Content Based Video Retrieval System for Distorted Video Queries

Boris Tseytlin[a], Ilya Makarov[a]

[a]National Research University, Higher School of Economics, Moscow, Russia

Abstract

We consider the task of content-based video retrieval (CBVR) given a query video, which is expected to match if it is a distorted short subsequence of a reference video from a database. In this paper, we present a CBVR system architecture that is both robust and scalable. We use a modified rHash frame fingerprint generation method. It is both, extremely robust to distortions and fast to compute. We utilize the Faiss library, developed by Facebook Research, to index fingerprint binary vectors. The VCDB dataset is used for benchmarking.

Keywords

information retrieval, multimedia database, content-based video retrieval, rHash

1. Introduction

In this paper we consider the task of searching a database of reference videos given a query video. A query video is expected to match if it is a distorted short subsequence of a reference video (also referred to as *partial copy*). In literature the task is known as content-based video retrieval (CBVR) [1], content-based video search [2], near-duplicate video matching [3] or content based video copy detection [4]. In all of these formulations the task involves searching for videos that contain a subsequence similar to the query video. To avoid confusion, we will refer to the task as CBVR further on.

The CBVR community benefited a lot from the advancements in content-based image retrieval (CBIR) [5], which is focused on searching a database of images for partial copies. It's common to use frame features as the basis for CBVR systems. A CBVR problem can be approached as a CBIR problem, however using temporal information present in video sequences have proven to provide better results as seen in [1].

The task of content-based video retrieval is challenging because of various distortions and content variations that query videos might be subject to. These include noise, compression artifacts, rotation, framerate alternation, logo attacks, frame removal, scale and lightning changes. Scalability is also a major concern, because videos contain thousands of frames, which leads to long query times and large memory requirements. Most approaches to CBVR require calculating the distance from the query video to all subsequences of all reference videos. Clearly, this approach becomes infeasible for real-world applications as database volume grows.

Robust hashing, also known as video fingerprinting, is a popular approach for partial copy detection. It involves generating a content-based signature assigned to video frames, and matching query video signatures to reference video signatures. In robust hashing, the hash function should be stable in regards to visual distortions.

MACSPro'20: Modeling and Analysis of Complex Systems and Processes, October 22–24, 2020, Venice, Italy
EMAIL: batseytlin@edu.hse.ru (B. Tseytlin); iamakarov@hse.ru, corresponding author (I. Makarov)
URL: http://hse.ru/en/staff/iamakarov (I. Makarov)
ORCID: 0000-0002-3308-8825 (I. Makarov)

In this paper we present a CBVR system architecture that is both robust and scalable. We use a modified rHash [4] frame fingerprint generation method. It's both extremely robust to distortions and fast to compute. We utilize the Faiss library [6], developed by Facebook Research, to index fingerprint binary vectors. This is done to speed up database queries. The VCDB dataset is used for benchmarking.

The rest of the paper is organized as follows. A brief overview of the related studies is presented in Section 2. In Section 3, the system architecture is presented. The video file preprocessing steps, the fingerprint generation approach, database indexing and searching strategies are described. In Section 4 we present our evaluation results on the VCDB dataset. Finally, Section 5 presents our conclusions.

2. Related work

Feature extraction is a central part of any CBVR system. Pairwise comparison of frames using pixel-by-pixel measures is inefficient. It's necessary to find a compact representation of video sequences in lower dimensionsionality. This is known as feature extraction or feature generation. A feature generation approach aims to strike a balance between matching accuracy, fast searching and compact database size.

Video feature generation approaches can be classified into categories:

- color-based features
- temporal features
- spatial features
- deep learning based features
- fusion of different types of features

An overview of image feature extraction techniques is provided in [7]. Color-based features are usually derived from histograms of pixel intensities or colors. Color histograms are sensitive to specific color spaces, which makes them sensitive to video formats. They are also highly affected by noise, saturation changes. However, a local region color histogram feature can be highly resistant to distortions. In that approach, the image is divided into non-overlapping regions, a color histogram is calculated for each region, and then histogram bin counts are concatenated together to form a feature vector.

Temporal features are based on time and frame differences. They are extracted over time from a video sequence. In [8] a temporal based sequence matching method was proposed. In this method each frame is divided into a grid of average pixel intensity values, and grids are stored in a ranking sequence. This provided a global and local description of temporal variation.

Spatial features are generated from individual frames. In video fingerprinting, the feature vector of an image is a binary hash. Average hash (aHash), proposed in [9] is one of the most commonly used fingerprinting methods. This method is simple and fast to compute. It uses the mean intensity value of a grayscale image to binarize it by thresholding. The output is a short binary sequence representing an image. [4] proposed an extension to this method called rHash. The extension is simple yet effective - each frame is divided into non-overlapping blocks, block mean intensity values are calculated, and each block is binarized by thresholding on the median of block mean values. The result is a short binary sequence of predefined size. This hash has proven to be more robust than aHash. It's also easy to compute. [10] proposed DCT hash, which is effective, but requires complex transformations.

Image descriptors utilizing SIFT, SURF and other key point extraction algorithms are often used to generate spatial features. Roth et al. [11] proposed a spatial feature based on SURF point counts. They divided each frame into non-overlapping blocks and counted the amount of key points in each frame. The resulting key point counts were used as frame descriptions. SURF points are known to be extremely robust to image manipulations, including mirroring and rotation. In [12] the idea was expanded by adding a temporal component. SURF point counts of each frame were used to generate a block ranking matrix. Block ranks produced a vector that described the whole video sequence.

Usually generating features from each video frame is redudant. Commonly, key frames and frame rate downsampling are used for speeding up feature extraction. In 2010 [13] proposed the concept of temporally informative representative images (TIRI). This method aims to aggregate sequences of frames together without loosing much information. It has been used in many studies ever since [4] [12] [14].

There have been multiple attempts at using neural networks to extract features for video retrieval tasks. A deep learning approach achieved state-of-the-art results on the VCDB dataset [15]. [16] used a VGG-16 architecture network, together with dimensionality reduction via PCA, to generate features for video copy detection. [17] describes a few cases of "Siamese twin" networks being used for feature generation. An approach using rHash [4] managed to obtain results comparable to deep-learning generated features, whilist keeping the computation costs low.

The scalability issue of CBVR was studied thoroughly. A CBVR system should be able to retrieve similar video sequences quickly while operating on a very large database of signatures. Inverted file indexes are commonly used in practice [4] [1] [17]. Faiss [6] by Facebook Research and Annoy [18] by Spotify are tools that provide in-memory inverted file indexes. In Locality Sensitive Hashing (LSH) higher dimensional data is projected into a lower dimensionality representation using random projections [19]. [20] proposed an extension to LSH for cases of searching for set queries, which is more efficient when dealing with feature sets, such as SURF descriptors.

The overview of CBVR datasets conducted by [17] shows that most works on topic benchmark their work on either TRECVID [21] or VCDB [15] datasets. VCDB is relatively new, however it's the only dataset containing real partial copies. Other datasets include only simulated copies. This made VCDB the standard dataset for benchmarking CBVR systems.

3. System overview

The basic architecture of the proposed CBVR system is presented on Figure 1. The diagram shows two scenarios - the index generation flow indicated by red arrows, when the database is populated with reference videos, and the querying flow indicated by blue arrows.

First, videos are transcoded to a predefined size and framerate, and converted to grayscale, to make the system robust to framerate alternations. Then TIRI frames are generated. A modified rHash called Quadrant rHash is obtained from each TIRI frame. In case of index generation, the obtained hash vectors are stored in a binary vector index using Faiss. In case of querying, the obtained hash vectors are matched against the database. In the final step, fingerprint matching results are used to make a decision on whether the query video matches any reference videos.

3.1. TIRI extraction

Temporally informative representative images (TIRI) [13] are obtained as weighted average of sequential frames. First, a sequence of frames $f_1, f_2, \ldots, f_N$ extracted from a video is divided into non-overlapping blocks of length T. For each block a TIRI frame f' is computed as a weighted average

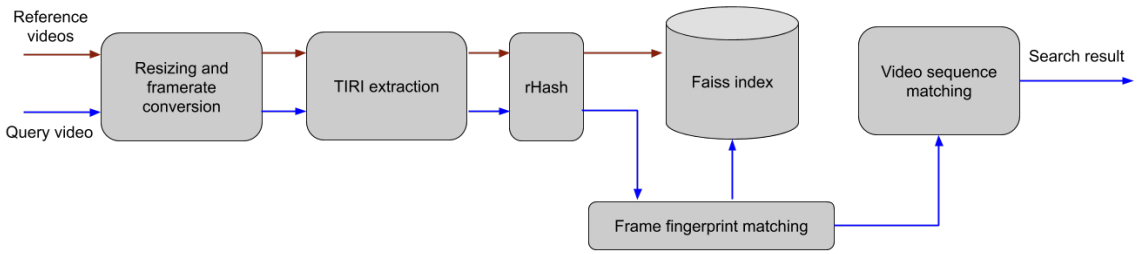

Figure 1: CBVR system overview

Figure 2: Five frames (left) and the obtained TIRI frame (right).

of pixel intensities of frames in block. There are many ways to pick weights w_k. We have chosen exponential weights $w_k = \gamma^k$ with $\gamma = 1.65$ as it has been proven to be effective by previous research [13] [4].

$$f'_{i,j} = \frac{1}{\sum_{k=m}^{m+T} w_k} \sum_{k=m}^{m+T} w_k \cdot f_{k,i,j} = \frac{1}{\sum_{k=m}^{m+T} \gamma^k} \sum_{k=m}^{m+T} \gamma^k \cdot f_{k,i,j}$$

Where m is the frame block start index, i, j are pixel positions.

An example is provided on Figure 2.

3.2. Quadrant rHash

rHash has been proposed in [4] as a more robust extension to the popular aHash. In rHash, each frame is divided into non-overlapping blocks. A sequence of block mean intensity values $M = (m_1, m_2, \ldots, m_N)$

is obtained. The sequence is then binarized using the following rule:

$$h(i) = \begin{cases} 1, & \text{if } m_i \geq median(M) \\ 0, & \text{otherwise} \end{cases}$$

We propose a further simple extension to rHash. In Quadrant rHash, the mean values sequence is divided into four non-overlapping blocks M_1, M_2, M_3, M_4. The sequence is then binarized using median values of corresponding blocks M_k. The binarization rule becomes:

$$h(i) = \begin{cases} 1, & \text{if } m_i \geq median(M_k), \text{where } M_k \in \{M_1, M_2, M_3, M_4\} \text{ s.t. } m_i \in M_k \\ 0, & \text{otherwise} \end{cases}$$

An example is presented on Figure 3.

Quadrant rHash vectors can be efficiently compared using Hamming distance. Computing them involves only simple operations like computing the mean, so it's extremely fast.

3.3. FAISS index

Faiss is a library for efficient approximate k-nearest neighboors search. It's a good choice for scalable systems because it has been developed with scale in mind [6]. It's possible to utilize it's GPU parallelization capabilities for speeding up search.

3.4. Video retrieval strategy

Given query hash vectors, extracted from a query video, a matching reference video needs to be found. We compared two strategies for video matching.

Majority vote (MV). Majority vote is a simple matching strategy. The hypotheses behind this method is formulated this way: if most frames of a query video are closest to a reference video, then the query video is most likely a partial copy of that reference video. This method involves the following steps:

1. For each query frame Q, find the tuple (f, d, V), where f is a reference video frame with minimal Hamming distance from Q, d is the distance value, and V is the id of the video f belongs to. This is a very fast operation with Faiss.
2. Threshold the tuples with a distance threshold value d_t. For each tuple (f, d, V), if $d > d_t$, replace V with a special value -1, indicating lack of match.
3. Construct a sequence of retrieved video ids: $V_1, V_2, \ldots, V_M$. Where V_i is the video id retrieved for Q_i, M is the query length.
4. Return the most frequent item in $V_1, V_2, \ldots, V_M$. It can either be a video id, or -1. -1 indicates, that the query video didn't match any videos in database.

Longest sequence with candidate elimination (LS). This is a slightly more complex matching strategy, similar to the approach used in [4]. It makes use of temporal information. It involves the following steps:

1. For each query frame obtain $n = 10$ closest frames in database. Construct a sequence of tuples:

   $$(Q_1, f_1, d_1, t_1, V_1), (Q_1, f_2, d_2, t_2, V_2), \ldots, (Q_M, f_{n\times M}, d_{n\times M}, t_{n\times M}, V_{n\times M})$$

   . Here each tuple represents a potential matched frame, Q_i is the index of a query frame, f is reference video frame, d is the Hamming distance value, t is the time in seconds at which f occurs in video, V is the video id. Threshold tuples on a distance threshold d_t.

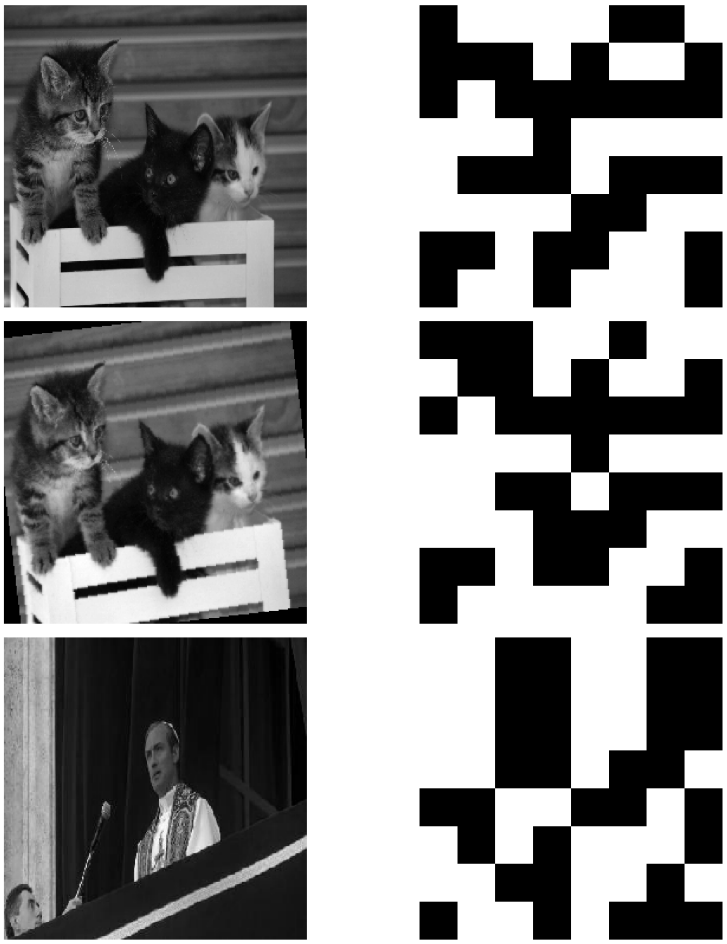

Figure 3: Images and their Quadrant rHash vectors vizualized.

The first row contains an image (left) and it's Quadrant rHash (right). The second row contains the same image, but with noise, rotation, rescaling, blur applied. The third row contains a different image, included for comparison. It's evident that hashes of the first two images are visually similar, whilist a different image produces a different hash.

2. Consider each two tuples $(Q_i, f_i, d_i, t_i, V_i)$ and $(Q_j, f_j, d_j, t_j, V_j)$. Eliminate j-th tuple if: $Q_j > Q_i$ and $t_j < t_i$. We remove those potential matches that are chronologically impossible: if query frame j occurs after query frame i, matched frame f_j must occur after f_i.
3. For each unique video id V_i present in the sequence of potential matches, count the amount of tuples that contain V_i. Return the video id with the largest count of matches.

4. Experiments and results

We evaluated the performance of our method on the VCDB core dataset. VCDB is an annotated dataset of real partial copies. It contains 519 videos, about 27 hours of video data in total. Partial copies in this dataset are subject to compression artifacts, framerate alternations, picture-in-picture, logo attacks, inserted frames, frame removal, noise, blur and more. It's a very challenging dataset for

Matching strategy	F-score	Mean query latency, ms
MV	0.715	15
LS	0.761	158
baseline	0.757	44

Table 1
Evaluation results

a CBVR system.

We have taken 9 videos as the reference videos, 106 partial copies of them as query videos and the rest 413 videos as distraction queries. Our experiments have shown that best results are achieved with resizing each video to 200x200, downsampling the framerate to 5 frames per second, extracting TIRI from every $T = 5$ frames. Similar conclusions were made in [4]. We extracted Quadrant rHashes of dimensionality 100. The distance threshold value was chosen as $d_t = 20$ based on experiments.

The traditional F-score measure was used for evaluation:

$$p = \frac{tp}{tp + fp}$$

$$r = \frac{tp}{tp + fn}$$

$$F = 2\frac{p \times r}{p + r}$$

All experiments were ran using Python 3.6, on an Ubuntu 18 Linux operation system, 2.50GHz Intel Core i5-7200U CPU.

The evaluation results are presented in Table 1. We compare obtained F-score results to [22], which used traditional rHash and temporal network matching approaches. We compare obtained query times to [4] which used a special inverted file-based index.

The longest sequence matching method achieves better performance at the cost of longer query processing times. Surprisingly, majority voting achieves a result similar to more complex methods. Our system is able to conduct very fast searches. LS provides more fine-grained results at the cost of searching times.

5. Conclusions

Content-based video retrieval is a challenging task due to many disturbances that query videos might be subject to. We propose a system architecture based on a modified image fingerprinting measure called Quadrant rHash that is very fast to compute and robust against image distortions. Temporally informative representative images (TIRI) are used during video preprocessing. In our approach, the Faiss library is used to build an index on obtained binary vectors. The proposed system was evaluated on the well-known VCDB core dataset. Two different video sequence matching strategies were evaluated. Experimental results have shown a slight performance increase when compared to similar approaches.

References

[1] S. A. Bhat, O. V. Sardessai, P. P. Kunde, S. S. Shirodkar, Overview of existing content based video retrieval systems, International Journal of Advanced Engineering and Global Technology 2 (2014) 476–483.

[2] E. Esen, S. Ozkan, I. Atil, Large-scale video search with efficient temporal voting structure, arXiv preprint arXiv:1607.07160 (2016).

[3] X. Wu, C.-W. Ngo, A. G. Hauptmann, H.-K. Tan, Real-time near-duplicate elimination for web video search with content and context, IEEE Transactions on Multimedia 11 (2009) 196–207.

[4] M. Liu, L.-M. Po, Y. A. U. Rehman, X. Xu, Y. Li, L. Feng, Video copy detection by conducting fast searching of inverted files, Multimedia Tools and Applications 78 (2019) 10601–10624.

[5] G. Amato, P. Bolettieri, F. Carrara, F. Falchi, C. Gennaro, Large-scale image retrieval with elasticsearch, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, ACM, 2018, pp. 925–928.

[6] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus, arXiv preprint arXiv:1702.08734 (2017).

[7] T. Deselaers, D. Keysers, H. Ney, Features for image retrieval: an experimental comparison, Information retrieval 11 (2008) 77–107.

[8] L. Chen, F. Stentiford, Video sequence matching based on temporal ordinal measurement, Pattern Recognition Letters 29 (2008) 1824–1831.

[9] B. Yang, F. Gu, X. Niu, Block mean value based image perceptual hashing, in: 2006 International Conference on Intelligent Information Hiding and Multimedia, IEEE, 2006, pp. 167–172.

[10] M. Malekesmaeili, M. Fatourechi, R. Ward, A robust and fast video copy detection system using content-based fingerprinting, IEEE Transactions on Information Forensics and Security 6 (2011) 213–226. doi:10.1109/TIFS.2010.2097593.

[11] G. Roth, R. Laganière, P. Lambert, I. Lakhmiri, T. Janati, A simple but effective approach to video copy detection, in: 2010 Canadian Conference on Computer and Robot Vision, IEEE, 2010, pp. 63–70.

[12] R. C. Harvey, M. Hefeeda, Spatio-temporal video copy detection, in: Proceedings of the 3rd Multimedia Systems Conference, ACM, 2012, pp. 35–46.

[13] M. M. Esmaeili, R. K. Ward, Robust video hashing based on temporally informative representative images, in: 2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE), IEEE, 2010, pp. 179–180.

[14] M. Steinebach, H. Liu, Y. Yannikos, Forbild: Efficient robust image hashing, in: Media Watermarking, Security, and Forensics 2012, volume 8303, International Society for Optics and Photonics, 2012, p. 83030O.

[15] Y.-G. Jiang, Y. Jiang, J. Wang, Vcdb: a large-scale database for partial copy detection in videos, in: European conference on computer vision, Springer, 2014, pp. 357–371.

[16] L. Wang, Y. Bao, H. Li, X. Fan, Z. Luo, Compact cnn based video representation for efficient video copy detection, in: International conference on multimedia modeling, Springer, 2017, pp. 576–587.

[17] Y. Jiang, J. Wang, Partial copy detection in videos: A benchmark and an evaluation of popular methods, IEEE Transactions on Big Data 2 (2016) 32–42. doi:10.1109/TBDATA.2016.2530714.

[18] Spotify, ANNOY library, https://github.com/spotify/annoy, 2017. Accessed: 2017-08-01.

[19] M. Bawa, T. Condie, P. Ganesan, Lsh forest: self-tuning indexes for similarity search, in: Proceedings of the 14th international conference on World Wide Web, ACM, 2005, pp. 651–660.

[20] P. Nagarkar, K. S. Candan, Pslsh: An index structure for efficient execution of set queries in high-dimensional spaces, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 477–486.

[21] G. Awad, P. Over, W. Kraaij, Content-based video copy detection benchmarking at trecvid, ACM Trans. Inf. Syst. 32 (2014) 14:1–14:40. URL: http://doi.acm.org/10.1145/2629531. doi:10.1145/2629531.

[22] L. Mengyang, L.-M. Po, Z. Chang, W. Y. Yuen, H.-K. Cheung, H. Peter, H.-T. Luk, K.-W. Lau, Content-based video copy detection using binary object fingerprints, in: 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), IEEE, 2018, pp. 1–6.