

Evaluation of a practical fingerprinting system

Daniel Pereira, Luis Loyola
Research and Development Department
SkillUpJapan Corporation
Tokyo, Japan
{d.pereira,loyola}@skillupjapan.co.jp

Abstract— Video fingerprinting is one of the most promising technologies to uniquely identify video content in the Internet in a totally format-independent way. The logic behind the creation of the video fingerprints or perceptual hashes is a key factor for the performance of the system, especially in terms of storage and search efficiency. Although several video fingerprinting methods have been proposed, most of them largely focus on the robustness of the perceptual hashes to distortions, giving much less importance to the required storage as well as the efficiency of the database structure for fast and reliable format-agnostic searches of video content. In this paper we propose a video fingerprinting system with an excellent robustness to distortions but also offering lower storage requirements and a more efficient database structure than prior schemes.

Keywords- video processing; pattern recognition; perceptual hash; information storage; video fingerprinting;

I. INTRODUCTION

The growth of video-related services in the Internet is increasingly creating needs for uniquely identifying audiovisual content in the Internet using fingerprinting techniques. Applications that may profit from such technology include automatic tracking of copyrighted titles in user generated content sites, tagging movies for reducing the cost of its distributed storage by creating perceptual indexes instead of storing multiple physical replicas, and improving video search engines in a totally video-codec and video-container independent way.

Video fingerprints are perceptual hashes that allow us to discriminate one video from all the rest in a totally binary-independent way, i.e. irrespective of the codec or format in which they have been compressed. The main requirements for such a system are given below:

- **Uniqueness:** a piece of video content must be associated to a unique hash value irrespective of its format.
- **Storage:** the amount of bits generated per unit of video time from the hashing system should be as low as possible.
- **Database:** the necessary time to find a specific piece of audiovisual content inside a huge pool of contents should be as low as possible. To do that the logic behind the creation of the hash is quite important since it defines the efficiency of the database structure.

- **Robustness:** the perceptual hash should be robust to malicious or intentional distortions including chromatic alterations, addition of watermark and subtitles as well as video rotations, changes to its resolution, frame size, number of frames per second, etc.

Several video fingerprinting methods have been proposed. Most of them relate to pixel-level statistical analysis due to their low computational complexity. Lee and Yoo [1] propose an approach based on centroid of gradient orientation (CGO), while Lee and Suh [2] proposal focuses on the orientation of luminance centroid and Lowe [3] uses the histogram of gradient orientation. Other approaches propose histogram similarity calculations [4], ordinal measures of block means [5] and differential block means [6], however the dimensionality of all these solutions is problematic due to the partitioning of the video frames that these approaches perform in order to create the fingerprints.

In this work we propose a video fingerprint generation system as well as its associated database structure. Furthermore, the structure of the perceptual hash allows to significantly reduce the amount of storage required for the hash values as well as to decrease the search time in the database.

The paper is organized as follows: some background is given in section II. The related work is covered in section III. Section IV describes in detail our proposed fingerprinting system. Experimental results and comparisons with prior art are shown in Section V. Finally, Section VI concludes the paper and points out some directions for future work.

II. BACKGROUND

Movies, or motion pictures, pose some interesting obstacles that need to be properly addressed in the digital domain. The amount of information present in full high-definition movies is so large that easily becomes a problem in areas such as information extraction and processing, data storage and the subsequent search within the stored information.

While information extraction can be as straightforward as decoding a movie, the subsequent processing cannot only be a simple, instantaneous, analysis of characteristics of each frame. Data needs to be processed as a whole so that more meaningful information can be obtained from a specific movie, or part of it. In particular, the perceptual analysis of the data must produce results that are independent of the format on which the data has been encoded or compressed. That is, rather than a binary analysis of the data, a perceptual analysis of it is necessary. For

instance, five different files corresponding to a 30-second scene from a movie encoded with MPEG-2, MPEG-4, VP-6 and VC-1 using different values of width, height and frame rate must originate exactly the same unique fingerprint. Thus, that fingerprint is intrinsically associated to the perceptual content of the scene of interest and not to the binary format in which it is presented or stored. Although in recent years the development of data storage has been enormous, the improvement of quality of motion pictures have gone even further. Storing, for analysis, many copies of a movie in its original raw format may involve an unnecessary cost and negatively impact the performance of a whole system. If, on the other hand, perceptual indexes can be stored in strategic places a considerable amount of storage can be saved and searches for audiovisual content can run much faster.

Moreover, searching for information following a continuous time-based approach is very inefficient. Efficient algorithms need to get rid of lots of irrelevant information in a first stage and only then start searching for similar content. The increasing number of content creators and holders need a solution to efficiently search, analyze and identify audiovisual contents stored in several formats that are spread across the storage system of several third-party networks. This is another field where the utilization of perceptual fingerprints has a great potential.

It is therefore essential to provide tools that allow a fast and efficient binary-independent perceptual search of information that cannot be organized as a simple liaison of metadata fields such as facts, keywords or occurrences. New video search engines and data storage systems can take a lot of advantage from perceptual fingerprinting.

III. RELATED WORK

Several video fingerprinting algorithms that work at the pixel level have been proposed. Working directly with pixels is, nowadays, computationally feasible and accurate. However, those solutions do not address the magnitude of the resources needed to build such a system and little analysis is provided in order to use the proposed video fingerprinting solution in practical scenarios.

Many solutions proposed in previous papers use no real databases but temporarily store the contents in RAM. Others provide no effective way to index the video contents in an effective and scalable manner, which can lead to significant gains in system resources. Additionally, recent approaches [1][2] rely on a heavy fingerprint extraction process, which generally consists of video frame partitioning, frame-rate conversion and image resizing.

Out of the analyzed solutions, the proposals based on CGO [1] are deemed the most interesting, incorporating some unaddressed practical points, while providing a challenging algorithm. For instance, the previous recent approaches give little or no insight on database indexing schemas or resources, as well as no information on the expected timings for a real fingerprint analysis. This is the reason for selecting this technique as our comparative benchmark.

IV. PROPOSED METHOD

Several characteristics can be analyzed in movies. While a plethora of information is readily available, our goal has been to use the least possible information in a way that guarantees an excellent performance, while using YUV based luma and color spaces. The use of this color space is widely spread over all the profiles of the current H.264 video format. YUV is a format that provides the luma (black and white contrast component of a frame) separated from chrominance (color components of a frame). The usage of RGB color spaces can be equally supported, as both luma and chrominance can be calculated from RGB chromatic components, however the usage of this color space is not addressed in this work.

In sum, out of the large set of characteristics and statistics that can be extracted from a movie, we have selected the luma in each frame as a base for our algorithm. The luma provides us enough information to uniquely identify a video and is very robust to distortions, especially when dealing with chromatic alterations, as the luma component is independent of the remaining color components.

The following figure depicts the overall mechanics of our system.

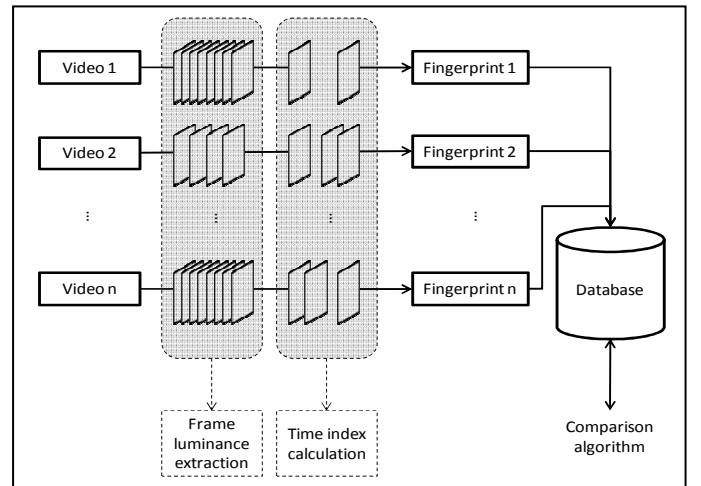


Figure 1. Overview of the proposed system

Figure 1 depicts how the videos are ingested into the system, on the left-hand side, followed by a frame by frame extraction of information. The next stage will calculate the time indexes used for the fingerprint, taking into consideration the video frame-rate. The resulting information is finally stored in a database, used for subsequent comparisons. Each of these processes will be further explained in the following subsections.

A. Data extraction

The Luma associated to a time period corresponds to the average Luma per frame calculated during such period T as shown in Eq. 1:

$$M_T = \frac{1}{w \times h \times N} \sum_{f=0}^{N-1} \sum_{x=0}^w \sum_{y=0}^h L[x, y, f] \quad (1)$$

For the movie M , the Luma L of each pixel (x,y) inside each frame f is averaged in time – over N frames within period T - and space – over $w \times h$ pixels.

Other works [1,2] segment or partition every frame in order to have better results, by assigning different fingerprints to each partition. This increases the dimensionality of the fingerprints, as well as the storage needed in the database. Our proposal uses the whole frame, with no segmentation.

B. Database indexing

The database indexing relies on an inherent aspect of the measured Luma. After an exhaustive analysis of large sets of movies, we found that Luma values are similar for a contiguous, and variable, amount of time. Thus, in order to reduce storage footprint we aggregate information from frames whose Luma falls into a range of values defined by $[A + t, A - t]$ where A is the average Luma of a time period (as shown in Eq.1) and t corresponds to an adjustable threshold. This is what we define as *database threshold*. When the Luma value goes beyond the maximum or minimum range value a new Luma index, that is, a new reference in time for the start of the current period of information, is created until all the information is processed for that movie. This allows the system to save a significant storage space with respect to the per-frame characteristics of the movie. This process is what we define as *information clustering*, i.e. storing clusters of similar values, with respect to the selected threshold.

Figure 1 depicts a typical Luma-value clustering process along the time in which two different indexing periods can be easily identified. Both periods are automatically detected by the proposed system. The left side of figure 2 shows the original measurements of the Luma values of a video file. On the right side, the black vertical bars depict where the indexes are detected, with the threshold-bounded luma box in grey.

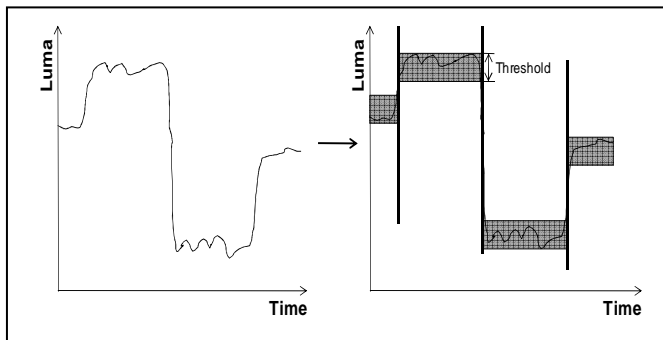


Figure 2. Detection of indexes

The threshold value heavily influences the amount of storage used by our proposed system.. A smaller threshold aggregates fewer Luma values and generates more indexes that in turn may compromise the system accuracy. The amount of storage needed also increases. Likewise, a large threshold aggregates more Luma values and thus it saves a large amount of space in the database at the expense of turning the stored information less valuable for video fingerprinting purposes due to an insufficient number of significant Luma changes detected.

Hence, the selected threshold becomes extremely important for the performance of the whole system.

Figure 3 shows how that information - taken from the index detection algorithm previously described – can be used to build the Luma vs. time graph on the right-hand side. When having the information stored in a database, depicted in the left-hand side, it is easy to reconstruct the information, while saving a large amount of space in the database. Instead of storing Luma information about every second or frame, only the time-and-luma-based indexes are stored.

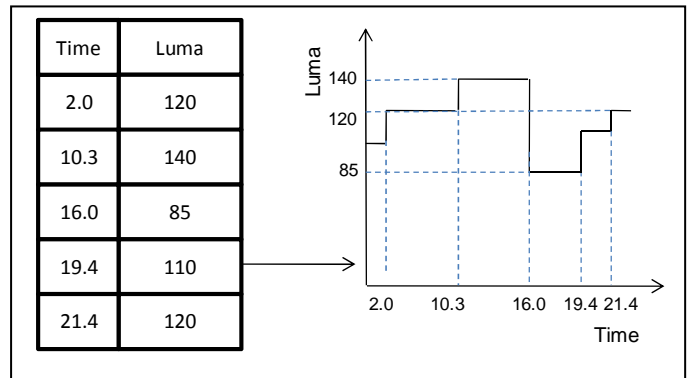


Figure 3. Information clustering algorithm

Additional to the information from each movie, a look-up table for the values of Luma is used. This table allows to quickly search for video content that has a specific pattern of sequential Luma values. The main purpose of this table is to accelerate the initial screening of the movies to analyze by discarding much of the unrelated content, as explained in the search algorithm on next section.

C. Search algorithm

The search algorithm consists of several levels or layers of filtering. By providing a hierarchical identification structure, we can achieve faster searches and, therefore, use fewer resources. Using such an approach it is possible to discard movies that are clearly not related to our source video, or clip, and focus the effort of the search on videos that have a higher probability of being a match to the searched video sequence.

The initial filtering consists of calculating the clustered Luma information of a given video clip or scene, to be found, calculate its time indexes and associated Luma values. Then the algorithm will proceed to perform two analyses. Firstly, search the look-up tables for movies with indexes comprising a similar sequence of Luma values. Secondly, analyze the time indexes of the selected movies and consider for further analysis only the movies that have a similar number and time distribution of indexes. That information is extremely effective in an initial phase, since the time indexes and their values of Luma provide a simple, computationally effective, way to discard unrelated movies, while retaining a high level of uniqueness. It is, therefore, possible to reduce the time wasted searching movies that are easily identified as not being a match.

Next step consists on analyzing all the videos that were previously found to be possible matches of our source video

clip. In this step, different correlation approaches are used. After trying out several of them, we have finally selected the Tanimoto correlation factor [7], defined in Eq. 2, due to its great efficiency in a large number of trial movie sets.

$$T(A, B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B} \quad (2)$$

The Tanimoto correlation compares vectors of values and calculates the probability of those vectors to be similar. In more detail, the vectors that are used in this similarity algorithm are the vectors containing the time indexes and associated Luma values from both the fingerprinting database contents and the input video clip. Initially, the time index correlation calculation is performed in periods of n seconds, being n the total number of seconds from the ingested clip, or the sequence to search.

However, not every index from the source video will be compared to every index of the video in the database. Instead, if a threshold is not met from that comparison, the algorithm will jump to the next index of that movie that is calculated as a possible match and a new comparison will be initiated. Allowing the algorithm to jump to the next relevant section makes the algorithm faster, depending on the threshold used. High gains are achievable at the expense of an increased number of False Positives; that is, videos in the database that appear as a matching result but not correspond to the input video clip. Subsequently, if the correlation of Luma vectors from both the input video clip and the database movie is high, a thorough examination is done, further comparing every second within the current index and the next one. The subsequent and final step is an analysis of the Euclidean distance of the vectors, defined by Eq. 3.

$$D = \sqrt{\sum_{i=0}^n (A_i - B_i)^2} \quad (3)$$

Those vectors are the same that were previously used in the Tanimoto correlation algorithm. However, this time, the distance between those vectors is calculated and the lower the distance, the higher the probability that those vectors are the same and, therefore, that the videos are a match. The purpose of this final analysis is to aid the previous Tanimoto correlation to get rid of both possible False Negative and False Positive cases.

V. RESULTS

To test our proposed algorithm, we compare it with the CGO [1] algorithm. The 210 targeted movies were downloaded from popular UGC websites, amounting to over 700 minutes of video. The movies feature a broad range of resolutions (from VGA (640x480) to FullHD (1920x1080)), lengths (from 1 minute to 60 minutes) and frame-rates (15fps to 30fps). The variety of the videos allows us to make the tests in a large diversity of genres like documentaries, music clips, movie trailers, animations and static image clips generated by users.

The parameters used for the CGO are the same defined by the authors: each frame was partitioned into 8 parts (4 columns and 2 rows), a P_{FA} of 0.03 was chosen and the search sequence

is made of 100 consecutive frames (10 seconds). The parameters for our algorithm are set to provide a clearer comparison with CGO. Therefore, the searched video sequence length is also 10 seconds. In order to provide a better insight of the influence the Luma value clustering can have on the system, a varying database threshold, between 1 and 30, is analyzed.

Our proposed algorithm, named SUJ, was tested by searching a fragment of each movie against itself and the remaining 209 videos in the database. Each test will, in the end, have 44100 comparisons.

The False Positive probability was fixed to $P_{FP}=0.03$ for these experiments. The goal of these experiments is to minimize the number of False Negatives, therefore we can tolerate a certain amount of False Positives. A probability of False Positives $P_{FP}=0.03$ was deemed to meet our requirements.

The hardware specifications of the PC used for the experiments were an AMD dual-core running at 2.3 Ghz with 3GB memory. All tests were ran using a single core, however. The platform is built to allow, in the future, to scale and the amount of tasks can easily be divided in order to distribute the processing power more evenly. For instance, nothing prevents our platform to split a video analysis into segments and use all the available resources to speed up both initial analysis and subsequent comparisons by means of parallel or distributed processing techniques.

Finally, our implementation does not include yet any major optimizations. In fact, important part of our platform was developed using Ruby 1.8 for the comparison results.

A. Analysis of database results

Table I presents the comparison between our proposed solution and the CGO-based solution. In our proposed solution the needed storage for the fingerprints of all 210 movies never exceeds 5Mbytes, while the CGO solution needs over 100Mbytes, i.e. our solution utilizes less than 5% the storage required by CGO.

TABLE I. DATABASE RESULTS

SUJ		CGO
DB threshold	Size (Kbytes)	Size (Kbytes)
1	4,944	107,900
2	3,390	
3	2,611	
5	1,783	
10	1,020	
15	678	
20	512	
25	412	
30	348	

Moreover, in Table I we also present several thresholds used in our proposal for the database. Those thresholds are used in the information clustering algorithm, when deciding what degree of Luma variance can be allowed until a new index is created. Lower values of this threshold mean that lower variances of Luma will create a richer set of indexes for a movie, causing the size of the database to grow. On the other hand, high thresholds allow to reduce the size of the database.

The implications of such thresholds will be further analyzed in the following sub-sections.

B. Analysis of timing results

This sub-section will present the results obtained when the time performance of the presented solutions is analyzed.

TABLE II. INITIAL IMPORT MEASUREMENTS

	SUJ	CGO
Time (minutes)	181	4,538

As explained in Section III, the two compared solutions have large differences in the initial extraction of fingerprints of movies which are later stored in a database. More specifically, the CGO-based solution needs to have several calculations done, such as frame partitioning, conversion and the CGO calculations. Those calculations, as presented in Table II are very expensive in terms of CPU consumption, and completely shatter any possibilities of using CGO in a real-time environment, with an initial fingerprint extraction time of over 75 hours for the 776 minutes of High Definition (1280x720p) content. On the other hand, our SUJ scheme took 3 hours to extract the fingerprint of the same 776 minutes of video, i.e. 4 times faster than real-time. That means our scheme is able to support real-time video transmission system even running in a single threaded process on a commodity PC.

Table III presents the time spent in the search of videos for both mechanisms. Here, a section of each one of the 210 videos was looked up in the database. The results were then averaged for all the 210 results obtained.

TABLE III. LOOK UP TIME RESULTS

DB threshold	SUJ	CGO
	Avg. Time (s)	Avg. Time (s)
1	74.16	92.41
2	28.38	
3	20.48	
5	13.27	
10	8.84	
15	8.66	
20	10.1	
25	12.42	
30	15.11	

The timing results from our SUJ scheme are lower than CGO, with a minimum average of 8.66 seconds for the time of one lookup averaged over the search within 210 movies. It is also relevant to look into the DB threshold column. A low threshold will make the number of indexes grow and the lookup time increase. As our threshold is increased the lookups become faster. However, a too large threshold will make the lookup time grow, due to the fact that valuable information is discarded when using large thresholds, making the algorithm analyze sections of videos that are not a real match in later stages. An example of this behavior is depicted in Figure 4.

In Figure 4 the search times are initially high, as expected, due to the high number of indexes created for low threshold values. With an increase of the threshold value, the average

search times converge to lower values up to a database threshold of 15.

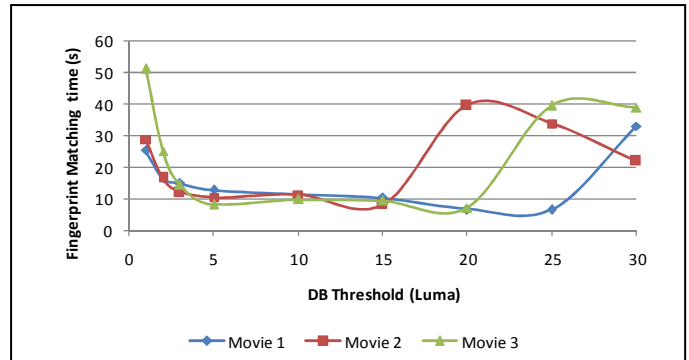


Figure 4. Example of fingerprint timings for 3 movies

However, after this threshold the values become higher and unreliable. This is generally seen in every movie, with minor variations of the threshold or timing measurements. This trend shows how important is to tune the threshold value in order to achieve good performances.

C. Analysis of robustness results

A video fingerprinting platform needs to give accurate results. That accuracy can be expressed by the amount of videos reported as found, but being mismatches (False Positives) or by the number of times a video is known to be in a database but the algorithm is not able to find it (False Negatives) [8]. Out of the two, our solution considers that False Negatives are more undesirable than False Positives.

For the SUJ approach, Table IV shows the average number of false positives and negatives for the search of every 10-second scene in our test dataset with 210 movies.

TABLE IV. ROBUSTNESS RESULTS

DB threshold	SUJ		CGO	
	False Positives	False Negatives	False Positives	False Negatives
1	8	0	211	1
2	12	0		
3	9	0		
5	11	0		
10	11	3		
15	8	8		
20	9	6		
25	11	14		
30	9	18		

For the SUJ approach, Table IV shows the average number of false positives and negatives for the search of every 10-second scene in our test dataset with 210 movies. From the results it can be seen that our solution can effectively find all the matches of every searched 10-second scene within the whole set comprising 776 minutes - or 46,380 seconds- of content in four of the analyzed database thresholds (thresholds 1, 2, 3 and 5). The total number of comparisons performed has been 21,506,406, as a result of all possible 4,638 x 4,637 comparisons of 10-second scenes across the whole test dataset of 210 movies. Hence, it can be seen that the number of false

positives obtained out of the whole dataset is proportionally very low.

As the database threshold increases, the number of false negatives also increases due to loss of information. The clustering algorithm will have less indexes and group larger clusters of Luma, leading to zones with no unique variations of Luma that can identify only small segments of videos.

In contrast, the CGO algorithm presents a larger amount of false positives. While having some false positives would be tolerated, the amount of false positives attained by CGO is quite large in comparison to our scheme even though there is much more processing involved in obtaining the fingerprints. The number of similar movies obtained that were, in fact, not related to the movie ingested for search would provide a high number of false reports.

D. Analysis of video distortion results

This section will present further analysis on the robustness of the analyzed algorithms, using a DB threshold of 5 for the SUJ algorithm.

In order to fully test these algorithms in a variety of situations, some distortions were applied to the videos, re-indexing them in the database. The distortions comprise changes on the frame size, lossy compression, change of frame-rate and rotation of the videos. The resize changes consist on resizing all movies, including Full HD movies to 352x288 pixels (CIF) and 176x220 pixels (QCIF); the lossy compression recompressed the movies at 250kbps; frame-rate change converted the movies to 15 and 5 frames-per-second (fps) movies; rotation consisted on rotating the movies 1, 2 and 3 degrees clockwise.

These tests are also reproduced by related work, for the same effect. Table V transcribes the results obtained.

TABLE V. VIDEO DISTORTION RESULTS

	SUJ P_{FN}	CGO P_{FN}
Resize to CIF	0.000413	0.001963
Resize to QCIF	0.000331	0.001632
Lossy compression	0.000124	0.00188
Frame-rate change to 15 fps	0.000579	0.001983
Frame-rate change to 5 fps	0.001074	0.001612
Rotation of 1 degree	0.000331	0.001715
Rotation of 2 degrees	0.000413	0.001736
Rotation of 3 degrees	0.000455	0.001653

From this table, it is clear that the SUJ platform outperforms the CGO algorithm, constantly presenting a lower Probability of False Negatives (P_{FN}) with the same, fixed, Probability of False Positives (P_{FP}) of 0.03 described earlier.

The lower frame-rate of 5 fps has the worst performances in the SUJ algorithm, however it maintains a lower probability than its CGO counterpart. All the remaining distortion tests show a performance gain of 3 to 4 times, in comparison to CGO.

Moreover, the behavior of both approaches is studied in Figure 5, by varying the algorithm thresholds and plotting them in the following RoC curves.

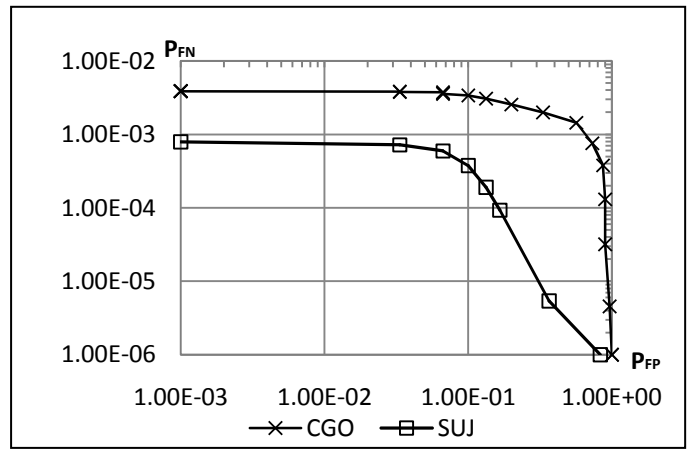


Figure 5. Example of fingerprint timings for 3 movies

As depicted, the P_{FN} and P_{FP} probabilities are lower for the SUJ algorithm, proving the robustness of the solution.

VI. SUMMARY AND FUTURE WORK

In this paper, we propose a novel video fingerprinting method based on Tanimoto correlation. We also propose an indexing method based on luma and time values, which greatly reduces the amount of database information needed to be stored and accelerates the search for fingerprints of videos. The results show that our solution can achieve a large improvement over the CGO scheme. Moreover, our method can reduce the amount of False Positives and False Negatives, the time spent on fingerprint matching and the amount of initial processing so that it can effectively be incorporated in a real-time, practical, situation. When distortion is applied to the ingested movies, we also outperform the CGO algorithm registering an improvement of a scale of, on average, 4 times in terms of false negatives' probability. Future work will focus on algorithm improvements and further comparisons with other methods.

REFERENCES

- [1] Sunil Lee and Chang D. Yoo, "Robust Video Fingerprinting for Content-Based Video Identification", in *IEEE Trans. Circuits and Systems for Video Technology*, vol. 18, no. 7, pp983-988, July, 2008
- [2] Seungjae Lee, Young Ho Suh, Video fingerprinting based on orientation of luminance centroid, *Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, p.1386-1389, June 28-July 03, 2009, New York, NY, USA
- [3] David G. Lowe, Object Recognition from Local Scale-Invariant Features, *Proceedings of the International Conference on Computer Vision-Volume 2*, p.1150, September 20-25, 1999
- [4] S.C. Cheung and Avidesh Zakhor, "Efficient video similarity measurement with video signature," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 1, pp. 59-74, Jan. 2003.
- [5] Changick Kim and B. Vasudev, "Spatiotemporal sequence matching for efficient video copy detection," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 127-132, Jan. 2005.
- [6] Job Oostveen, Ton Kalker, Jaap Haitsma, Feature Extraction and a Database Strategy for Video Fingerprinting, *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems*, p.117-128, March 11-13, 2002
- [7] T.T. Tanimoto, 1957, IBM Internal Report 17th Nov.
- [8] L.M. James and L.C. David, *Decision and Estimation Theory*. New York; McGraw-Hill, 1978, pp27-38