VIDEO REPLICA DETECTION UTILIZING R-TREES AND FRAME-BASED VOTING

Dimitrios Zotos, Nikolaos Nikolaidis, Ioannis Pitas

Department of Informatics
Aristotle University of Thessaloniki
54124 Thessaloniki, Greece
Email: zotodi@yahoo.gr, nikolaid@aiia.csd.auth.gr, pitas@aiia.csd.auth.gr

ABSTRACT

A novel color-based two-step, coarse-to-fine video replica detection system is proposed in this paper. The first step uses an R-tree in order to perform a coarse selection of the database (original) videos that potentially match the query video. A training procedure that utilizes attacked versions of the database videos and aims at achieving robustness to attacks is being used. A frame-based voting procedure is also involved. A refinement step that processes the set of videos returned by the first step in order to select the final matching video (if any) follows. The performance of the system has been evaluated on a database of short videos with good results.

Keywords— Replica detection, copy detection, fingerprinting

1. INTRODUCTION

Advances in the area of multimedia content distribution have resulted in a number of issues that require immediate attention. Valuable digital artworks can be reproduced and distributed arbitrarily, sometimes without any control by their owners. Identification of replicated digital data is considered important for a number of applications such as digital rights management, content-aware networks, multimedia management and organization, content filtering, etc. Among the various types of multimedia content, videos are a particularly valuable asset and their replica detection will be the focus of this manuscript. Video replica detection, also referred as replica recognition, near-replica detection, perceptual or robust hashing [1], [2], content-based copy detection [3], and fingerprinting [4], [5] aims at uniquely identifying all videos that have been reproduced from an original video through intentional or unintentional manipulations. The type and severity of the manipulations that should be successfully handled by a replica detection system depend on the target application.

Although the problem formulation as described above, bears many similarities with content based video retrieval (CBVR), significant differences do exist. This is because the notion of similarity is considerably different in replica detection and in general purpose content-based video retrieval. Furthermore, if replica detection is to be used in a forensic framework, it should be able to operate in a hostile environment and thus robustness to malicious manipulations, which is not central to content based image retrieval, should be taken into account. Finally, a replica detection system should be able to return no matches in response to a query whereas in most cases a CBVR system always returns one or more matching videos.

Several video fingerprinting or replica detection techniques have been proposed in the literature [6], [4], [7], [8], [9], [10], [11], [12]. In this paper, a novel approach for implementing a video replica detection system operating upon a database of stored video originals is proposed. The system can be queried with a certain video and decide whether this video is a replica of a stored original or not. The novelty of the system stems from the fact that video similarity is dealt as a classification problem that employs an appropriate training scheme in order to increase the system robustness and achieve a high rate of correct replica detection. The training samples are selected based on the types of attacks that the system is designed to handle and they drive both the process of indexing the original videos, as well as the construction of robust classification functions. More specifically, videos are represented by color feature vectors. Then a multidimensional indexing structure based on R-trees [13], [14], [15] is implemented. R-trees are an extension of B-trees to more than two dimensions and are considered very efficient for indexing high-dimensional spaces. More specifically, an R-tree is a height-balanced tree with index records in its leaf nodes (containing pointers to data objects). The efficiency of the indexing structure in reducing retrieval time and producing accurate and robust results depends directly on the selection of optimal "hyper-bounding boxes" for use in the R-tree. For selecting these "bounding boxes", we introduce an attack-oriented training strategy that aims at modeling potential attacks that the system is designed to encounter. In the implementation described in this paper, scaling (resizing), additive Gaussian noise and MPEG-4 compression (which are some of the most frequently encountered video manipulations) are considered. The R-tree returns a set of videos that are candidates for being the original of the query video. This is achieved by a voting scheme where each frame from the query video casts a vote to a video in the database. Subsequently, a refinement step that utilizes distances between frame color histograms is applied on the videos returned by the R-tree in order to reach

978-1-4244-7493-6/10/$26.00

the final decision.

The rest of the paper is organized as follows. Section 2 describes the proposed approach. Section 3 provides the results obtained from the experimental performance evaluation of this approach. Conclusions are drawn in Section 4.

2. METHOD DESCRIPTION

2.1. Video description using color histograms

The proposed approach utilizes the color histograms of video frames in order to describe a video. More precisely, it makes use of the MacBeth palette [16]. This palette, also known as MacBeth Color Checker is widely used in the fields of photography and video for assessing the color rendering accuracy of imaging devices. The palette consists of 24 colors selected to emulate common natural colors such as skin colors, foliage and sky, in addition to additive and subtractive primaries and six shades of gray.

For each video that is to be inserted in the database, we evaluate and use as video descriptors the histograms of its frames with respect to the MacBeth palette. In order to speed up calculations, the histogram of every fifth frame is used. This results in N_F histograms. In more detail, let $\mathbf{h_{ij}} = [h_{ij}^R, h_{ij}^G, h_{ij}^B]$ be the color of pixel (i, j) in the RGB color space and $\mathbf{M_k} = [M_k^R, M_k^G, M_k^B]$ be the $k-th$ color in the MacBeth palette then $\mathbf{h_{ij}}$ is assigned to $\mathbf{M_k}$, if:

$$||\mathbf{M_k} - \mathbf{h_{ij}}||_2 < T_0 \quad (1)$$

where $||\cdot||_2$ is the Euclidean distance and T_0 a threshold selected so that RGB colors are assigned to a single palette color. Obviously, RGB colors that are not close enough to a palette color are not assigned to some color and the corresponding pixels are ignored. Afterwards, the 24-bin histogram of the frame is evaluated.

The three attacks mentioned before, namely additive Gaussian noise, scaling and MPEG-4 compression are applied on each of the N_V videos that are to be inserted in the database, thus resulting in 3 attacked versions of each video. For each original video and each attacked version, we extract a set of color histograms as mentioned above. Thus for each video we extract a total of $N_{CH} = 4N_F$ histograms where N_F is the number of frames used from this video.

These histograms are subsequently used in order to build/train the R-Tree structure that is used for storing information for the original videos and for retrieving a set of original videos in response to a query. This procedure will be described in the next subsections and is schematically represented in Figure 1.

2.2. Training the R-tree structure

Let $\mathbf{T}^r$ be the $24 \times N_{CH}$ matrix that corresponds to the r-th video ($r = 1...N_V$), whose columns contain the N_{CH} color histograms of the original video and its attacked versions. $\mathbf{T}^r$ is used to evaluate the boundaries of a hyper bounding box (BB) or polytope that encloses all histograms for this video in the 24-dimensional space. More specifically, the values that determine the boundaries for the $i-th$ dimension of the BB are evaluated as:

$$MINBE_i^r = min_j(\mathbf{T}_{ij}^r), \quad MAXBE_i^r = max_j(\mathbf{T}_{ij}^r)$$
$$i = 1, ..., 24, \quad j = 1, ..., N_{CH}, \quad r = 1, ..., N_V \quad (2)$$

where $\mathbf{T}_{ij}^r$ is the i, j element of $\mathbf{T}^r$. By doing so, each of the N_{CH} color histograms for the r-th video is enclosed in the hyper bounding box defined by $MINBE_i^r$ and $MAXBE_i^r$. These bounding boxes are then used to index the videos in the R-Tree.

2.3. Querying the system

Querying the system with a test video in order to decide whether it is a replica of a video in the database or not, is a two - step procedure that will be described below.

2.3.1. Coarse response

When a query/unknown video enters the system, N_Q frames are selected and their color histograms are evaluated as explained in subsection 2.1. Each histogram is inserted into the R-Tree and all the BBs (videos) that enclose it are found. Let $[C_1, ..., C_{24}]$ be the histogram values of a certain frame. In order for the query frame to be contained in the bounding box that corresponds to the $r - th$ video, the following inequalities should hold:

$$MINBE_i^r < C_i < MAXBE_i^r \quad i = 1, ..., 24 \quad (3)$$

In this case, the corresponding video receives one vote. Obviously, if the frame's color histogram belongs to more than one BBs, due to BB overlaps, then all the corresponding videos will receive a vote. The same procedure is applied for all N_Q frames of the query video. Thus, the maximum number of votes that one database video can receive equals the number of query frames N_Q. Videos that receive at least $a\%$ of the votes cast by the frames of the query video are selected for further processing. We chose $a = 90$ for our experiments.

The above procedure aims at a first, quick selection of videos that are close to the query one. The subset of videos returned by this step is utilized in the second, refinement step in order to reach the final replica detection decision.

2.3.2. Refinement

Let W be the subset of the database videos obtained from the first step. If V_i is one of the videos in W ($i = 1, ..., N_W$, where N_W the number of videos in W), then for every one of the N_Q selected frames of the query video we calculate the minimum of the L_1 distances between its histogram and all the N_{CH} color histograms extracted from the V_i video, i.e., the histograms of all selected frames of the database video and its attacked versions. In more detail, if $\mathbf{Q}_j$, $j = 1, ..., N_Q$ is the histogram of the $j - th$ frame of the query video, we evaluate the distance:

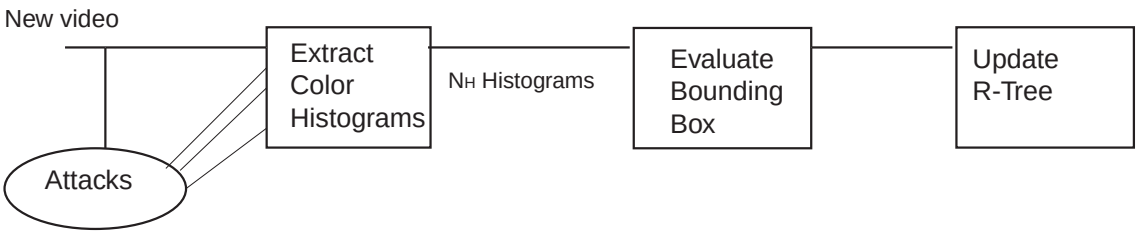

Fig. 1. Video database creation and indexing procedure.

$$D_{ij} = min_k ||\mathbf{Q}_j - \mathbf{V}_i(k)||_1 \quad (4)$$

where $\mathbf{V}_i(k)$ is the $k-th$ histogram of video i. Consequently, the total distance of the query video from the $i-th$ video in W is evaluated as:

$$D_i = \sum_{j=0}^{N_Q-1} (D_{ij})^2 \quad (5)$$

The final winner is the video in W with the minimum total distance D_k from the query video.

$$k = argmin\ D_i\ ,\ i = 1, ..., N_W \quad (6)$$

This distance is subsequently compared against an experimentally evaluated threshold T_1 in order to reach the final decision:

- The query video matches the $k-th$ video if $D^k < T_1$
- The query video matches no video if $D^k > T_1$

A diagram of the query handling procedure is presented in Figure 2.

3. EXPERIMENTAL RESULTS

The proposed system was tested on a database of 589 short videos collected from the Internet, mainly from the popular YouTube website. These videos are of relatively poor quality, have a size of 320x240 pixels, 25 fps frame rate and on average they are no longer than 5 minutes (7500 frames). The videos differ considerably in content due to their random choice.

When a video enters the database we apply the aforementioned attacks and we extract all the necessary color histograms as mentioned in Section 2.1. Then, we evaluate the corresponding bounding box and insert it into the R-Tree. The specific attacks used in the experimental evaluation were: a) uniform scaling with a scaling factor of 0.8 in each dimension, b) XviD MPEG-4 compression with a target quantizer of value 4.00 and c) additive Gaussian noise as implemented in the Adobe Premiere software package with 15% noise.

In order to test the system, we create 4 sets of query videos. The first three sets, A, B, and C were used to test the system performance, when queried with videos that belong to the database or with modified versions of these videos. Two types of errors are expected in this case: missclassification (MC) error i.e., the percentage of query videos that were classified to a wrong original video in the database, and false rejection (FR) error which is the percentage of query videos that were erroneously tagged as not matching any video in the database. The last set, D, consisted of videos that did not belong to the database. In this case, the performance is measured in terms of false acceptance (FA) error i.e., the percentage of query videos that were erroneously tagged as matching a video in the database. For all query videos only a subsampled version (e.g. one out of N_S frames) is used for querying. Thus, for a query video consisting of N frames $N_Q = \frac{N}{N_S}$. In our experiments we chose $N_S = 15$. The procedures and results for each query set are described below.

Query set A: 100 videos were selected from the database and the three attacks with the same parameters as in the training phase were applied on each of them. Thus, this query set consists of 400 videos. Both the missclassification rate and the false rejection rate in this case was 0%.

Query set B: In this case, the 100 selected database videos were modified by the same three attacks but, this time, different attack parameters from the ones used in training were used. In more detail, 20% additive Gaussian noise, MPEG-4 compression with XviD codec (target quantizer: 6.00) and scaling with a scale factor of 0.6 were applied on the selected videos. The fact that the missclassification rate and also the false rejection rate was 0% shows that the system is robust for the attacks it has been trained for, despite the fact that different attack parameters were used.

Query set C: In this case, the 100 selected videos were modified by combinations of more than one of the three attacks. In more detail, four subsets (with 200 videos each) were created. The first subset (C_{ns}) consisted of videos degraded by 18% additive Gaussian noise and scaled with a scale factor of 0.5. Both the missclassification rate and the false rejection rate were 0% . The second subcategory (C_{nm}) consisted of videos that have been attacked by 18% additive Gaussian noise and MPEG-4 compression using the XviD codec with a target quantizer value of 5.00. In this case the missclassification rate was 0% and the false rejection rate was 1.5%. The third subset (C_{ms}) consisted of videos modified by MPEG compression and scaling using the parameters mentioned above. In this case the missclassification rate was 0% and the false rejection rate was 0.5%. Finally, the fourth subset (C_{nms}) consisted of videos that have been attacked by all three afore mentioned attacks. The miss-

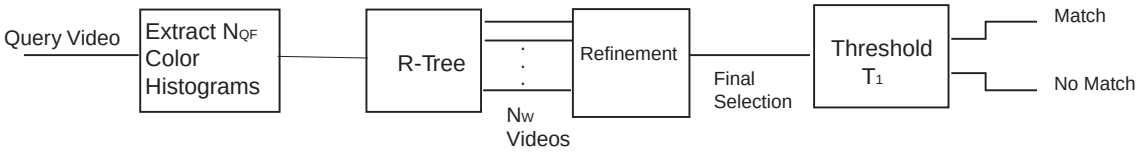

Fig. 2. Schematic representation of the video query handling.

Table 1. Error rates for the four query categories.

Category	Number of Videos	MC	FR	FA
A	400	0%	0%	-
B	300	0%	0%	-
C_{ns}	200	0%	0%	-
C_{nm}	200	0%	1.5%	-
C_{ms}	200	0%	0.5%	-
C_{nms}	200	0%	2.5%	-
D	300	-	-	3.3%

classification rate was found to be 0% and the false rejection rate was 2.5%.

Query set D: 300 videos that did not belong to the database were used in this case. The false acceptance rate was found to be 3.3%.

The above results are summarized in Table 1, verifying that the proposed method achieves good performance.

Regarding the computational complexity of the proposed method, the total time required to answer a query is:

$$t_{ans} = t_{CH} + t_{Rtree} + t_{refine} \quad (7)$$

where t_{CH} is the time required to evaluate the N_{QF} color histograms, t_{refine} is the time needed to calculate all ($N_{CH} * N_{QF} * N_W$) L_1 distances and find the required minimum values in the refinement step and t_{Rtree} is the time required to search in the R-tree structure and perform voting. t_{CH} depends on the length of the query video and is on average 3.5 min for the videos used in the experiments. t_{Rtree} is approximately constant and was found to be 0.1 msec. Finally, t_{refine} depends on N_W which is the number of the videos obtained from the first step. On average $t_{refine} = 3.4$ min. The above figures correspond to experiments conducted on an Intel Core Duo E6750 2.6GHz, 2048MB RAM computer. Thus, the total average query time for the proposed system is 6.9 min per query video.

4. CONCLUSIONS

In this paper a novel color-based two-step, coarse-to-fine video replica detection system has been proposed. The first step performs a coarse selection of the database (original) videos that are candidates for matching the query video. This step uses an R-tree and a training procedure that involves attacked versions of the database videos and aims at achieving robustness to attacks. A frame-based voting procedure is also involved in this step. The next step is a refinement step that processes the set of videos returned by the first step in order to select the final matching video (if any). The performance of the proposed system has been evaluated with short videos collected from the Internet. The experimental results show that the proposed system is robust to common video attacks. It should be noted that the same framework can be also applied upon other types of features, which will be the subject of future research.

5. REFERENCES

[1] C. Y. Hsu and C.S. Lu, "Geometric distortion-resilient image hashing system and its application scalability," in *ACM International Workshop on Multimedia and Security*, Magdeburg, Germany, 2004, pp. 81–92.

[2] A. Swaminathan, Y. Mao, and M. Wu, "Image hashing resilient to geometric and filtering operations," in *IEEE Workshop on Multimedia Signal Processing (MMSP'04)*, Siena, Italy, Sept. 2004, pp. 355–358.

[3] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *IEEE Transactions on Multimedia*,, vol. 9, no. 2, pp. 293–306, 2007.

[4] J. Oostveen, T. Kalker, and J. Haitsma, "Feature extraction and a database strategy for video fingerprinting," in *5th International Conference on Recent Advances in Visual Information Systems (VISUAL '02)*, London, UK, 2002, pp. 117–128.

[5] J. S. Seo, J. Haitsma, T. Kalker, and C. D. Yoo, "Affine transform resilient image fingerprinting," in *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 03)*, April 2003, pp. 61–64.

[6] P. Indyk, G. Iyengar, and N. Shivakumar, "Finding pirated video sequences on the internet," *Tecnical Report, Stanford University*, 1999.

[7] B. Coskun, B. Sankur, and N. Memon, "Spatio-temporal transform-based video hashing," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1190–1208, December 2006.

[8] A. Hampapur and R. Bolle, "Comparison of sequence matching tecniques for video copy detection," in *Conference on Storage and Retrieval for Media Databases*, 2002, pp. 194–201.

[9] J. Law-To, V. Gouet-Brunet, O. Buisson, and N. Boujemaa, "Video copy detection on the internet: The challenges of copyright and multiplicity," in *IEEE International Conference on Multimedia and Expo (ICME 2007)*, July 2007, pp. 2082–2085.

[10] C. Kim and B. Vasudev, "Spatiotemporal sequence matching for efficient video copy detection," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, pp. 127–132, 2005.

[11] S. Lee and C. D. Yoo, "Robust video fingerprinting based on 2D-OPCA of affine covariant regions," in *IEEE International Conference on Image Processing*, 2008.

[12] C. Cotsaces, N. Nikolaidis, and I. Pitas, "Semantic video fingerprinting and retrieval using face information," *Signal Processing: Image Communication*, vol. 24, no. 7, pp. 598–613, August 2009.

[13] V. Gaede and O. Gunther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, no. 2, pp. 170–231, 1998.

[14] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos, and N. Y. Theodoridis, *R-trees: Theory and Applications*, Springer-Verlag, 2005.

[15] A. Gutmann, "R-trees: a dynamic index structure for spatial searching," in *ACM International Conference on Management and Data (SIGMOD'88)*, Siena, Italy, 1988, pp. 47–57.

[16] C.S McCamy, H. Marcus, and J.G Davindson, "A color-rendition chart," *Journal of Applied Photographic Engineering*, vol. 2, pp. 95–99, 1976.