this selection, two cases may occur: either the duration of the abstract is too long, or it is too short. As long as the duration of the summary exceeds the limited time, we remove the shortest shot. We assume that the more motion a shot contain the more time it requires to sum it up. Hence, as long as the summary is not long enough, we increase the duration of shots that contain the most motion.

In this way we guarantee that the duration of a summury does not exceed the limited time.

Once the time is divided between shots in this way, we sample each shot by uniformaly selecting the number of frames assigned throughout shot.

6. RESULTS

To evaluate the TRECVID 2008 Rushes Task, the NIST considers 8 criteria : DU - duration of the summary (sec), XD - difference between target and actual summary size (sec), TT - total time spent judging the inclusions (sec), VT - total video play time (versus pause) judging the inclusions, IN - fraction of inclusions found in the summary, JU - Summary contained lots of junk, RE - Summary contained lots of duplicate video and TE - Summary had a pleasant tempo/rythm.

Regarding the visual aspect of summaries, our results are not good, we are ranked 37^{th} over 43 runs for TE. We can conclude that our frame selection consisting of uniformely sample shots are not a good choice. Indeed, summary resulting are very jerking. Moreover, this effect also makes difficult the understanding of our summary, that also affects our ranking for TT and VT (respectively 38^{th} and 34^{th}).

Our results for IN and RE are very promising, we are ranked 18^{th} and 17^{th}. Indeed, all our scheme is based on the detection and removal of duplicate shots. The fact that we are properly classified for these two tasks shows that we have a good trade-off between non-detection and false detection.

Despite the fact that we are classified 40^{th} for effort consuming with 15236,2 sec on average per summary, the strong point of our method is that we have proposed a very fast scheme based on LSH to built summary. Indeed, once the features are extracted, our system creates the summary in less than 12 sec (for junk frame filtering, redundant shot detection and frame skimming) when run on a 3.2GHz processor and 8Go of main memory. We explain this difference of time by the fact that the program used to extract both HSV color histogram and phase correlation, is developed with Matlab whitout optimization. We can reasonably hope to decrease the computational time by a factor 10 by optimizing the program and reach a factor 100 by rewritting the program in C.

7. CONCLUSIONS

To summarize the video content, we proposed a system using mainly one simple global descriptor, HSV color histogram, as input of an approximate clustering based on LSH to reach good computational performance. Our system gives encouraging results for redundancy removal and relevant video fragment extraction. We use a classic skimming process, to adapt the frame sampling rate in each relevant fragment, based on an average motion estimation provided by our phase correlation descriptor. However, our frame sampling rate, for each summarized fragment, produces parkling effects and thus provided us with a low score for "pleasant rythm/tempo" criterion. The computational power of LSH approach allow us to further consider larger or more complex features which should increase the precision of the semantic fragment extraction and consequently facilitate the skimming process and improve the visual quality of our summaries.

8. REFERENCES

[1] A. Andoni. E2lsh. http://www.mit.edu/∼andoni/LSH/.

[2] G. Camara-Chavez, M. Cord, S. Philipp-Foliguet, F. Precioso, and A. de Araujo Albuquerque. Robust scene cut detection by supervised learning. *EUSIPCO*, 2006.

[3] F. Chen, M. Cooper, and J. Adcock. Generating comprehensible summaries of rushes sequences based on robust feature matching. *ACM int. workshop on TRECVID video summarization*, pages 30–34, 2007.

[4] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. *ACM CIVR*, pages 549–556, 2007.

[5] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Symposium on Computational Geometry*, pages 253–262, 2004.

[6] A. M. Ferman. Two-stage hierarchical video summary extraction to match low-level user browsing preerences. *IEEE Trans. on Multimedia*, 5(2):244–256, 2003.

[7] Y. Gong and X. Liu. Summarizing video by minimizing visual content redundancies. *IEEE ICME*, 2001.

[8] Y.-H. Gong. Summarizing audio-visual contents of a video program. *In EURASIP JASP*, 2003.

[9] R. L. A. Hanjalic and J. Biemond. Automated high-level movie segmentation for advanced video retrieval systems. *IEEE Trans. CSVT*, 9(4):580–588, 1999.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *30th Symposium on Theory of Computing*, pages 604–613, 1998.

[11] B. T. Truong and S. Venkatesh. Generating comprehensible summaries of rushes sequences based on robust feature matching. *ACM int. workshop on TRECVID video summarization*, pages 30–34, 2007.