Efficient and Robust Detection of Duplicate Videos in a Large Database

Anindya Sarkar, *Student Member, IEEE,* Vishwarkarma Singh, Pratim Ghosh, *Student Member, IEEE,* B. S. Manjunath, *Fellow, IEEE* and Ambuj Singh, *Senior Member, IEEE*

Abstract

We present an efficient and accurate method for duplicate video detection in a large database using video fingerprints. We have empirically chosen the Color Layout Descriptor, a compact and robust frame-based descriptor, to create fingerprints which are further encoded by vector quantization. We propose a new non-metric distance measure to find the similarity between the query and a database video fingerprint and experimentally show its superior performance over other distance measures for accurate duplicate detection. Efficient search can not be performed for high dimensional data using a non-metric distance measure with existing indexing techniques. Therefore, we develop novel search algorithms based on pre-computed distances and new dataset pruning techniques yielding practical retrieval times. We perform experiments with a database of 38000 videos, worth 1600 hours of content. For individual queries with an average duration of 60 sec (about 50% of the average database video length), the duplicate video is retrieved in 0.032 sec, on Intel Xeon with CPU 2.33GHz, with a very high accuracy of 97.5%.

Index Terms

Video fingerprinting, duplicate detection, color layout descriptor, non-metric distance, vector quantization.

I. Introduction

COPYRIGHT infringements and data piracy have recently become serious concerns for the ever growing online video repositories. Videos on commercial sites e.g., *www.youtube.com, www.metacafe.com*, are mainly textually tagged. These tags are of little help in monitoring the content and preventing copyright infringements. Approaches based on *content-based copy detection (CBCD)* and *watermarking* have been used to detect such infringements [20], [27]. The watermarking approach tests for the presence of a certain watermark in a video to decide if it is copyrighted. The other approach (CBCD) finds the duplicate by comparing the fingerprint of the

Anindya Sarkar, Pratim Ghosh and B.S. Manjunath are with the Vision and Research Laboratory, Department of Electrical and Computer Engineering, while Vishwarkarma Singh and Ambuj Singh are with the Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA, 93106 USA.
E-mail: {anindya, pratim, manj}@ece.ucsb.edu, {vsingh, ambuj}@cs.ucsb.edu.

query video with the fingerprints of the copyrighted videos. A fingerprint is a compact signature of a video which is robust to the modifications of the individual frames and discriminative enough to distinguish between videos. The noise robustness of the watermarking schemes is not ensured in general [27], whereas the features used for fingerprinting generally ensure that the *best match in the signature space remains mostly unchanged even after various noise attacks*. Hence, the fingerprinting approach has been more successful.

We define a "duplicate" video as the one *consisting entirely of a subset of the frames in the original video - the individual frames may be further modified and their temporal order varied*. The assumption that a duplicate video contains frames only from a single video has been used in various copy detection works, e.g., [32], [39], [42]. In [42], it is shown that for a set of 24 queries searched in YouTube, Google Video and Yahoo Video, 27% of the returned relevant videos are duplicates. In [7], each web video in the database is reported to have an average of five similar copies - the database consisted of 45000 clips worth 1800 hours of content. Also, for some popular queries to the Yahoo video search engine, there were two or three duplicates among the top ten retrievals [32].

In Fig. 1, we present the block diagram of our duplicate video detection system. The relevant symbols are explained in Table I. The database videos are referred to as "model" videos in the paper. Given a model video V_i, the decoded frames are sub-sampled at a factor of 5 to obtain T_i frames and a p dimensional feature is extracted per frame. Thus, a model video V_i is transformed into a $T_i \times p$ matrix Z^i. We empirically observed in Sec. III that the Color Layout Descriptor (CLD) [24] achieved higher detection accuracy than other candidate features. To summarize Z^i, we perform k-means based clustering and store the cluster centroids $\{X^i_j\}_{j=1}^{F_i}$ as its fingerprint. The number of clusters F_i is fixed at a certain fraction of T_i, e.g., a fingerprint size of 5x means that $F_i = (5/100)T_i$. Therefore, the fingerprint size varies with the video length. K-means based clustering generally produces compact video signatures which are comparable to those generated by sophisticated summarization techniques as discussed in [36]. In [2], we have compared different methods for keyframe selection for creating the compact video signatures.

The duplicate detection task is to retrieve the best matching model video fingerprint for a given query fingerprint. The model-to-query distance is computed using a new non-metric distance measure between the fingerprints as discussed in Sec. IV. We also empirically show that our distance measure results in significantly higher detection accuracy than traditional distance measures (L_1, partial Hausdorff distance [18], [19], Jaccard [9] and cosine distances). We design access methods for fast and accurate retrieval of duplicate videos. The challenge in developing such an access method is two-fold. *Firstly, indexing using such distances has not been well-studied till date* - the recently proposed distance based hashing [3] performs dataset pruning for arbitrary distances. Secondly, video fingerprints are generally of high dimension and varying length. *Current indexing structures (M-tree [10], R-tree [17], kd tree [4]) are not efficient for high-dimensional data.*

To perform efficient search, we propose a two phase procedure. The first phase is a coarse search to return

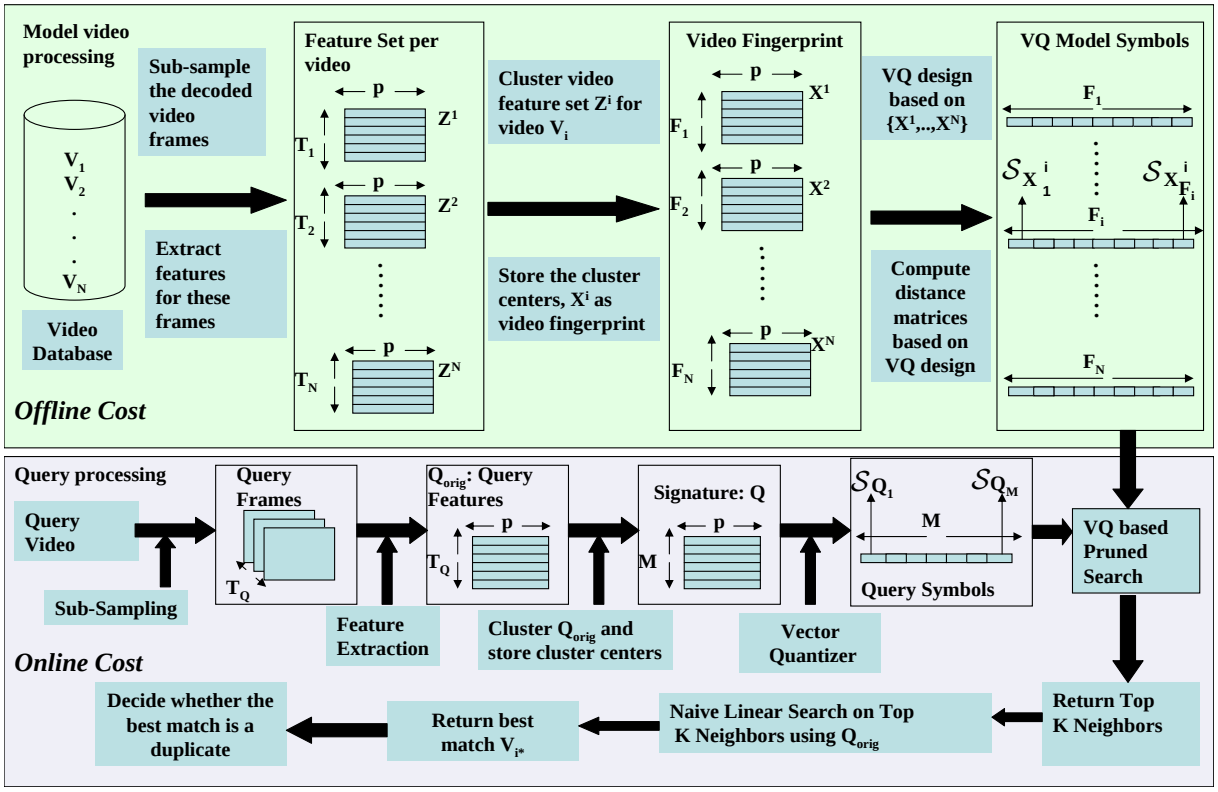


Fig. 1. Block diagram of the proposed duplicate detection framework - the symbols used are explained in Table I.

the top-K nearest neighbors (NN) which is the focus of the paper. We perform vector quantization (VQ) on the individual vectors in the model (or query) fingerprint X^i (or Q) using a codebook of size U (= 8192) to generate a sparse histogram-based signature $\overrightarrow{x_i}$ (or $\overrightarrow{q}$). This is discussed in Sec. V-B. Coarse search is performed in the VQ-based signature space. Various techniques proposed to improve the search are the use of pre-computed information between VQ symbols, partial distance based pruning, and the dataset pruning as discussed in Sec. V. The second phase uses the unquantized features (X^i) for the top-K NN videos to find the best matching video V_{i^*}. The final module (Sec. VII) decides whether the query is indeed a duplicate derived from V_{i^*}.

The computational cost for the retrieval of the best matching model video has two parts (Fig. 1).

1) **Offline cost** (model related) - consists of the un-quantized model fingerprint generation, VQ design and encoding of the model signatures, and computation and storing of appropriate distance matrices.

2) **Online cost** (query related) - the query video is decoded, sub-sampled, keyframes are identified, and features are computed per keyframe - these constitute the query **pre-processing cost**. In this paper, we report the **query time** - this comprises of the time needed to obtain k-means based compact signatures, perform VQ-based encoding on the signatures to obtain sparse histogram-based representations, compute the relevant lookup tables, and then perform two-stage search to return the best matched model video.

The paper is organized as follows. Sec. II contains some relevant previous work. Feature selection for fingerprint

TABLE I
Glossary of Notations

Notation	Definition
N	number of database videos
V_i	i^{th} model video in the dataset
V_{i^*}	the best matched model video for a given query
p	dimension of the feature vector computed per video frame
$Z^i \in \mathbb{R}^{T_i \times p},\ T_i$	Z^i is the feature vector matrix of V_i, where V_i has T_i frames after temporal sub-sampling
$X^i \in \mathbb{R}^{F_i \times p},\ F_i$	X^i is the k-means based signature of V_i, which has F_i keyframes
X^i_j	j^{th} vector of video fingerprint X^i
U	size of the vector quantizer (VQ) codebook used to encode the model video and query video signatures
$Q_{orig} \in \mathbb{R}^{T_Q \times p}$	query signature created after sub-sampling, where T_Q refers to the number of sub-sampled query frames
$Q \in \mathbb{R}^{M \times p}$	keyframe based query fingerprint, where M is the number of query keyframes
C_i	the i^{th} VQ codevector
$\overrightarrow{x_i},\ \overrightarrow{q}$	$\overrightarrow{x_i}$ is VQ based signature of V_i, while $\overrightarrow{q}$ is VQ based query signature
$\mathcal{S}_{X^i_j}$	VQ symbol index to which X^i_j is mapped
$\mathcal{A}$	the set of N "model signature to query signature" distances
$\mathbb{D} \in \mathbb{R}^{U \times U}$	Inter VQ-codevector distance matrix, for L_1 distance between VQ codevectors
$\mathbb{D}^* \in \mathbb{R}^{N \times U}$	Lookup table of shortest distance values from each VQ-based model signature to each VQ codevector
$\mathbb{C}(i)$	the cluster containing the video indices whose VQ-based signatures have the i^{th} dimension as non-zero
$\|\|\overrightarrow{x_1} - \overrightarrow{x_2}\|\|_p$	the L_p norm of the vector $(\overrightarrow{x_1} - \overrightarrow{x_2})$.
$\|E\|$	the cardinality of the set E
ℓ	**fractional query length** = (number of query frames/number of frames in the original model video)

creation is discussed in Sec. III. Sec. IV introduces our proposed distance measure. The various search algorithms, along with the different pruning methods, are presented in Sec. V. The dataset creation for this task is explained in Sec. VI-A. Sec. VI-B contains the experimental results while Sec. VII describes the final decision module which makes the "duplicate/non-duplicate decision".

Main Contributions of the Paper

• We propose a new non-metric distance function for duplicate video detection when the query is a noisy subset of a single model video. It performs better than other conventional distance measures.

• For the VQ-based model signatures retained after dataset pruning, we reduce the search time for the top-K candidates by using suitable pre-computed distance tables and by discarding many non-candidates using just the partially computed distance from these model video signatures to the query.

• We present a dataset pruning approach, based on our distance measure in the space of VQ-encoded signatures, which returns the top-K nearest neighbors (NN) even after pruning. We obtain significantly higher pruning than that provided by distance based hashing [3] methods, trained on our distance function.

In this paper, the terms "signature" and "fingerprint" have been used interchangeably. "Fractional query length" (ℓ in Table I) refers to the fraction of the model video frames that constitute the query. Also, for a VQ of codebook size U, the 1-NN of a certain codevector is the codevector itself.

II. Literature Survey

A good survey for video copy detection methods can be found in [27]. Many schemes use global features (e.g., color histogram computed over the entire video) for a fast initial search for prospective duplicates [42]. Then, keyframe-based features are employed for a more refined search.

Keypoint based features: In an early duplicate detection work by Joly *et al*. [22], the keyframes correspond to extrema in the global intensity of motion. Local interest points are identified per keyframe using the Harris corner detector and local differential descriptors are then computed around each interest point. These descriptors have been subsequently used in other duplicate detection works [20], [21], [26], [27]. In [42], PCA-SIFT features [25] are computed per keyframe on a host of local keypoints obtained using the Hessian-Affine detector [34]. Similar local descriptors are also used in [45], where near-duplicate keyframe (NDK) identification is performed based on matching, filtering and learning of local interest points. A recent system for fast and accurate large-scale video copy detection, the Eff^2 Videntifier [11], uses Eff^2 descriptors [29] from the SIFT family [33]. In [44], a novel measure called Scale-Rotation Invariant Pattern Entropy (SR-PE) is used to identify similar patterns formed by keypoint matching of near-duplicate image pairs. A combination of visual similarity (using global histogram for coarser search and local point based matching for finer search) and temporal alignment is used to evaluate video matching for duplicate detection in [40]. VQ based techniques are used in [8] to build a SIFT-histogram based signature for duplicate detection.

Global Image Features: In some approaches, the duplicate detection problem involves finding the similarity between sets of time-sequential video keyframes. A combination of MPEG-7 features such as the Scalable Color Descriptor, Color Layout Descriptor (CLD) [24] and the Edge Histogram Descriptor (EHD) has been used for video-clip matching [5], using a string-edit distance measure. For image duplicate detection, the Compact Fourier Mellin transform (CFMT) [13] has also been shown to be very effective in [15] and the compactness of the signature makes it suitable for fingerprinting.

Entire Video based Features: The development of "ordinal" features [6] gave rise to very compact signatures which have been used for video sequence matching [35]. Li *et al*. [30] used a binary signature to represent each video, by merging color histogram with ordinal signatures, for video clip matching. Yuan *et al*. [43] also used a similar combination of features for robust similarity search and copy detection. UQLIPS, a recently proposed real-time video clip detection system [39], uses RGB and HSV color histograms as the video features. A localized color histogram based global signature is proposed in [32].

Indexing Methods: Each keyframe is represented by a host of feature points, each having a descriptor. The matching process involves comparison of a large number of interest point pairs which is computationally intensive. Several indexing techniques have been proposed for efficient and faster search. Joly *et al*. [22] use an indexing method based on the Hilbert's space filling curve principle. In [20], the authors propose an improved index structure for video fingerprints, based on Statistical Similarity Search (S^3) where the "statistical query" concept was based on the distribution of the relevant similar fingerprints. A new approximate similarity search technique was proposed in [23] and later used in [21], [26] where the probabilistic selection of regions in the feature space is based on the distribution of the feature distortion. In [45], an index structure LIP-IS is proposed for fast filtering of keypoints under one-to-one symmetric matching. For the Videntifier [11] system, the approximate NN search in the high-dimensional database (of Eff^2 descriptors) is done using the NV-tree [28], an efficient disk-based data structure.

Hash-based Index: The above mentioned indexing methods are generally compared with locality sensitive hashing (LSH) [12], [16], a popular approximate search method for L_2 distances. Since our proposed distance function is non-metric, LSH cannot be used in our setup as the locality sensitive property holds only for metric distances. Instead, we have experimented with the recently proposed distance based hashing (DBH) [3] scheme, which can be used for arbitrary distance measures.

Final Duplicate Confirmation: From the top retrieved candidate, the duplicate detection system has to validate whether the query has indeed been derived from it. The keyframes for a duplicate video can generally be matched with the corresponding frames in the original video using suitable spatio-temporal registration methods. In [21], [27], the approximate NN results are post-processed to compute the most globally similar candidate based on a registration and vote strategy. In [26], Law-To *et al*. use the interest points proposed in [22] for trajectory building along the video sequence. A robust voting algorithm utilizes the trajectory information, spatio-temporal registration, as well as the labels computed during the off-line indexing to make the final retrieval decision. In our duplicate detection system, we have a "distance threshold based" (Sec. VII-A) and a registration-based framework (Sec. VII-B) to determine if the query is actually a duplicate derived from the best-matched model video.

The advantages of our method over other state-of-the-art methods are summarized below.

• In current duplicate detection methods, the query is assumed to contain a large fraction of the original model video frames. Hence, the query signature, computed over the entire video, is assumed to be similar to the model video signature. This assumption, often used as an initial search strategy to discard outliers, does not hold true when the query is only a small fraction (e.g., 5%) of the original video. For such cases, the query frames have to be individually compared with the best matching model frames, as is done by our distance measure. As shown later in Figs. 3 and 7, we observe that our proposed distance measure performs much better than other distances for duplicate detection for shorter queries.

• We develop a set of efficient querying techniques with the proposed distance measure which achieves much better dataset pruning than distance based hash (DBH) - DBH is the state-of-the-art method for querying in non-metric space.

• In [21], [27], the registration step is performed between query frames and other model frames to confirm whether the query is a duplicate derived from the model video. In our distance computation procedure, we also end up computing which model vector serves as the best match for a query vector - this inter-vector correspondence helps in faster identification of the best matching model keyframe for a given query keyframe (discussed in Sec. VII-B). This frame-to-frame correspondence is needed for effective registration.

III. Feature Extraction

Candidate Features

We performed duplicate video detection with various frame based features - CLD, CFMT [13], Localized Color Histogram (LCH) [32] and EHD [41]. The LCH feature divides the image into a certain number of blocks and the 3D color histogram is computed per block. E.g., if each color channel is quantized into 4 levels, the 3D histogram per image block has $4^3 = 64$ levels. If the image is divided into two partitions along each direction, the total LCH feature dimension is $4^3 \times 2^2 = 256$. To study the variation of detection accuracy with signature size, we have considered the LCH feature for dimensions 256 and 32 ($2^2 \times 2^3 = 32$). For the frame-based features, we use our proposed distance measure, which is explained in Sec. IV.

We also considered video features (computed globally, i.e. over the entire video and not per key-frame). One such global feature used is the m-dimensional histogram obtained by mapping each of the 256-dimensional LCH vectors to one of m codebook vectors (this signature creation is proposed in [32]), obtained after k-means clustering of the LCH vectors. We have experimented with $m = 20$, 60, 256 and 8192, and L_2 distance is used. The other global feature is based on a combination of the ordinal and color histograms [43]. Both the ordinal and color histograms are 72-dimensional (24 dimensions along each of the Y, Cb and Cr channels) and the distance measure used is a linear combination of the average of the distance between the ordinal histograms and the minimum of the distance between the color histograms, among all the 3 channels.

Experimental Setup and Performance Comparison

We describe the duplicate detection experiments for feature comparison. We use a database of 1200 video fingerprints and a detection error occurs when the best matched video is not the actual model from which the query was derived. The query is produced using one of various image processing/noise addition methods, discussed in Sec. VI-A. The query length is gradually reduced from 50% to 2.5% of the model video length and the detection error (averaged over all noisy queries) is plotted against the fractional query length (Fig. 2). For our dataset, a fractional query length of 0.05 corresponds, on an average, to 6 sec of video $\approx$ 30 frames, assuming 25 frames/sec

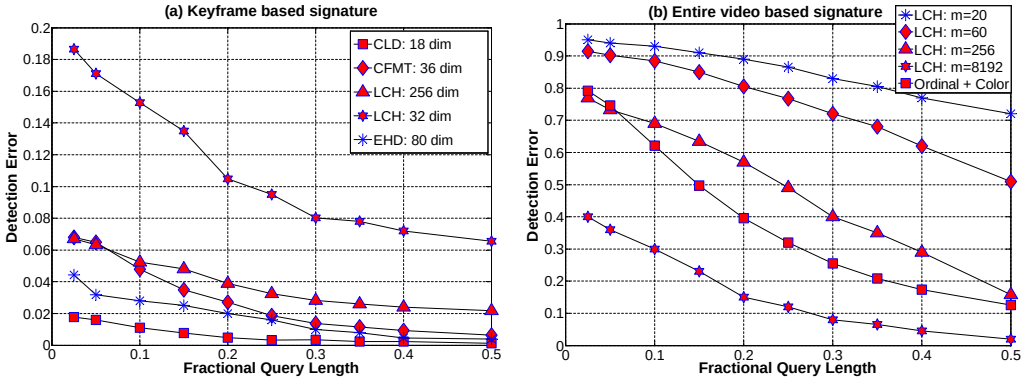


Fig. 2. Comparison of the duplicate video detection error for (a) keyframe based features and (b) entire video based features: the query length is varied from 2.5% to 50% of the actual model video length. The error is averaged over all the query videos generated using noise addition operations, as discussed later in Sec. VI-A. The model fingerprint size used in (a) is 5x.

and a sub-sampling factor of 5. Fig. 2(a) compares frame based features while Fig. 2(b) compares video based features.

In our past work [38], we have shown that the CFMT features perform better as video fingerprints than SIFT features for duplicate detection. Here, it is seen that for duplicate detection, 18-dim CLD performs slightly better than 80-dim EHD, which does better than 36-dim CFMT and 256-dim LCH (Fig. 2(a)). Due to the lower signature dimension and superior detection performance, we choose 18-dim CLD feature per keyframe for fingerprint creation. It is seen that for a short query clip, the original video histogram is often not representative enough for that clip leading to higher detection error, as in Fig. 2(b). Hence, using a global signature with L_2 distance works well only for longer queries.

We briefly describe the CLD feature vector and also provide some intuition as to why it is highly suited for the duplicate detection problem. The CLD signature [24] is obtained by converting the image to a 8×8 image, on averaging, along each (Y/Cb/Cr) channel. The Discrete Cosine Transform (DCT) is computed for each image. The DC and first 5 (in zigzag scan order) AC DCT coefficients for each channel constitute the 18-dimensional CLD feature. The CLD feature is compact and captures the frequency content in a highly coarse representation of the image. As our experimental results suggest, different videos can be distinguished even at this coarse level of representation for the individual frames. Also, due to this coarse representation, *image processing and noise operations, which are global in nature, do not alter the CLD significantly* so as to cause detection errors; thus, the feature is robust enough. Significant cropping or gamma variation can distort the CLD sufficiently to cause errors - a detailed comparison of its robustness to various attacks is presented later in Table VII. Depending on the amount of cropping, the 8×8 image considered for CLD computation can change significantly, thus severely perturbing the CLD feature. Also, significant variations in the image intensity through severe gamma variation can change the frequency content, even for an 8×8 image representation, so as to cause detection errors.

Storage-wise, our system consumes much less memory compared to methods which store key-point based descriptors [11], [21]. The most compact key-point based descriptor is the 20-dim vector proposed in [21] where each dimension is represented by 8 bits and 17 feature vectors are computed per second. The corresponding storage is 10 times that of our system (assuming 18-dimensional CLD features per frame where each dimension is stored as a double, 25 frames/sec, temporal sub-sampling by 5, 5% of the sub-sampled frames being used to create the model signature).

IV. Proposed Distance Measure

Our proposed distance measure to compare a model fingerprint X^i with the query signature Q is denoted by $d(X^i, Q)$ [1] (1). This distance is the sum of the best-matching distance of each vector in Q with all the vectors in X^i. In (1), $\|X_j^i - Q_k\|_1$ refers to the L_1 distance between X_j^i, the j^{th} feature vector of X^i and Q_k, the k^{th} feature vector of Q. Note that $d(\cdot, \cdot)$ is a quasi-distance.

$$d(X^i, Q) = \sum_{k=1}^{M} \left\{ \min_{1 \le j \le F_i} \|X_j^i - Q_k\|_1 \right\} \quad (1)$$

What is the motivation behind this distance function? We assume that each query frame in a duplicate video is a tampered/processed version of a frame in the original model video. Therefore, the summation of the best-matching distance of each vector in Q with all the vectors in the signature for the original video (X^i) will yield a small distance. Hence, the model-to-query distance is small when the query is a (noisy) subset of the original model video. Also, this definition accounts for those cases where the duplicate consists of a reordering of scenes from the original video.

A comparison of distance measures for video copy detection is presented in [18]. Our distance measure is similar to the Hausdorff distance [18], [19]. For our problem, the Hausdorff distance $h(X^i, Q)$ and the partial Hausdorff distance $h_P(X^i, Q)$ are interpreted as:

$$h(X^i, Q) = \max_{1 \le k \le M} \left\{ \min_{1 \le j \le F_i} \|X_j^i - Q_k\|_1 \right\} \quad (2)$$

$$h_P(X^i, Q) = \underbrace{P^{th}\text{largest}}_{1 \le k \le M} \left\{ \min_{1 \le j \le F_i} \|X_j^i - Q_k\|_1 \right\} \quad (3)$$

For image copy detection, the partial Hausdorff distance (3) has been shown to be more robust than the Hausdorff distance (2) in [18]. We compare the performance of $h_P(X^i, Q)$ (3) for varying P, with $d(X^i, Q)$, as shown in Fig. 3, using the same experimental setup as in Sec. III. It is seen that the results using $d(X^i, Q)$ are better - *the improved performance is more evident for shorter queries*.

[1] For ease of understanding, the quasi-distance measure $d(\cdot, \cdot)$ is referred to as a distance function in subsequent discussions.

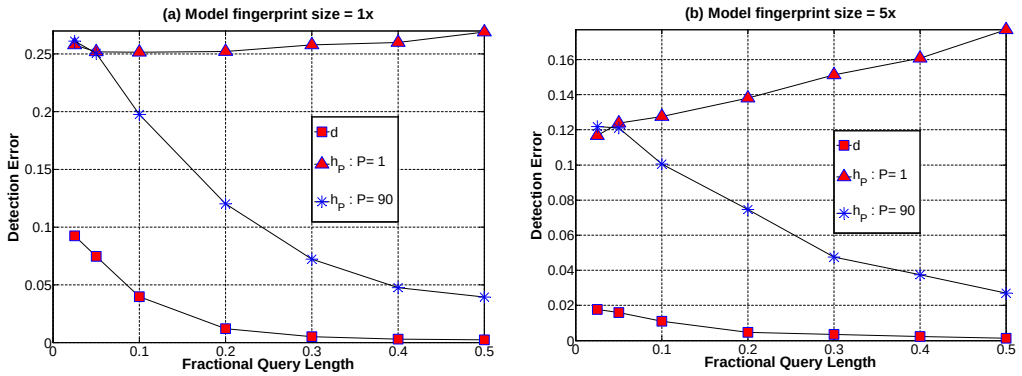


Fig. 3. Comparison of the duplicate video detection error for the proposed distance measure $d(\cdot,\cdot)$ (1) and the Hausdorff distances: here, $(h_p : P = k)$ refers to the partial Hausdorff distance (3) where the k^{th} maximum is considered.

Intuitively, why does our distance measure perform better than the Hausdorff distance? In (2) (or (3)), we first find the "minimum query frame-to-model video" distance for every query frame and then find the maximum (or P^{th} largest) among these distances. *Thus, both $h(X^i, Q)$ and $h_P(X^i, Q)$ effectively depend on a single query frame and model video frame, and errors occur when this query (or model) frame is not representative of the query (or model) video*. In our distance function (1), $d(X^i, Q)$ is computed considering all the "minimum query frame-to-model video" terms and hence, the effect of one (or more) mismatched query feature vector is compensated.

Dynamic time warping (DTW) [37] is commonly used to compare two sequences of arbitrary lengths. The proposed distance function has been compared to DTW in [2], where it is shown that DTW works well only when the query is a continuous portion of the model video and not a collection of disjoint parts. *This is because DTW considers temporal constraints and must match every data point in both the sequences*. Hence, when there is any mismatch between two sequences, DTW takes that into account (thus increasing the effective distance), while the mismatch is safely ignored in our distance formulation.

V. Search Algorithms

In this section, we develop a two-phase approach for fast duplicate retrieval. The proposed distance measure (1) is used in our search algorithms for duplicate detection. First, we discuss a *naive linear search algorithm* in Sec. V-A. Search techniques based on the vector quantized representation of the fingerprints that achieve speedup through suitable lookup tables are discussed in Sec. V-B. Algorithms for further speedup based on dataset pruning are presented in Sec. V-C.

We perform temporal sub-sampling of the query video to get a signature Q_{orig} having T_Q vectors (see Fig. 1 and Table I). The initial coarse search (first pass) uses a smaller query signature Q, having M ($M < T_Q$) vectors. Q consists of the cluster centroids obtained after k-means clustering on Q_{orig}. When $M = (5/100)T_Q$, we refer to the query fingerprint size as 5x. The first pass returns the top-K NN from all the N model videos. The larger query

TABLE II
WE PRESENT THE TIME COMPLEXITY OF THE VARIOUS MODULES INVOLVED IN COMPUTING $\mathcal{A} = \{d(X^i, Q)\}_{i=1}^N$ (4), RETURNING THE TOP-K NN, AND THEN FINDING THE BEST MATCHED VIDEO V_{i^*} FROM THEM. $\bar{F} = \sum_{i=1}^N F_i/N$ DENOTES THE AVERAGE NUMBER OF VECTORS IN A MODEL FINGERPRINT. FOR THE VQ-BASED SCHEMES, THE DISTANCE $d(\cdot,\cdot)$ IS REPLACED BY THE DISTANCE $d_{VQ}(\cdot,\cdot)$ (9), WHILE THE OTHER OPERATIONS INVOLVED REMAIN SIMILAR.

Time	Operation involved	Complexity
T_{11}	computing L_1 distance between vectors X_j^i and Q_k : $\|X_j^i - Q_k\|_1$	$O(p)$
$T_{12} = T_{11}.F_i$	finding the best matched model vector for a query vector : $\min_{1\le j\le F_i} \|X_j^i - Q_k\|_1$	$O(F_i p)$
$T_2 = M.T_{12}$	finding best match for all M frames in Q to compute $d(X^i, Q)$	$O(MF_i p)$
$T_3 = \sum_{i=1}^N T_2$	computing all N model-to-query distances : $\mathcal{A} = \{d(X^i, Q)\}_{i=1}^N$	$O(MN\bar{F}p)$
T_4	use minimum K values from $\mathcal{A}$ to return top-K videos using a priority queue	$O(N \log K)$
T_5	finding V_{i^*} from top-K videos using larger query signature Q_{orig}	$O(T_Q K\bar{F}p + K)$

signature (Q_{orig}) is used for the second pass to obtain the best matched video from these K candidates using a naive linear scan. As the query length decreases, the query keyframes may differ significantly from the keyframes of the actual model video; hence, the first pass needs to return more candidates to ensure that the actual model video is one of them.

A naive approach for the search is to compute all the N model-to-query distances and then find the best match. This set of N distances is denoted by $\mathcal{A}$ (4). We speedup the coarse search by removing various computation steps involved in $\mathcal{A}$. For the purpose of explaining the speedup obtained by various algorithms, we provide the time complexity breakup in Table II.

$$\mathcal{A} = \{d(X^i, Q)\}_{i=1}^N = \left\{ \sum_{k=1}^{M} \left\{ \min_{1\le j\le F_i} \|X_j^i - Q_k\|_1 \right\} \right\}_{i=1}^N \tag{4}$$

A. Naive Linear Search (NLS)

The Naive Linear Search (NLS) algorithm implements the two-pass method without any pruning. In the first pass, it retrieves the top-K candidates based on the smaller query signature Q by performing a full dataset scan using an ascending priority queue L of length K. The priority queue is also used for the other coarse search algorithms in this section to keep track of the top-K NN candidates. The k^{th} entry in L holds the model video index ($L_{k,1}$) and its distance from the query ($L_{k,2}$). A model signature is inserted into L if the size of L is less than K or its distance from the query is smaller than the largest distance in the queue. In the second pass, NLS computes the distance of the K candidates from the larger query signature Q_{orig} so as to find the best matched candidate. The storage needed for all the model signatures = $O(N\bar{F}p)$, where $\bar{F}$ denotes the average number of vectors in a model fingerprint.

B. Vector Quantization and Acceleration Techniques

From Table II, it is observed that time T_{11} can be saved by pre-computing the inter-vector distances. When the feature vectors are vector quantized, an inter-vector distance reduces to an inter-symbol distance, which is fixed once the VQ codevectors are fixed. Hence, we vector quantize the feature vectors and represent the signatures as histograms, whose bins are the VQ symbol indices. For a given VQ, we pre-compute and store the inter-symbol distance matrix in memory.

We now describe the VQ-based signature creation. Using the CLD features extracted from the database video frames, a VQ of size U is constructed using the Linde-Buzo-Gray algorithm [31]. The distance $d(\cdot,\cdot)$ (1) reduces to $d_{VQM}(\cdot,\cdot)$ (5) for the VQ-based framework, where $\mathbb{D}$ is the inter-VQ codevector distance matrix (6). $C_{\mathcal{S}_{X_j^i}}$ refers to the $\mathcal{S}_{X_j^i}{}^{th}$ codevector, i.e. the codevector to which the VQ maps X_j^i.

$$d_{VQM}(X^i,Q)=\sum_{k=1}^{M}\min_{1\le j\le F_i}\left\|C_{\mathcal{S}_{X_j^i}}-C_{\mathcal{S}_{Q_k}}\right\|_1=\sum_{k=1}^{M}\min_{1\le j\le F_i}\mathbb{D}(\mathcal{S}_{X_j^i},\mathcal{S}_{Q_k}) \tag{5}$$

$$\text{where } \mathbb{D}(k_1,k_2)=\|C_{k_1}-C_{k_2}\|_1,\ 1\le k_1,k_2\le U \tag{6}$$

Let $\overrightarrow{q}=[q_1,q_2,\cdots,q_U]$ denote the normalized histogram-based query signature (7) and $\overrightarrow{x_i}=[x_{i,1},x_{i,2},\cdots,x_{i,U}]$ denote the corresponding normalized model signature (8) for video V_i.

$$q_k = |\{j:\mathcal{S}_{Q_j}=k,\ 1\le j\le M\}|/M \tag{7}$$

$$x_{i,k} = |\{j:\mathcal{S}_{X_j^i}=k,\ 1\le j\le F_i\}|/F_i \tag{8}$$

Generally, consecutive video frames are similar; hence, many of them will get mapped to the same VQ codevector while many VQ codevectors may have no representatives (for a large enough U). Let $\{t_1,t_2,\cdots,t_{N_q}\}$ and $\{n_{i,1},n_{i,2},\cdots,n_{i,N_{x_i}}\}$ denote the non-zero dimensions in $\overrightarrow{q}$ and $\overrightarrow{x_i}$, respectively, where N_q and N_{x_i} denote the number of non-zero dimensions in $\overrightarrow{q}$ and $\overrightarrow{x_i}$, respectively.

The distance between the VQ-based signatures $\overrightarrow{x_i}$ and $\overrightarrow{q}$ can be expressed as:

$$d_{VQ}(\overrightarrow{x_i},\overrightarrow{q}) = \sum_{k=1}^{N_q}q_{t_k}.\left\{\min_{1\le j\le N_{x_i}}\mathbb{D}(t_k,n_{i,j})\right\} \tag{9}$$

It can be shown that the distances in (5) and (9) are identical, apart from a constant factor.

$$d_{VQM}(X^i,Q)=M.d_{VQ}(\overrightarrow{x_i},\overrightarrow{q}) \tag{10}$$

The model-to-query distance (9) is same for different model videos if their VQ-based signatures have the same non-zero dimensions. For our database of 38000 videos, the percentage of video pairs (among $\binom{38000}{2}$ pairs) that have the same non-zero indices is merely $3.2\times10^{-4}\%$ [2]. A note about our VQ-based signature - since we discard

the temporal information and are concerned with the relative frequency of occurrence of the various VQ symbols (one symbol per frame), the signature is similar to the "bag-of-words" model commonly used for text analysis and computer vision applications [1].

The distance computation involves considering all possible pairs between the N_q non-zero query dimensions and the N_{x_i} non-zero model dimensions. We propose a technique where the distance computation can be discarded based on a partially computed (not all $N_q.N_{x_i}$ pairs are considered) distance - we call it "Partial Distance based Pruning" (PDP) (Sec. V-B1). We then present two VQ-based techniques (VQLS-A in Sec. V-B2 and VQLS-B in Sec. V-B3) which use different lookup tables, utilize PDP for faster search and significantly outperform the NLS scheme.

1) Partial Distance Based Pruning (PDP): We present a technique (PDP) that reduces time T_3 (Table II) by computing only partially N model-to-query distances in $\mathcal{A}$. This speedup technique is generic enough to be used for both the un-quantized and the VQ-based signatures. We insert a new distance in the priority queue L if it is smaller than the largest distance in the queue ($L_{K,2}$). The logic behind PDP is that if the partially computed model-to-query distance exceeds $L_{K,2}$, the full distance computation is discarded for that model video.

Let $\hat{d}(X^i, Q, k')$ be the distance between X^i and the first k' vectors of Q - this is a partially computed model-to-query distance for $k' < M$. If $\hat{d}(X^i, Q, k')$ exceeds $L_{K,2}$, we discard the model video signature X^i as a potential top-K NN candidate and save time spent on computing $d(X^i, Q)$, its total distance from the query. Though we spend additional time for comparison in each distance computation (comparing $\hat{d}(X^i, Q, k')$ to $L_{K,2}$), we get a substantial reduction in the search time as shown later in Fig. 5(b).

When PDP is used in the un-quantized feature space, we call that method as Pruned Linear Search (PLS). The total storage space required for PLS is also $O(N\bar{F}p)$, like NLS. Since we do not consider all the M vectors of Q in most of the distance computations, we have $m \leq M$ vectors participating, on an average, in the distance computation. Therefore, the time required to compute $\mathcal{A}$, T_3 (Table II) now reduces to $O(mN\bar{F}p)$. The other computational costs are same as that for NLS.

$$\hat{d}(X^i, Q, k') = \sum_{k=1}^{k'} \left\{ \min_{1 \leq j \leq F_i} \|X_j^i - Q_k\|_1 \right\} \tag{11}$$

$$\text{For } k' \leq M,\ \hat{d}(X^i, Q, k') \leq \hat{d}(X^i, Q, M),\ \text{and } \hat{d}(X^i, Q, M) = d(X^i, Q)$$

$$\therefore\ \hat{d}(X^i, Q, k') \geq L_{K,2} \Rightarrow d(X^i, Q) \geq L_{K,2} \tag{12}$$

2) Vector Quantization based Linear Search - Method A (VQLS-A): In VQLS-A, we pre-compute the inter-VQ codevector distance matrix $\mathbb{D}$ (6) and store it in memory. We perform a full search on all the video signatures using $d_{VQ}(\overrightarrow{x_i}, \overrightarrow{q})$ (9) to find the top-K NN signatures - however, it directly looks up for a distance between

two VQ symbols in the matrix $\mathbb{D}$ (e.g., $\mathbb{D}(t_k, n_{i,j})$ in (9)) and hence saves time (T_{11} in Table II) by avoiding the L_1 distance computation. This method also uses PDP for speedup. *PDP in NLS implies searching along a lesser number of query frames. Here, it implies searching along a lesser number of non-zero query dimensions.* Sorting of the non-zero dimensions of the query signature results in improved PDP-based speedup. As in NLS, we maintain an ascending priority queue L.

$$\hat{d}_{VQ}(\overrightarrow{x_i}, \overrightarrow{q}, k') = \sum_{k=1}^{k'} q_{t_k^*}.\left\{\min_{1\leq j\leq N_{x_i}} \mathbb{D}(t_k^*, n_{i,j})\right\} \tag{13}$$

where $q_{t_1^*} \geq q_{t_2^*} \cdots \geq q_{t_{N_q}^*}$ represents the sorted query signature. After considering the first k' non-zero (sorted in descending order) query dimensions, we discard the distance computation if $\hat{d}_{VQ}(\overrightarrow{x_i}, \overrightarrow{q}, k')$ (13) exceeds $L_{K,2}$.

The storage requirement for $\mathbb{D}$ is $O(U^2)$. Let the average number of non-zero dimensions in the VQ-based model signatures be F', where $F' = (\sum_{i=1}^{N} N_{x_i})/N$. We need to encode Q before search which incurs a time of $O(MU)$. Since this algorithm uses a constant time lookup of $O(1)$, the complexity of T_2 is reduced to $O(N_q F')$. The time T_3 to compute all the N model-to-query distances, without PDP, is $O(MU + N_q N F')$. Using PDP, the average number of non-zero query dimensions considered reduces to N_q', where $N_q' < N_q$. The corresponding reduced value of T_3 is $O(MU + N_q' N F')$. The time needed to sort the query dimensions is $O(N_q \log N_q)$, which is small enough compared to $(MU + N_q' N F')$.

3) Vector Quantization based Linear Search - Method B (VQLS-B): This method obtains higher speedup than VQLS-A by directly looking up the distance of a query signature symbol to its nearest symbol in a model video signature (e.g., $\{\min_{1\leq j\leq N_{x_i}} \mathbb{D}(t_k, n_{i,j})\}$ in (9)). Thus, the computations involved in both T_{11} and T_{12} (Table II) can be avoided, hence reducing the time to find a model-to-query distance to $O(N_q)$. We pre-compute a matrix $\mathbb{D}^* \in \mathbb{R}^{N\times U}$ where $\mathbb{D}^*(i,k)$ (15) denotes the minimum distance of a query vector, represented by symbol k after the VQ encoding, to the i^{th} model.

$$d_{VQ}(\overrightarrow{x_i}, \overrightarrow{q}) = \sum_{k=1}^{N_q} q_{t_k}.\mathbb{D}^*(i, t_k), \text{ using (9)} \tag{14}$$

$$\text{where } \mathbb{D}^*(i, t_k) = \min_{1\leq j\leq N_{x_i}} \mathbb{D}(t_k, n_{i,j}), 1 \leq i \leq N, 1 \leq k \leq N_q \tag{15}$$

VQLS-B differs from VQLS-A only in the faster distance computation using $\mathbb{D}^*$ instead of $\mathbb{D}$; the distance $\hat{d}_{VQ}(\overrightarrow{x_i}, \overrightarrow{q}, k')$ is now computed using (16) instead of (13).

$$\hat{d}_{VQ}(\overrightarrow{x_i}, \overrightarrow{q}, k') = \sum_{k=1}^{k'} q_{t_k^*}.\mathbb{D}^*(i, t_k^*) \tag{16}$$

There is an increase in the storage required for lookup - $\mathbb{D}^*$ needs storage of $O(NU)$ but the time T_3 to compute all the distances in $\mathcal{A}$, without PDP, is now reduced to $O(MU + NN_q)$. Using PDP, T_3 reduces to $O(MU + NN_q')$,

TABLE III
THE RUNTIME NEEDED TO COMPUTE ALL THE MODEL-TO-QUERY DISTANCES (T_3) AND STORAGE (IN BITS) ARE COMPARED FOR VQLS-A AND VQLS-B.

	VQLS-A	VQLS-B
T_3	$O(MU + N'_q NF')$	$O(MU + NN'_q)$
storage	$U^2.b_{SQ,A}/2 + NF'b_{VQ} + 64.K\bar{F}p + 64.2^{b_{SQ,A}} + 64.18.2^{b_{VQ}}$	$NUb_{SQ,B}$+$64.K\bar{F}p$+$64.2^{b_{SQ,B}}$+$64.18.2^{b_{VQ}}$

as explained for VQLS-A in Sec. V-B2. Our experiments do confirm that this method has the lowest query time among all proposed methods (Table VIII), the only disadvantage being that the storage cost (linear in N) may become prohibitively high for very large datasets.

4) Storage Reduction for VQLS Methods: For a large codebook size U, the storage cost for the distance matrix $\mathbb{D}$ can be significantly high. The solution is to perform a non-uniform scalar quantization (SQ) on the elements in $\mathbb{D}$. Suppose, we have used a SQ of codebook size U_1. In that case, we just need to send the quantizer indices (each index needs $\lceil \log_2(U_1) \rceil$ bits) and maintain a table of the U_1 SQ centroids. Depending on the codebook size used, the memory savings can be substantial - without quantization, each element is a double needing 8 bytes = 64 bits. Our experiments have shown that we can do without very high resolution for the distance values and a 3-bit quantizer also works well in general. A low-bit scalar quantizer has also been used for the elements in $\mathbb{D}^*$, where the storage needed is $O(NU)$.

We present a quick comparison of the two VQ-based search methods, VQLS-A and VQLS-B, in Table III. The time complexity has already been explained while introducing the methods. Here, we elaborate on the storage complexity. For VQLS-A, the storage cost for $\mathbb{D}$ is $U^2.b_{SQ,A}/2$ where $2^{b_{SQ,A}}$ is the SQ codebook size used to encode the elements in $\mathbb{D}$. The SQ codebook is stored with a cost of $64.2^{b_{SQ,A}}$ bits. The storage cost for all the non-zero dimensions in the model video signatures is $NF'b_{VQ}$ where the CLD features are quantized using a VQ of size $2^{b_{VQ}}$. The storage size for the VQ that is used to encode the CLD features = $(64.2^{b_{VQ}}.18)$ bits = 9.43 MB (for $b_{VQ} = 13$). For VQLS-B, the storage cost for $\mathbb{D}^*$ is $NUb_{SQ,B}$ where $2^{b_{SQ,B}}$ is the scalar quantizer size used to encode the NU members in $\mathbb{D}^*$. The storage cost for the unquantized signatures of the top-K model videos returned by the first pass is $64.K\bar{F}p$, where the video signatures are assumed to have $\bar{F}$ feature vectors on an average.

C. Search Algorithms with Dataset Pruning

The VQLS schemes described above *consider all the N model videos* to return the top-K NN videos. Further speedup is obtained by reducing the number of model videos accessed during the search. We present two dataset pruning methods for VQ-based signatures. The first method (VQ-M1) guarantees that the same top-K NN videos are returned even after pruning, as using naive linear search. The second method (VQ-M2) is *an approximation*

of the first and achieves much higher pruning, though it is not guaranteed to return the correct top-K NN. The model-to-query distance (for the videos retained after pruning) can be computed using VQLS-A or VQLS-B (with PDP), for both VQ-M1 and VQ-M2.

1) Method VQ-M1: VQ-M1 uses a multi-pass approach for pruning. The logic is that *for a given query, the model videos which are nearest to it are likely to have some or all of the non-zero dimensions, as the query signature itself, as non-zero*.

The pre-computed information needed for VQ-M1 is listed below.

• We store a proximity matrix $\mathbb{P} \in \mathbb{R}^{U \times U}$ which stores the U nearest neighbors, in ascending order, for a certain VQ codevector, e.g., $\mathbb{P}(i,j)$ denotes the j^{th} NN for the i^{th} VQ codevector. For $U = 8192(2^{13})$, the storage cost of $\mathbb{P} = U^2.13$ bits (each of the U^2 terms represents an integer $\in [0, 2^{13}-1]$ and hence, is represented using 13 bits, giving a total storage cost of 109 MB).

• We also maintain a distance matrix $\mathbb{D}' \in \mathbb{R}^{U \times U}$ which stores the NN distances, in ascending order, for each VQ codevector. Here, $\mathbb{D}'(i,j)$ denotes the distance of the $\{\mathbb{P}(i,j)\}^{th}$ codevector from the i^{th} VQ codevector, i.e. $\mathbb{D}'(i,j) = \mathbb{D}(i, \mathbb{P}(i,j))$. We do not need to store $\mathbb{D}'$ explicitly as it can be computed using $\mathbb{D}$ and $\mathbb{P}$.

• We also store U clusters $\{\mathbb{C}(i)\}_{i=1}^{U}$, where $\mathbb{C}(i)$ denotes the cluster which contains those model video indices whose signatures have the i^{th} dimension as non-zero. The storage cost for 8192 clusters containing 38000 videos (the total model video dataset size for our experiments as mentioned in Sec. VI-A) is found to be equal to 6.3 MB.

$$\mathbb{C}(i) = \{j : x_{j,i} > 0,\ 1 \leq j \leq N\} \tag{17}$$

We now provide a list of symbols used in VQ-M1 (Algorithm 1) along with their definitions:

• $\mathbb{S}_j$: the set of distinct model videos considered in the j^{th} pass,

• G: the set of non-zero query dimensions, where $G = \{t_1, t_2, \cdots, t_{N_q}\}$,

• d_j^*: the minimum of the distances of all non-zero query dimensions to their j^{th} NN codevectors,

$$d_j^* = \min_{t_k \in G} \mathbb{D}'(t_k, j) \tag{18}$$

• A_j: the set of distinct VQ indices which are encountered on considering the first j NN for all the elements in G. Therefore, $(A_j \setminus A_{j-1})$ denotes the set of distinct (not seen in earlier passes) VQ indices encountered in the j^{th} pass, when we consider the j^{th} NN of the elements in G.

We maintain an ascending priority queue L of size K, for the K-NN videos, which is updated after every iteration. In the first iteration, we consider the union of the clusters which correspond to the non-zero query dimensions. We consider all the model videos from this union for distance computation. For the 1^{st} iteration, d_1^* equals 0 and the second iteration is almost always required. In the j^{th} iteration, we find the j-NN codevector of the non-zero

query dimensions and the new codevectors (not seen in the earlier iterations) are noted. We obtain the new model videos which have common non-zero dimensions with these newly encountered dimensions and consider them for distance computation. For the j^{th} iteration, we terminate the search for top-K NN if $d_j^* \geq L_{K,2}$ (or if all the N model videos have already been considered). For a formal proof that we are assured of finding the correct top-K NN if $d_j^* \geq L_{K,2}$, see [2]. If the terminating condition is satisfied at iteration $j=J$, the sequence of model videos considered is given by $\{\mathbb{S}_1, \mathbb{S}_2, \cdots, \mathbb{S}_{J-1}\}$.

Algorithm 1 Algorithm for VQ-M1 - here, unique(E) returns the unique (without repeats) elements in E

Input: N model video signatures, $\overrightarrow{x_i} \in \mathbb{R}^U,\ 1 \leq i \leq N$
Input: the query signature $\vec{q}$, and lookup matrices $\mathbb{P}$ and $\mathbb{D}'$ (along with the lookup tables needed by the distance computation method VQLS-A/B)
Output: Best sequence to search N videos for top-K NN and also top-K NN (model video indices)

Initialization: (1^{st} **pass**)
$G = \{t_1, t_2, \cdots, t_{N_q}\}$, the non-zero query dimensions
$A_1 = G$, set of 1-NN of elements in G is G itself
$\mathbb{S}_1 = \bigcup_{1 \leq i \leq N_q} \mathbb{C}(t_i)$, set of model videos having at least 1 non-zero dimension from G
$d_1^* = \min_{t_k \in G}\{\mathbb{D}'(t_k, 1)\} = 0$
We maintain an ascending priority queue L of length K, based on the elements in $\mathbb{S}_1$, where $d_{VQ}(\overrightarrow{x_i}, \overrightarrow{q})$ is found using (9) or (14), depending on whether VQLS-A/B is being used.
End of 1^{st} **pass**
for $j = 2$ to U **do**
  $d_j^* = \min_{t_k \in G}\{\mathbb{D}'(t_k, j)\}$, minimum distance between non-zero query dimensions to their j^{th} NN
  if $L_{K,2} \leq d_j^*$ **or** $\sum_{k=1}^{j} |\mathbb{S}_k| = N$ (all model videos have been considered) **then**
    break;
  end if
  $B_i = \mathbb{P}(t_i, j),\ 1 \leq i \leq N_q$, $B =$ set of VQ indices which are j^{th} NN of elements in G
  $E = B \setminus A_{j-1}$, $E =$ unique(E), set of VQ indices that are j^{th} NN of elements in G and were not seen in earlier iterations
  $\mathbb{S}_j = \bigcup_{1 \leq i \leq |E|} \mathbb{C}(E_i)$
  $\mathbb{S}_j = \mathbb{S}_j \setminus \bigcup_{1 \leq i < j} \mathbb{S}_i$, set of all model videos having at least one element in E as a non-zero dimension and these videos were not seen in earlier iterations
  $A_j = A_{j-1} \cup E$, set of all VQ indices which belong to one of the top j-NN for elements in G
  Update the priority queue L based on the elements in $\mathbb{S}_j$
end for
return the sequences observed so far $\{\mathbb{S}_1, \mathbb{S}_2, \cdots, \mathbb{S}_{J-1}\}$ (assuming that the search terminates at iteration $j = J$) and top-K NN from the priority queue L

We find that the maximum number of iterations (J) needed to obtain all the K-NN for a given query increases with both K and the fractional query length (ℓ), as shown in Table IV. For example, from Table IV, for K=10 and $\ell = 0.10$, the value of J is 500. Since we consider the j-NN for a VQ codevector at the j^{th} iteration, the number of NN that needs to be stored for each codevector equals the maximum number of iterations (J). Hence, the corresponding storage cost for $\mathbb{P}$ reduces to $(500/8192).109 = 6.65$ MB. We refer to this fraction (J/U) as $f(K, \ell)$ (a function of K and ℓ) when referring to the effective storage cost of $\mathbb{P}$, as used later in Table VIII.

We compare the dataset pruning obtained using DBH [3], trained using our distance function, with that of VQ-

TABLE IV
WE TABULATE THE AVERAGE J_{avg} (AVERAGING OVER ALL QUERIES) AND MAXIMUM NUMBER OF ITERATIONS J FOR VARYING FRACTIONAL QUERY LENGTHS (ℓ) AND K, FOR U = 8192. BOTH J_{avg} AND J INCREASE WITH K AND ℓ.

K	$J_{avg}(\ell=0.05)$	$J(\ell=0.05)$	$J_{avg}(\ell=0.10)$	$J(\ell=0.10)$	$J_{avg}(\ell=0.50)$	$J(\ell=0.50)$
10	2	450	3	500	31	1300
50	4	750	9	850	62	1600
100	8	950	17	1000	82	1750

TABLE V
WE COMPARE THE **PERCENTAGE OF MODEL VIDEOS RETAINED AFTER DATASET PRUNING** FOR VQ-M1 WITH THAT OBTAINED USING DBH, FOR DIFFERENT FRACTIONAL QUERY LENGTHS (ℓ) AND K. FOR DBH, $p_{error} = 0.05$ IS USED.

ℓ	VQ-M1($K=10$)	VQ-M1($K=50$)	VQ-M1($K=100$)	DBH($K=10$)	DBH($K=50$)	DBH($K=100$)
0.05	15.03	20.52	23.90	71.85	78.37	81.97
0.10	21.22	27.23	30.83	73.64	80.93	82.13
0.50	42.04	48.21	51.51	78.16	81.40	83.24

M1 (Table V) [2]. It is observed that the pruning obtained using VQ-M1 is significantly higher than that obtained using DBH. For DBH, the pruning obtained depends on the allowed error probability (p_{error}) - we report results for p_{error} of 0.05. As mentioned earlier, we are guaranteed ($p_{error}=0$) to return the top-K NN using VQ-M1.

2) Method VQ-M2: Based on empirical observations, we assume that *the signature of the duplicate video, created from a subset of frames in the original video with noise attacks on the frames, will have common non-zero dimensions with the original model video signature*. Hence, the list of model videos considered for K-NN candidates corresponds to $\mathbb{S}_1$, the sequence of videos returned by the first iteration of VQ-M1. Thus, VQ-M2 is a single iteration process.

This method introduces errors only if there is no overlap between the non-zero dimensions of the query and the original model video, i.e. if the best matched video index $i^* \notin \mathbb{S}_1$. When the noise attacks introduce enough distortion in the feature vector space (*so that non-zero query dimensions may not overlap with the non-zero dimensions of the original model signature*), a simple extension is to consider P NN ($P>1$) across each non-zero query dimension. Thus, P should increase with the amount of distortion expected, and pruning gains decrease with increasing P. The number of videos in $\mathbb{S}_1$ and the storage cost for the proximity matrix $\mathbb{P}$ (defined for VQ-M1) depend on P. For $P > 1$, the storage cost for $\mathbb{P}$ is $O(UP)$.

The sequence $\mathbb{S}_1$, for $P \geq 1$, is obtained as follows (for VQ-M1, $\mathbb{S}_1$ corresponds to $P = 1$):

$$\mathbb{S}_1 = \bigcup_{1 \leq i \leq N_q} \mathbb{C}(t_i) \text{ using (17), for } P = 1$$

$$\mathbb{S}_1 = \bigcup_{j \in \mathcal{B}} \mathbb{C}(j) \text{ using (17), where } \mathcal{B} = \bigcup_{1 \leq i \leq N_q, 1 \leq k \leq P} \mathbb{P}(t_i, k), \text{ for } P \geq 1$$

In Fig. 4, we compare the dataset pruning obtained for different choices of P, fractional query lengths, and using different number of keyframes for creating the query signatures. Using a higher fraction of query keyframes

[2]The DBH implementation is courtesy Michalis Potamias, a co-author in [3].

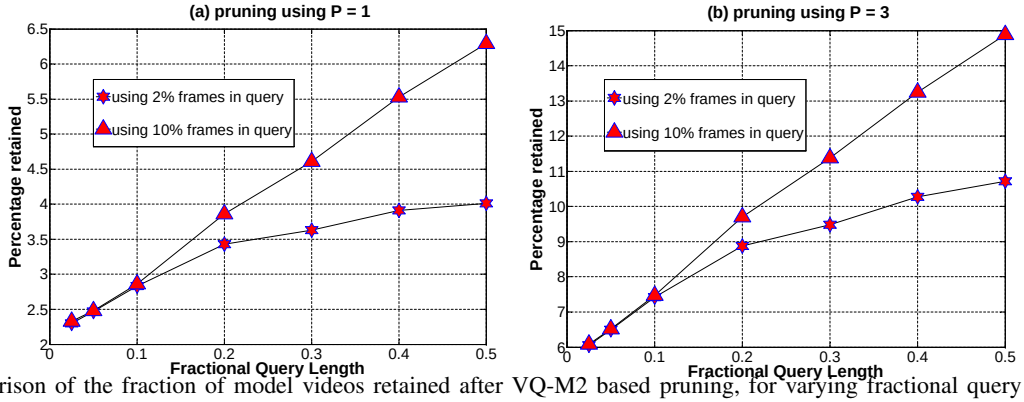


Fig. 4. Comparison of the fraction of model videos retained after VQ-M2 based pruning, for varying fractional query lengths, and using different sized query signatures. The number of cluster centers for the query is fixed at 2% and 10% of the number of query frames, after temporal sub-sampling, i.e. $M/T_Q = 0.02$ and 0.10 (notations as in Fig. 1) for the 2 cases.

TABLE VI

THE TIME NEEDED (IN TERMS OF T_{pr}, T_3, AND T_4) AND STORAGE (IN BITS) ARE COMPARED FOR VQ-M1 AND VQ-M2.

	T_{pr}	T_3	T_4	storage
VQLS-A	0	$O(MU + N'_q N F')$	$O(N \log K)$	$U^2.b_{SQ,A}/2 + NF'b_{VQ} + 64.K\bar{F}p + 64.2^{b_{SQ,A}} + 64.18.2^{b_{VQ}}$
VQ-M1(A)	$T_{pr,1}$	$O(MU + N'_q N_{pr,1} F')$	$O(N_{pr,1} \log K)$	$U^2.13.f(K,\ell)$ + 6.3 MB (for $b_{VQ} = 13$) + storage(VQLS-A)
VQ-M2(A)	$T_{pr,2}$	$O(MU + N'_q N_{pr,2} F')$	$O(N_{pr,2} \log K)$	6.3 MB (for $b_{VQ} = 13$) + storage(VQLS-A)

(M/T_Q), the pruning benefits are reduced, as more model videos are now considered due to the higher number of non-zero query dimensions. The percentage of videos retained after VQ-M2 based pruning is 3% and 7.5%, for 10% length queries, for P = 1 and P = 3, respectively. From Table V, the corresponding pruning obtained using VQ-M1 varies from 21%-31% as K is varied from 10-100.

For dataset pruning, we have presented two methods: VQ-M1 and VQ-M2. We present a quick overview of these methods through Table VI, where we compare their runtime and storage requirements. $T_{pr,1}$ and $T_{pr,2}$ refer to the time needed for dataset pruning for VQ-M1 and VQ-M2, respectively.

For VQ-M1(A), the additional costs, over that of VQLS-A, needed for pruning are $U^2.13.f(K,\ell)$ (the cost for maintaining the proximity matrix $\mathbb{P}$) and 6.3 MB (the cost for maintaining the 8192 clusters). For VQ-M2(B), the proximity matrix $\mathbb{P}$ is not needed. The number of model videos retained after pruning are denoted by $N_{pr,1}$ and $N_{pr,2}$ for VQ-M1(A) and VQ-M2(A), respectively. This helps to reduce T_3 and T_4, which are defined in Table II. The pruning obtained by VQ-M1 and VQ-M2 are determined at runtime depending on the query and we have numerically compared the pruning achieved using Fig. 4 and Table V. To reiterate, VQ-M1 is an iterative process (e.g., we need $J \geq 1$ iterations) while VQ-M2 is a one-pass process. Thus, in general, $T_{pr,1} > T_{pr,2}$ and $N_{pr,1} > N_{pr,2}$.

VI. Experimental Setup and Results

Sec. VI-A explains the dataset creation for duplicate detection. We have performed a variety of noise attacks and we empirically compare the duplicate detection accuracy over these attacks. Sec. VI-B presents the comparison of the different speedup techniques proposed for improving the coarse search. Sec. VI-C shows how our distance measure outperforms other histogram-based distances for VQ-based signatures.

A. Dataset Generation and Evaluation of Duplication Attacks

Two online video repositories *www.metacafe.com* and *www.youtube.com* are crawled to obtain a database of 38000 model videos, worth about 1600 hours of video content. A randomly chosen subset of 1200 videos ($\approx$ 50 hours of content), is used to generate the query videos. We perform various modifications on the decoded query frames for each of these 1200 videos to generate 18 duplicates per video. We empirically observe that the CLD feature is robust to the discussed modifications. The number of duplicates for each noise class is shown in parentheses.

- Gaussian blurring using a 3×3 and 5×5 window, (2)
- resizing the image along each dimension by a factor of 75% and 50%, respectively, (2)
- gamma correction by -20% and 20%, (2)
- addition of AWGN (additive white Gaussian noise) using SNR of -20, 0, 10, 20, 30 and 40 dB, (6)
- JPEG compression at quality factors of 10, 30, 50, 70 and 90, (5)
- cropping the frames to 90% of their size (1).

The frame drops that are considered can be random or bursty. We simulate the frame drops by creating a query video as a fraction (2.5%-50%) of the model video frames. The duplicate detection accuracy after the individual noise attacks is shown in Table VII.

Re-encoding Attacks: The downloaded videos are originally in Flash Video Player (FLV) format and they are converted to MPEG-1 format to generate the query video. We have also re-encoded the video using MPEG-2, MPEG-4, and Windows Media Video (WMV) formats. The CLD feature is robust against global attacks induced by strong AWGN and JPEG compression attacks and hence, robustness is expected against video re-encoding attacks - this is also experimentally verified. For MPEG-4 compressed videos, we experiment with varying frame rates (5, 10, 20, 30, 40 and 80 frames/sec) and the average detection accuracy is 99.25% - the results remain almost constant for different frame rates.

Color to Gray-scale Conversion: We have also converted the model video frames from color to gray-scale to create the query - here, the Y component is slightly modified. For gray-scale videos, we consider the first 6 dimensions of the CLD feature, which correspond to the DCT terms for the Y channel, as the effective signature. *The decision to use 6 or 18 dimensions is made based on whether dimensions 8-12 and 14-18 (AC DCT coefficients*

TABLE VII
THE DETECTION ERROR OBTAINED USING CLD FEATURES, FOR INDIVIDUAL NOISE ATTACKS, AVERAGED OVER FRACTIONAL QUERY LENGTHS FROM 2.5%-50%, AND OVER VARYING PARAMETERS FOR A GIVEN ATTACK, ARE SHOWN.

Attack	Error	Attack	Error	Attack	Error	Attack	Error
blur	0.0114	resize	0.0111	gamma	0.0221	AWGN	0.0113
JPEG	0.0125	crop	0.0145	(blur + crop)	0.0154	(resize + crop)	0.0148
(blur+resize)	0.0119	(AWGN+crop)	0.0156	(gamma + crop)	0.0243	(AWGN + resize)	0.0128
MPEG-2	0.0098	MPEG-4	0.0088	WMV	0.0076	gray-scale	0.0388
logo (5%)	0.0140	logo (10%)	0.1780	logo (15%)	0.0198	logo (20%)	0.0228
caption (30%)	0.0155	caption (50%)	0.0190	caption (70%)	0.02301	caption (90%)	0.0288

for Cb and Cr channels) are all zero, i.e. it is a gray-scale frame. If frames of a different video are added to the query video, then as the percentage of inserted frames (from other videos) increases, the detection accuracy decreases significantly as shown in Fig. 5(a).

Logo and Caption Insertions: We have also experimented with logo and caption insertions. The initial logo considered is a 60×90 binary patch with 700 pixels (they constitute the logo pattern) being set to 1. We then resize the logo to 5%, 10%, 15% and 20% of the image size. We superimpose the logo pattern on the bottom leftmost part of the image and the image pixels, whose positions coincide with the 1's in the logo, are set to zero (black logo). For the caption insertion, the original block of text can be captured in a 50×850 binary patch where 2050 pixels (constituting the caption) are set to 1. We then resize the caption such that it can span a different number of columns (30%, 50%, 70% and 90% of the image size). The same principle is used to modify the image as in the logo insertion example. The coarseness of the CLD feature explains its relative robustness against logo and caption insertions. The averaging of the entire image to an 8×8 representation dilutes the effect of local changes.

B. Empirical Evaluation of Various Proposed Algorithms

We analyze the performance of the proposed algorithms for duplicate detection. The final detection accuracy, for a certain query length, is obtained by averaging over all the (1200×18) noisy queries, where the 18 duplication schemes were introduced in Sec. VI-A.

• Firstly, we show the speedup obtained using PDP, by comparing PLS (NLS + PDP) with NLS, and comparing VQLS-A and VQLS-B schemes, with and without PDP (Fig. 5(b)). It is also seen that the VQ-based schemes significantly outperform NLS and PLS, that use un-quantized features.

• Secondly, we show the performance improvements obtained using VQ-M1(A) and VQ-M2(A), in place of VQLS-A, and using VQ-M2(B) in place of VQLS-B - these methods achieve additional speedup through dataset pruning (Fig. 6(a) and 6(b)).

Speedup Obtained Using PDP: We show the runtime needed ($T_3 + T_4$ from Table II), with and without PDP for NLS, VQLS-A and VQLS-B schemes, in Fig. 5(b-1), (b-2) and (b-3), respectively, to return the top-K model

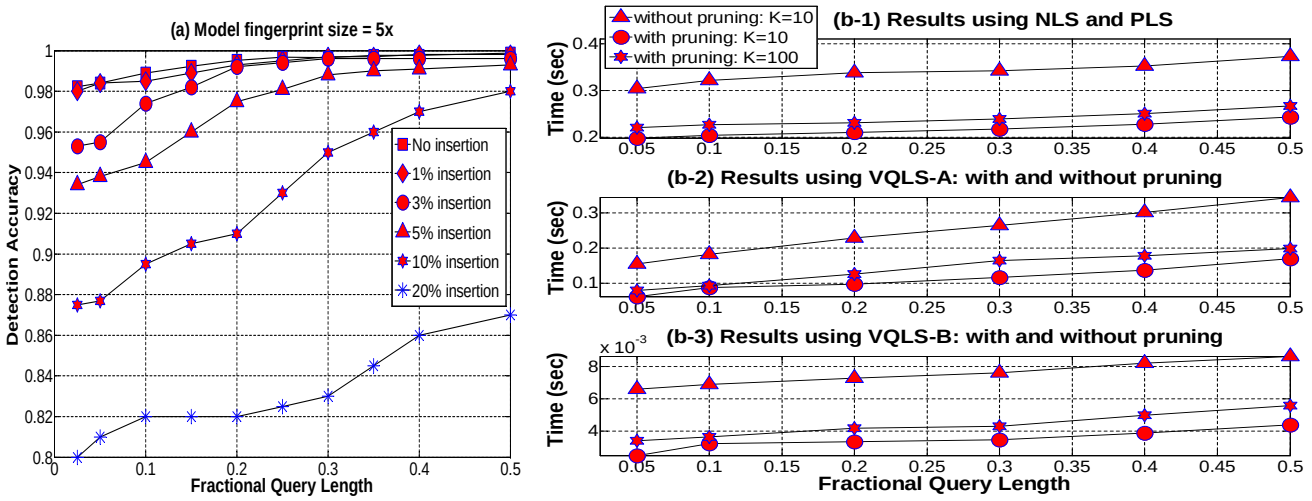


Fig. 5. (a) Variation of the detection accuracy with varying levels of video clip (from a different video) insertion - a fractional query length of 0.1 means that the query consists of 10% frames present in the (original query + inserted video clip). (b) Runtime improvements due to PDP are shown for the PLS and VQ-based linear search schemes. "Pruning/ no pruning" indicates whether or not PDP has been used. Here, runtime = $(T_3 + T_4)$ is the time needed to return the top-K model videos after the first pass.

videos. T_3 is reduced by using PDP. $T_4 = O(N \log K)$ increases with K and thus, the effective runtime saving decreases as K increases. PDP provides significant runtime saving so that "with pruning: $K = 100$" takes lesser time than "without pruning: $K = 10$". Also, comparing (b-2) and (b-3) with (b-1) in Fig. 5, we observe that the runtime needed by VQLS-A and VQLS-B (with PDP) is much lower than that for PLS and NLS.

Speedup Obtained through Dataset Pruning: We observe the runtime saving obtained through dataset pruning (using VQ-M1 and VQ-M2) using VQLS-A and VQLS-B for the model-to-query distance computation, in Fig. 6(a) and 6(b), respectively. PDP is employed for all the methods and "prune/no prune" denotes whether or not we employ dataset pruning methods (VQ-M1 or VQ-M2).

• For VQLS-A, the runtime comparison for the different methods is: VQLS-A > VQ-M1(A) > VQ-M2(A). Hence, using dataset pruning results in significant speedup (Fig. 6(a)).

• For VQLS-B, the use of the lookup table D^* reduces runtime significantly, so that the time required for the iterative pruning technique (VQ-M1) is higher than the runtime without pruning, especially for higher values of K and longer queries. Hence, for VQLS-B, for fractional query lengths exceeding 0.10, the runtime comparison for the various methods is: VQ-M1(B) > VQLS-B > VQ-M2(B) (Fig. 6(b)).

Storage and Time Comparison: We present the variation of the detection accuracy with query time, along with the associated storage costs, for the various methods in Table VIII. The query lengths considered were 10% and 50% of the actual model video lengths. *It is seen that among methods with higher storage costs (using $\mathbb{D}^*$, where storage $\propto N$), VQ-M2(B) has the minimum query time while for methods with lower storage costs (using $\mathbb{D}$, where storage $\propto U^2$), VQ-M2(A) has the minimum query time*. The various values used in Table VIII are $\bar{F} = 25$, $F' = 18$, $b_{VQ} = 13$, $b_{SQ,A} = 3$, $b_{SQ,B} = 3$, $N = 38000$ (dataset size) and $U = 8192$ (VQ size).

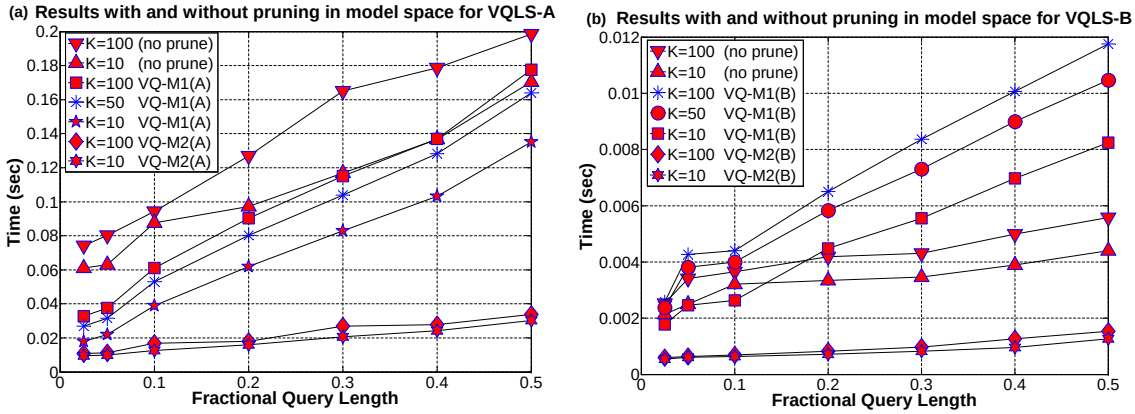


Fig. 6. Runtime improvements due to pruning in the model video space, for VQLS-A and VQLS-B, are shown. By "*no prune*", we mean that *pruning in model video space (VQ-M1 or VQ-M2) is absent, while PDP is used for all the methods*. Significant runtime savings are obtained for VQ-M1(A) and VQ-M2(A) over VQLS-A (Fig. a) and for VQ-M2(B) over VQLS-B (Fig. b).

TABLE VIII

WE COMPARE ALL THE 3 PARAMETERS - **DETECTION ACCURACY, QUERY TIME (EXPRESSED IN SECONDS) AND STORAGE** FOR THE DIFFERENT METHODS, AT VARYING K, AND OBSERVE THE ASSOCIATED TRADE-OFFS. THE QUERY TIME EQUALS $(T_3 + T_4 + T_5)$ (ALONG WITH THE TIME FOR KMEANS-CLUSTERING TO OBTAIN Q FROM Q_{orig} AND THE TIME FOR SORTING THE QUERY DIMENSIONS). UNLESS OTHERWISE MENTIONED, THE ELEMENTS ARE STORED IN "DOUBLE" FORMAT (= 64 BITS). THE STORAGE COST OF VQ-M1(A) DEPENDS ON THE FRACTIONAL QUERY LENGTH (ℓ): THUS, FOR $K = 10$, THE STORAGE COST EQUALS 35.86 AND 46.51 MB FOR ℓ = 0.10 AND 0.50, RESPECTIVELY.

Index	Method	K	Storage (bits)	Storage (MB)	ℓ = 0.10		ℓ = 0.50	
					query time	Accuracy	query time	Accuracy
1	NLS	10	$64.N\bar{F}p$	133.47	0.42	0.989	0.48	0.998
	NLS	50			0.43	0.994	0.49	0.999
2	PLS	10	$64.N\bar{F}p$	133.47	0.27	0.989	0.32	0.998
	PLS	50			0.28	0.994	0.33	0.999
3	VQLS-A	10	$64.18.2^{b_{VQ}}$ (9.43 MB) +	22.91	0.096	0.883	0.201	0.975
	VQLS-A	50	$U^2.b_{SQ,A}/2 + NF'b_{VQ}$	23.06	0.102	0.958	0.227	0.994
	VQLS-A	**100**	$+64.K\bar{F}p + 64.2^{b_{SQ,A}}$	**23.24**	**0.109**	**0.969**	**0.257**	**0.996**
4	VQ-M1(A)	10	$U^2.13.f(K,\ell)$ + 6.3 MB +	35.86, 46.51	0.048	0.883	0.165	0.975
	VQ-M1(A)	50	$U^2.b_{SQ,A}/2 + NF'b_{VQ}+$	40.67, 50.65	0.065	0.958	0.206	0.994
	VQ-M1(A)	**100**	$64.K\bar{F}p + 64.2^{b_{SQ,A}}$ + 9.43 MB	**42.85, 52.83**	**0.076**	**0.969**	**0.237**	**0.996**
5	VQ-M2(A)	10	9.43 MB + cluster cost (6.3 MB)	29.21	0.014	0.883	0.047	0.975
	VQ-M2(A)	50	$+U^2.b_{SQ,A}/2 + NF'b_{VQ}$	29.36	0.020	0.958	0.062	0.994
	VQ-M2(A)	**100**	$+64.K\bar{F}p + 64.2^{b_{SQ,A}}$	**29.54**	**0.024**	**0.969**	**0.080**	**0.996**
6	VQLS-B	10	9.43 MB +	123.36	0.012	0.883	0.035	0.975
	VQLS-B	50	$NUb_{SQ,B} + 64.K\bar{F}p$	123.51	0.015	0.958	0.047	0.994
	VQLS-B	**100**	$+64.2^{b_{SQ,B}}$	**123.69**	**0.019**	**0.969**	**0.065**	**0.996**
7	VQ-M2(B)	10	9.43 MB + cluster cost (6.3 MB)	129.66	0.010	0.883	0.032	0.975
	VQ-M2(B)	50	$+NUb_{SQ,B} + 64.K\bar{F}p$	130.26	0.013	0.958	0.043	0.994
	VQ-M2(B)	**100**	$+64.2^{b_{SQ,B}}$	**130.99**	**0.016**	**0.969**	**0.061**	**0.996**

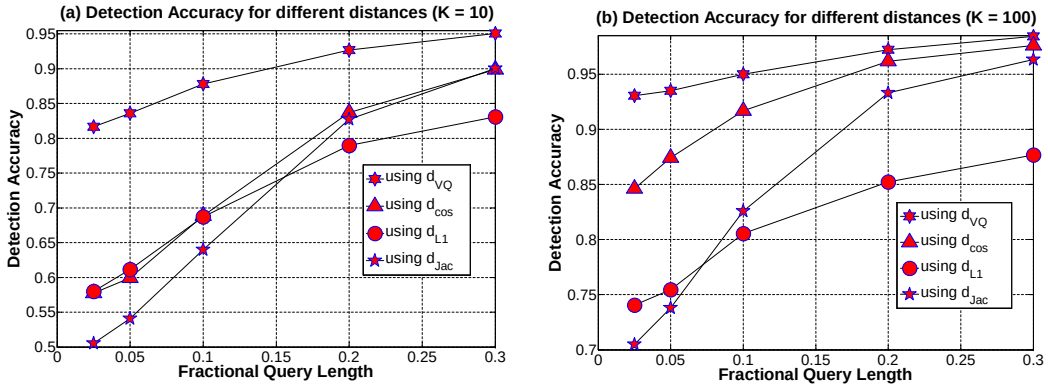


Fig. 7. Comparison of the detection accuracy obtained using the different VQ based distances, for $K = 10$ and $K = 100$, is shown. Results using d_{int} and d_{L1} are near-identical and so, only d_{L1} based results are shown. Results using d_{VQ} are significantly better than that using d_{cos} (which in turn performs better than d_{L1} and d_{Jac}) at smaller query lengths.

C. Comparison of Other Histogram based Distances for VQ-based Signatures

We compare our distance measure between the VQ-based signatures with the L_1 distance (d_{L1}), an intersection based distance (d_{int}), the cosine distance (d_{cos}) and the Jaccard coefficient based distance (d_{Jac}), which was used for copy detection in [9]. The different distance measures are defined here:

$$d_{int}(\overrightarrow{x_i}, \overrightarrow{q}) = 1 - \sum_{k=1}^{U} \min(x_{i,k}, q_k), \quad d_{cos}(\overrightarrow{x_i}, \overrightarrow{q}) = (\sum_{j=1}^{U} x_{i,j} q_j)/(||\overrightarrow{x_i}||_2 . ||\overrightarrow{q}||_2)$$

$$\text{Jaccard coefficient } J_{coeff} = \sum_{k=1}^{U} \frac{\min(x_{i,k}, q_k)}{\max(x_{i,k}, q_k)}, \text{ and } d_{Jac} = -J_{coeff}$$

The performance comparison of the different distance measures (Fig. 7) shows that the detection accuracy using d_{VQ} is significantly higher than the other distances, especially for small query lengths. *For our proposed measure, the effective distance is the sum of distances between "query vector to best matching vector in model signature".* For traditional histogram-based distances, the effective distance is computed between corresponding bins in the model and query signatures - this distance is small only when the query signature is similar to the entire model signature, which is true mainly for longer queries. Hence, *the advantage of using our asymmetric distance is more obvious for shorter query lengths.*

VII. Duplicate Confirmation

After finding the best matched video V_{i^*}, we discuss a distance threshold based (Sec. VII-A) and a registration-based (Sec. VII-B) approach to confirm whether the query is a duplicate derived from V_{i^*}.

A. Distance Threshold based Approach

The training phase to obtain the distance threshold involves finding the 1-NN and 2-NN distances for 1200 query videos, over various noise conditions and query lengths. The distance between X^{i^*}, the fingerprint of the

1-NN video V_{i^*}, and the larger query signature Q_{orig}, is computed using (1) and is normalized by the query length T_Q, so as to make the threshold independent of the query length. Thus, the effective 1-NN distance equals $\{d(X^{i^*}, Q_{orig})/T_Q\}$. Since the same 1200 videos were considered as the model videos, the 1-NN always refers to a duplicate video and the 2-NN to a non-duplicate one. Ideally, the threshold δ_s should be such that all the 1-NN (or 2-NN) distances are less (or greater) than it. By equally weighing the probability of false alarm P_{FA} (wrongly classifying the 2-NN retrieval as a duplicate) and missed detection P_{MD} (failing to classify the 1-NN retrieval as a duplicate), the threshold δ_s is empirically set at 230 - distribution of 1-NN and 2-NN distances and illustrative explanation of threshold selection are shown in [2]. The corresponding P_{FA} and P_{MD} values equal 0.07. Depending on whether the emphasis is on minimizing P_{FA} or P_{MD}, δ_s can be decreased or increased, accordingly.

For verifying the effectiveness of the distance threshold, we repeat the duplicate detection experiments *on an unseen dataset of 1700 videos ($\approx$ 75 hours of video), all of which are different from the model videos*. For each video, 18 duplicates are created as in Sec. VI-A. Using a threshold δ_s of 230, 3% of the videos were classified as "duplicates" - for them, the 1-NN distance is less than δ_s.

For those cases where the query-to-model distance is very close to the threshold δ_s, we use a registration-based approach (Sec. VII-B). The registration method is computationally intensive but is more accurate in determining if the query is indeed a duplicate of the retrieved candidate.

B. Registration based Approach

In this approach, we need to know which model keyframe should be considered for registration for a given query keyframe. While computing the distance $d(X^{i^*}, Q_{orig})$ in the second pass of the search process, we have already obtained the best matching vector in the model signature ($X^{i^*} \in \mathbb{R}^{F_{i^*} \times p}$) for every query vector in Q_{orig}. What we now need is a way to map every model (query) vector to its corresponding keyframe. This is done as follows. Considering the cluster centers (X^{i^*}) obtained after k-means clustering on the feature matrix (Z^{i^*}), we can find which vector in Z^{i^*} best matches to a certain vector in X^{i^*} - the frames corresponding to the selected vectors in Z^{i^*} constitute the model keyframes.

Registration method: First, a set of salient points is detected in the respective frames. Then the SIFT feature descriptor is computed locally around those points followed by establishing correspondences between them by computing the distance in the SIFT feature space. As this usually yields a lot of false matches (more than 50% in some scenarios), RANSAC [14] is included in this framework to filter out the bad point correspondences and to get a robust estimate of homography parameters. Finally, we conclude that the query video is a duplicate of V_{i^*} if majority of the query frames (approximately 70% in our case) can indeed be registered with the best matching keyframes in V_{i^*}. This fraction (70%) can be increased or decreased depending on whether the emphasis is on minimizing P_{FA} or P_{MD}.

VIII. Discussion

Here we have addressed the duplicate video detection problem. We empirically selected CLD for fingerprinting as it was robust to the duplication attacks. However, if there is extensive cropping, padding, or rotation/shear, salient point-based descriptors can be more effective. We developed a new non-metric distance measure which is very effective for short queries. This distance measure has high computational complexity as it computes the distances between all model-to-query keyframe pairs. We reduce the computational cost using pre-computed information, partial distance based pruning and dataset pruning. This distance measure can be explored in other domains which require subset matching. The proposed dataset pruning method has been effective for our distance function and VQ histogram based signatures. It would be interesting to study how well the pruning method generalizes for histogram-based distances.

IX. Conclusion

The problem of *fast and real-time duplicate detection* in a large video database is investigated through a suite of efficient algorithms. We retrieve the duplicate video for about a minute long query in 0.03 sec with an average detection accuracy of over 97%. Our proposed distance measure is shown to perform very well when the query is a noisy subset of a model video and keyframe-based signatures are used. In the future, we will explore how the duplicate detection system scales to larger sized datasets.

In our problem, we have assumed that the query is entirely constituted from a model video. If, however, a query contains portions of multiple videos, the same asymmetric distance will not be effective. In that scenario, one can consider disjoint windows (of suitable length) of the query video and issue multiple queries. The aim is to identify the model to which a certain query window can be associated. This topic will be explored in future.

Acknowledgment

We would like to thank all the anonymous reviewers for their valuable comments. Anindya Sarkar was supported by ONR grants #N00014-05-1-0816 and #N00014-10-1-0141. Vishwakarma Singh and Pratim Ghosh were supported by NSF grants, NSF ITR #0331697 and NSF III #0808772.

References

[1] http://people.csail.mit.edu/fergus/iccv2005/bagwords.html.

[2] http://vision.ece.ucsb.edu/publications/Sarkar_tech_report_DTW_09.pdf.

[3] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. *Proc. of ICDE*, pages 327–336, April 2008.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[5] M. Bertini, A. D. Bimbo, and W. Nunziati. Video clip matching using MPEG-7 descriptors and edit distance. In *Proc. of CIVR*, pages 133–142, 2006.

[6] D. N. Bhat and S. K. Nayar. Ordinal measures for image correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 415–423, 1998.

[7] S. Cheung and A. Zakhor. Estimation of web video multiplicity. In *Proc. SPIE–Internet Imaging*, volume 3964, pages 34–36, 1999.

[8] C. Chiu, C. C. Yang, and C. S. Chen. Efficient and effective video copy detection based on spatiotemporal analysis. In *Ninth IEEE International Symposium on Multimedia*, pages 202–209, Dec 2007.

[9] C. Y. Chiu, J. H. Wang, and H. C. Chang. Efficient histogram-based indexing for video copy detection. *Multimedia Workshops, 2007. ISMW '07. Ninth IEEE International Symposium on*, pages 265–270, Dec. 2007.

[10] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[11] K. Dadason, H. Lejsek, F. Asmundsson, B. Jonsson, and L. Amsaleg. Videntifier: identifying pirated videos in real-time. In *Proc. of the 15th International Conference on Multimedia*, pages 471–472. ACM, 2007.

[12] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of the 20th Annual Symposium on Computational Geometry*, pages 253–262, 2004.

[13] S. Derrode and F. Ghorbel. Robust and efficient Fourier-Mellin transform approximations for gray-level image reconstruction and complete invariant description. *Computer Vision and Image Understanding*, 83(1):57–78, 2001.

[14] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[15] P. Ghosh, E. D. Gelasca, K. Ramakrishnan, and B. Manjunath. *Duplicate Image Detection in Large Scale Databases*. Book Chapter in Platinum Jubilee Volume, Oct 2007.

[16] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[17] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 47–57. ACM, 1984.

[18] A. Hampapur and R. M. Bolle. Comparison of distance measures for video copy detection. In *Proc. of ICME*, pages 737 – 740, Aug 2001.

[19] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sept 1993.

[20] A. Joly, O. Buisson, and C. Frelicot. Statistical similarity search applied to content-based video copy detection. *Int. Conf. on Data Engineering Workshops*, page 1285, 2005.

[21] A. Joly, O. Buisson, and C. Frelicot. Content-based copy retrieval using distortion-based probabilistic similarity search. *Multimedia, IEEE Transactions on*, 9(2):293–306, Feb. 2007.

[22] A. Joly, C. Frelicot, and O. Buisson. Robust content-based video copy identification in a large reference database. In *Int. Conf. on Image and Video Retrieval*, pages 414–424, 2003.

[23] A. Joly, C. Frelicot, and O. Buisson. Feature statistical retrieval applied to content based copy identification. *Image Processing, 2004. ICIP '04. 2004 International Conference on*, 1:681–684 Vol. 1, Oct. 2004.

[24] E. Kasutani and A. Yamada. The MPEG-7 color layout descriptor: a compact image feature description for high-speed image/video segment retrieval. In *Proc. of ICIP*, volume 1, pages 674–677, 2001.

[25] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. of CVPR*, pages 506–513, 2004.

[26] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. In *Proc. of the 14th annual ACM International Conference on Multimedia*, pages 835–844. ACM, 2006.

[27] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford. Video copy detection: a comparative study. In *Proc. of CIVR*, pages 371–378. ACM, 2007.

[28] H. Lejsek, F. H. Asmundsson, B. Jonsson, and L. Amsaleg. NV-tree: An efficient disk-based index for approximate search in very large high-dimensional collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 2008.

[29] H. Lejsek, F. H. Asmundsson, B. T. Jonsson, and L. Amsaleg. Scalability of local image descriptors: a comparative study. In *ACM MULTIMEDIA '06*, pages 589–598, New York, NY, USA, 2006. ACM.

[30] Y. Li, J. S. Jin, and X. Zhou. Matching commercial clips from TV streams using a unique, robust and compact signature. In *Digital Image Computing: Techniques and Applications, 2005. DICTA'05. Proceedings 2005*, pages 39–39, 2005.

[31] Y. Linde, A. Buzo, R. Gray, et al. An algorithm for vector quantizer design. *IEEE Transactions on communications*, 28(1):84–95, 1980.

[32] L. Liu, W. Lai, X. Hua, and S. Yang. Video Histogram: A Novel Video Signature for Efficient Web Video Duplicate Detection. *Lecture Notes in Computer Science*, 4352:94–103, 2007.

[33] D. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, volume 20, pages 91–110, 2003.

[34] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proc. European Conf. Computer Vision*, pages 128–142. Springer Verlag, 2002.

[35] R. Mohan. Video sequence matching. In *Proc. of ICASSP*, pages 3697–3700, 1998.

[36] P. Over, A. F. Smeaton, and P. Kelly. The TRECVID 2007 BBC rushes summarization evaluation pilot. In *TVS '07: Proc. of the International Workshop on TRECVID Video Summarization*, pages 1–15. ACM, 2007.

[37] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, NJ, 1993.

[38] A. Sarkar, P. Ghosh, E. Moxley, and B. S. Manjunath. Video fingerprinting: Features for duplicate and similar video detection and query-based video retrieval. In *Proc. of SPIE, Multimedia Content Access: Algorithms and Systems II*, volume 6820, pages 68200E–68200E–12, 2008.

[39] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou. UQLIPS: a real-time near-duplicate video clip detection system. In *VLDB*, pages 1374–1377. VLDB Endowment, 2007.

[40] H. Tan, X. Wu, C. Ngo, and W. Zhao. Accelerating near-duplicate video matching by combining visual similarity and alignment distortion. In *ACM MULTIMEDIA '08*, pages 861–864. ACM, 2008.

[41] C. Won, D. Park, and S. Park. Efficient use of MPEG-7 edge histogram descriptor. *Etri Journal*, 24(1):23–30, 2002.

[42] X. Wu, A. G. Hauptmann, and C. Ngo. Practical elimination of near-duplicates from web video search. In *Proceedings of the 15th International Conference on Multimedia*, pages 218–227. ACM, 2007.

[43] J. Yuan, L. Y. Duan, Q. Tian, and C. Xu. Fast and robust short video clip search using an index structure. In *Proceedings of the 6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, pages 61–68, 2004.

[44] W. Zhao and C. Ngo. Scale-Rotation Invariant Pattern Entropy for Keypoint-Based Near-Duplicate Detection. *IEEE Transactions on Image Processing*, 18(2):412–423, 2009.

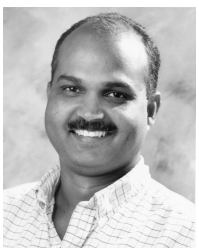
[45] W. L. Zhao, C. W. Ngo, H. K. Tan, and X. Wu. Near-duplicate keyframe identification with interest point matching and pattern learning. *IEEE Transactions on Multimedia*, 9:1037–1048, 2007.

Anindya Sarkar received his B.E. from Jadavpur University, Kolkata, India, in 2003 in the Electronics and Telecommunication Engineering Department. He finished his M.E. (with distinction) in Signal Processing from the Indian Institute of Science, Bangalore, in 2005 where his research was on speech segmentation. He received the Alfred Hay Gold Medal for the best M.E. thesis in the Electrical Engineering Department. Since Fall 2005, he has been a PhD student in the ECE Department at UCSB working with Prof. Manjunath. In the summer of 2008, he interned at Google Research in Mountain View, CA. His research interests include data hiding, steganography, image forensics, video fingerprinting, and efficient video search and retrieval.

Vishwakarma Singh received his B.Tech. in 2003 from the Department of Computer Scince and Engineering, Institute of Technology, Benaras Hindu University, India. He was employed with Oracle (2003-2005) and Qwest (2005-2006). Since Fall 2006, he is a PhD student in the Department of Computer Science at UCSB working with Prof. Ambuj Singh. He interned at Nokia Research and Dolby Laboratories Research in summer of 2007 and 2009 respectively. His research emphasis is high dimensional data modeling, retrieval, and mining in large scale datasets.

Pratim Ghosh received his B.E. degree in Electrical Engineering from Jadavpur University, Kolkata in 2004 and M.E. degree in System Science and Automation (with distinction) from Indian Institute of Science, Bangalore in 2006. He has interned at Janelia Farm, Howard Hughes Medical Institute in the summer of 2009. He is currently a Ph. D. student in the Department of Electrical and Computer Engineering at the University of California, Santa Barbara. His research interests include computer vision and pattern recognition with emphasis on variational methods for image segmentation.

B. S. Manjunath received the B. E. in Electronics from the Bangalore University in 1985, and M. E. in Systems Science and Automation from the Indian Institute of Science in 1987, and the Ph. D. degree in Electrical Engineering from the University of Southern California in 1991. He is now a Professor of Electrical Computer Engineering and director of the Center for Bio-Image Informatics at the University of California, Santa Barbara. His current research interests include image processing, computer vision, image and video forensics, multimedia databases and bio-image informatics. He was an Associate Editor of the IEEE Trans. Image Processing, IEEE Trans. PAMI, and IEEE Trans. Multimedia and the IEEE SP Magazine. He is a fellow of the IEEE.



Ambuj Singh is a Professor of Computer Science at the University of California at Santa Barbara. He received his B.Tech. from the Indian Institute of Technology and a PhD degree from the University of Texas at Austin. His current research interests are in graph and bioimage databases. He has written over 130 technical papers in the areas of distributed computing, databases, and bioinformatics.