Fast Tracking of Near-Duplicate Keyframes in Broadcast Domain with Transitivity Propagation

Chong-Wah Ngo
Dept of Computer Science
City University of Hong Kong
cwngo@cs.cityu.edu.hk

Wan-Lei Zhao
Dept of Computer Science
City University of Hong Kong
wzhao2@cs.cityu.edu.hk

Yu-Gang Jiang
Dept of Computer Science
City University of Hong Kong
yjiang@cs.cityu.edu.hk

ABSTRACT

The identification of near-duplicate keyframe (NDK) pairs is a useful task for a variety of applications such as news story threading and content-based video search. In this paper, we propose a novel approach for the discovery and tracking of NDK pairs and threads in the broadcast domain. The detection of NDKs in a large data set is a challenging task due to the fact that when the data set increases linearly, the computational cost increases in a quadratic speed, and so does the number of false alarms. This paper explores the symmetric and transitive nature of near-duplicate for the effective detection and fast tracking of NDK pairs based upon the matching of local keypoints in frames. In the detection phase, we propose a robust measure, namely pattern entropy (PE), to measure the coherency of symmetric keypoint matching across the space of two keyframes. This measure is shown to be effective in discovering the NDK identity of a frame. In the tracking phase, the NDK pairs and threads are rapidly propagated and linked with transitivity without the need of detection. This step ends up with a significant boost in speed efficiency. We evaluate our proposed approach against a month of the TRECVID-2004 broadcast videos. The experimental results indicate that our approach outperforms other techniques in terms of recall and precision with a large margin. In addition, by considering the transitivity and the underlying distribution of NDK pairs along time span, a speed-up of 3 to 5 times is achieved when keeping the performance close enough to the optimal one obtained by exhaustive evaluation.

Categories and Subject Descriptors

I.2.10 [**Vision and Scene Understanding**]: Video Analysis; Perceptual Reasoning; H.2.8 [**Database Application**]: Image Databases; Data Mining

General Terms

Algorithms, Design, Experimentation

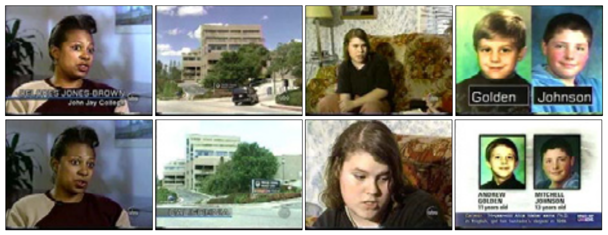


Figure 1: Examples of NDK pairs due to lighting, acquisition time, lens variations and manual editing.

Keywords

Near-Duplicate Detection, Keypoint Matching, Pattern Entropy, Keyframe Tracking, Transitivity Propagation.

1. INTRODUCTION

Near-duplicate keyframes (NDKs) are a group of keyframes that are similar or nearly duplicate of each other, but appear differently due to variations introduced during acquisition time, lens setting, lighting condition and editing operation. Figure 1 shows four pairs of NDKs undergone various changes. Previous studies in [5, 15] have demonstrated the difficulty of NDK retrieval, where the use of simple and global features such as color histogram can lead to poor performance. In this paper, we investigate a more challenging problem: the automatic and fast discovery of NDK pairs and NDK threads without any prior knowledge of the underlying data set (except knowing the videos are from the broadcast domain).

Figure 2 illustrates the difficulty of our task. Given a set of keyframes, our goal is to find the groups of NDK pairs which link up each other to form separate threads. In this figure, there are three threads of NDK groups (linked by blue, red and green arrows). To detect these threads, a straightforward approach is to exhaustively evaluate all pairs of available keyframes. Basically two keyframes with similar content are paired up, and subsequently the groups of NDK pairs are formed to become threads. However, finding a pair of NDKs is indeed a difficult task. Take a dataset of $7,000$ keyframes as an example. There are more than 24 millions of candidate pairs for comparison. Suppose among them $3,000$ pairs are near-duplicates. The chance of successfully drawing one NDK pair is only $1.22e^{-4}$. Compared with NDK retrieval [5, 15], the automatic discovery problem is challenging since there is no query to start with. Compared with keyframe clustering, the problem appears complicated since the number of non near-duplicate keyframes (outliers) is much more than the near-duplicate ones. With a large portion of data being outliers, clustering algorithms can become highly ineffective.

The findings of recent works in [5, 15] coincidentally conclude that local keypoints [6], compared with color features, are robust for NDK retrieval. A reliable measure is to match keypoints across two frames, then count the number of matching point-pairs as the similarity [5]. The cost, however, is the time spent in keypoint matching. In a keyframe of resolution 352×264, the number of keypoints typically ranges from a few to several thousands. Comparing two keyframes each of $1,000$ keypoints involves up to half a million of keypoint matchings. As a consequence, the problem of NDK discovery becomes unaffordable in terms of time and difficulty, due to the large amount candidate pairs at both keyframe and keypoint levels.

In this paper, we propose a novel approach to solve the discovery problem. The symmetric and transitive properties of near-duplicate are explored for this purpose. At the keypoint level, NDK pairs are detected by a reliable keypoint matching algorithm. While at the keyframe level, the NDK threads are rapidly propagated along the time span without going through the detection phase. The novelty of our approach lies in the proposal of pattern entropy to depict keypoint matching for NDK detection, while transitivity propagation is incorporated for fast tracking of keyframes. The contributions of this paper are summarized as follows.

- *Matching strategy.* Based on the symmetric property of NDK, we propose a one-to-one symmetric (OOS) keypoint matching strategy to reliable match keypoints across frames. In addition, an index structure LIP-IS is proposed to support the fast filtering of keypoints under OOS matching. The effectiveness of keypoint matching has also been demonstrated for keyframe retrieval and high-level concept classification in our recent work [16].

- *Matching pattern.* The matching patterns of NDK pairs formed by OOS keypoint mapping are investigated. Due to the nature of near-duplicate, the matched lines between keypoints often show patterns that are coherently matched in space. We explore this fact by evaluating the spatial coherence of patterns in both horizontal and vertical directions with a novel measure called pattern entropy (PE). The potential of PE is confirmed with our empirical evaluation on a month of TRECVID videos.

- *Keyframe tracking.* Exhaustive detection of NDK pairs is time consuming. By observing the transitivity property and the underlying distribution of NDK pairs, we propose to efficiently propagate NDK pairs and their threads without detailed investigation. The performance of propagation can be close to the optimal one obtained by the exhaustive approach.

The remaining of this paper is organized as follows. Section 2 describes the existing approaches in near-duplicate retrieval and detection. Section 3 gives an overview of our proposed framework and approach. Section 4 presents our proposed keypoint matching strategy and filtering algorithm. Section 5 proposes the measure of pattern entropy to evaluate the patterns formed by keypoint matching. Section 6 discusses the transitivity propagation of NDK. Section 7 describes the NDK retrieval experiments, while Section 8 presents the experimental results on the discovery of NDK pairs and threads with transitivity propagation. Finally, Section 9 concludes this paper.

2. RELATED WORKS

Recently, NDK identification has attracted numerous research attentions, mainly due to its unique role in news search [2], topic detection and tracking (TDT) [14] and copyright infringement detection [5]. NDKs are commonly found in broadcast videos. In TDT, NDKs provide a strong cue to link and track topic-relevant news stories across sources, languages and times. The recent work in [2] has also demonstrated the usefulness of NDKs in boosting the performance of interactive multimedia search.

The issues of duplicate and near-duplicate detection have also been addressed in image fingerprinting literature [1, 7, 11, 12]. The ultimate aim is to identify the content (or specifically fingerprint) by the image itself based on its own low-level features for retrieving the suspicious pirated copies of the image for human inspection. Our proposed work is different from image fingerprinting in two aspects. First, we use locally salient features [4, 6] to depict a keyframe content, while others [1, 7, 11, 12] mostly employ the global or local statistics of features (e.g., color) and concatenate them into a vector for retrieval. The use of single feature vector, nevertheless, tends to generate many false positives and negatives. Local features based upon keypoints, in contrast, are tolerant to the perturbation of lighting, color and certain transformations, despite the fact that additional computational cost is incurred for the need of point set matching. Second, the problem nature is different for image fingerprinting and NDK detection in the broadcast domain. The former is to identify suspicious candidates for human inspection, while the latter is to identify keyframes that nearly duplicate each other but span across the time axis and various broadcast sources.

Representative works in NDK identification include [3, 5, 15]. In [3], NDKs (or Candidate Repeating Keyframes) are detected by ordering and examining the N neighbors of a keyframe. A "jump" indicator is used to detect NDKs by investigating the first derivation of keyframe similarity. This approach is heuristic and sensitive to the setting of several empirical parameters. In [15], a stochastic attributed relational graph (ARG) matching with part-based representation is proposed for NDK identification. Under this setting, ARG is a fully connected graph with SUSAN detected keypoints as vertices, and the matching of ARGs is constrained by the spatial relation imposed by keypoints. For speed reason, a distribution-based similarity model is learnt for NDK identification. Although interesting, the approach in [15] suffers from the limitations of slow matching speed and the requirement of heuristic parameters for learning.

In contrast to [15], the PCA-SIFT descriptors of keypoints are utilized in [5] for direct point set matching without learning. To accelerate matching speed, an efficient index structure based on locality sensitive hashing (LSH) is further proposed. Nevertheless, LSH requires several user-defined parameters which have an effect on the distortion and granularity of the search.

In this paper, as in [5], we adopt direct keypoint matching, and utilize an index structure LIP-IS, which is shown to

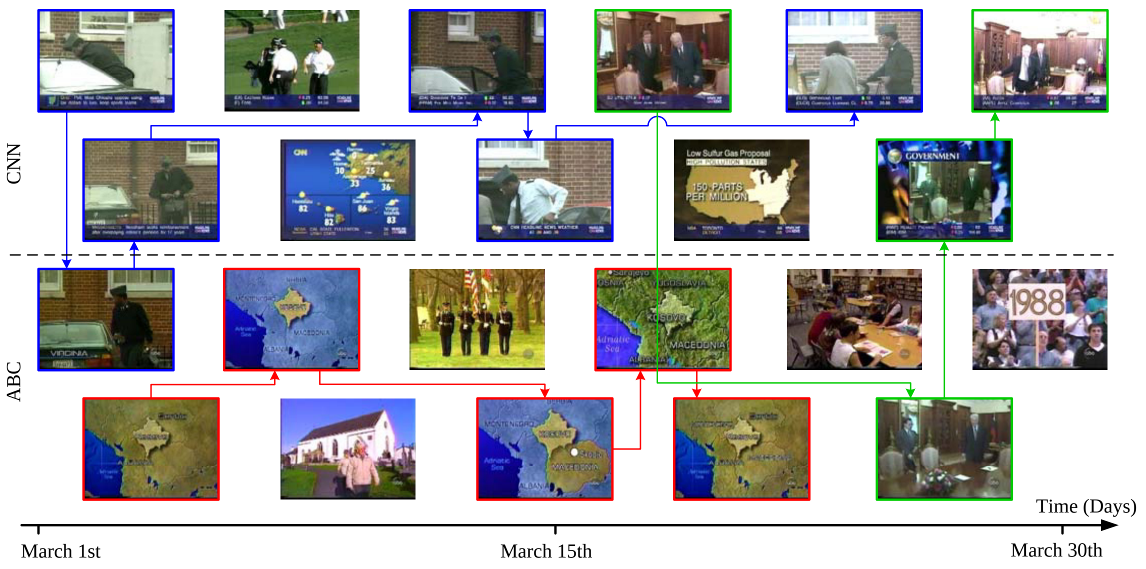


Figure 2: Automatic discovery of NDK groups from a bunch of keyframes spanned across time.

be empirically more reliable than LSH, for fast filtering. For NDK detection, instead of basing on the number of matched keypoints [5] or a learned model [15], we explore the pattern of keypoint matching with an entropy measure which is demonstrated to be highly effective without the need of learning.

3. OVERVIEW

3.1 Definition of Near-Duplicate

The definition of NDK in the broadcast domain has been previously discussed in [15]. The variations in NDKs include scene, camera setting, and photometric and digitization changes. Broadly, we can categorize the factors of "near-duplicate" into three groups: (i) same scenes and objects but being captured by the same or different cameras probably under slightly different snapshots of time; (ii) reuse of old materials, either at frame or region level, with additional editing operations, (iii) a mixture of (i) and (ii). Group (i) introduces keyframes of varying lighting, viewpoint and camera setting. Group (ii) imposes the fact that any two keyframes sharing near-duplicate regions can be declared as NDK pairs. Group (iii) can basically populate a large and diverse set of NDK pairs that are difficult to detect.

3.2 Properties of Near-Duplicates

We explore the symmetric and transitivity of NDK pairs for detection and propagation, respectively. The symmetric property states that if a keyframe k_1 is a near-duplicate of k_2, it implies that k_2 is also a near-duplicate of k_1. This property indeed affects the design of the keypoint matching strategy. In our approach, we utilize this property for the reliable matching of keypoints, resulting in an effective way of pruning false NDK pairs.

The transitive nature, to a certain extent, does exist in NDKs. Given two pairs of NDK (k_1, k_2) and (k_2, k_3), one may infer the NDK identity of (k_1, k_3). Figure 3 shows three

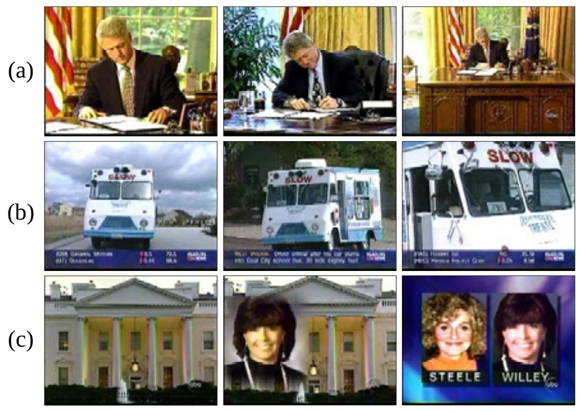


Figure 3: Transitivity of NDKs (top and middle: positive samples, bottom: negative sample).

chains of NDKs. The transitivity holds for the chains of 3(a) and 3(b), but not 3(c). The bridging keyframe in 3(c) is a near-duplicate to the other two keyframes which are indeed not an NDK pair. Interestingly, although Figure 3(c) is not an NDK chain, no doubt that it is still useful for tasks like news story threading, video search and high-level feature annotation. Based on the NDK definition, we treat 3(a) and 3(b) as the true positives of an NDK pair, while 3(c) as a false positive. The detection of transitivity violation is not considered in this paper. In our approach, the transitivity of NDK is assumed and explored to rapidly track and link the NDK threads across time without going through the detection phase.

In the broadcast domain, the distribution of NDK pairs across day difference follows the normal distribution. The peak of the distribution occurs on the same day where near-duplicate keyframes can be readily found across various channels. Figure 4 shows the distribution of NDK pairs across a span of one month (March, 1998) of broadcast news from

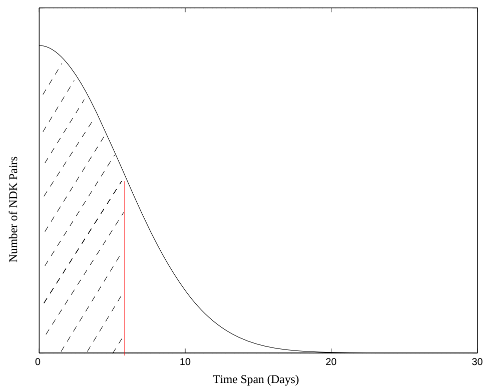


Figure 4: NDK distribution along time span.

CNN and ABC. The peak is hit for the NDKs happened on the same day. As the span increases, the number of NDKs drops exponentially and one could rarely find an NDK pair from two broadcasts with a date difference of 20 or more days. With normal distribution, we propose an approach to exhaustively find the majority of NDKs within d days, and then grow the NDKs to locate pairs with day span larger than d with transitivity propagation. Compared to the brute-force approach which evaluates all pairs of keyframes, transitivity propagation can speed up the detection process by as fast as ten times, depending on the setting of d.

3.3 Proposed Framework

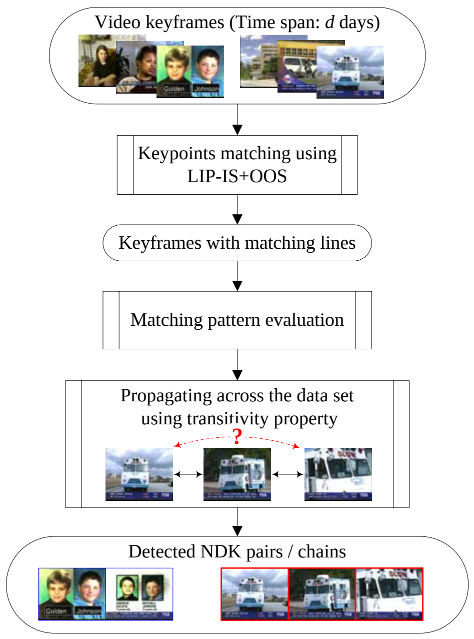
Figure 5 illustrates the overview of our framework. The framework is composed of two portions: exhaustive detection within d days and transitivity propagation. In the detection phase, the NDK identity is assessed based upon keypoint matching with the one-to-one symmetric (OOS) mapping strategy. An index structure LIP-IS is proposed for the efficient filtering of keypoints with OOS. The mapping of keypoints across frames ultimately forms patterns depicted by the matching lines of keypoints. To this end, we propose a novel method to measure the degree of matching coherency in space by evaluating the entropy of the patterns. The detected NDK pairs are then transitively grown to track the remaining parts of the threads across time span.

4. KEYPOINT MATCHING

Keypoints are salient regions detected over image scales. The descriptors of keypoints are invariant to certain transformations that exist in different images. In the current literature, there are numerous keypoint detectors and descriptors. A good survey can be found in [10, 9]. In this paper, we adopt Hessian-Affine [8] as the keypoint detector, and PCA-SIFT [4] as the descriptor.

4.1 One-to-One Symmetric Matching

Given two sets of keypoints, the alignment between them can be solved with bipartite graph matching algorithms. Depending on the mapping constraint being imposed, we can categorize them as many-to-many (M2M), many-to-one (M2O), one-to-many (O2M) and one-to-one (O2O) matching. The factors that affect the choice of matching strategy include noise tolerance, similarity measure, matching effectiveness and efficiency. In videos, frames always suffer from



Video keyframes (Time span: d days)
Keypoints matching using LIP-IS+OOS
Keyframes with matching lines
Matching pattern evaluation
Propagating across the data set using transitivity property
?
Detected NDK pairs / chains

Figure 5: Framework Overview.

low-resolution, motion-blur and compression artifact. Noise becomes a crucial factor in selecting a matching algorithm, particularly when the matching decision is made upon the small local patches surrounding keypoints. Noise can affect the performance of keypoint detectors [6]. Localization errors caused by detectors can deteriorate the distinctiveness of PCA-SIFT. It becomes common that a keypoint fails to find its nearest neighbor in another keyframe, and on the other extreme, a keypoint can simply be matched to many other keypoints due to mapping ambiguity. In principle, to suppress faulty matches, O2O matching appears to be noise tolerant although some correct matches may be missed.

For effective keyframe retrieval, the matching algorithm should filter out as many false matches as possible. To retain only the most reliable matches for retrieval, we introduce a new scheme – namely one-to-one symmetric (OOS) matching. OOS ensures all the matches are the nearest neighbors. The symmetric property is also emphasized so that if keypoint P matches to Q, then P is the nearest neighbor of Q (i.e., $P \rightarrow Q$) and similarly $P \leftarrow Q$. This property indeed makes OOS stable and unique, i.e., the result of matching a keypoint set A to set B is exactly the same as B to A, unless there are keypoints that have more than one nearest neighbor. Generally speaking, O2O matching cannot guarantee each matched keypoints pair to be meaningful. Some false matches indeed could exist with high similarity value. However, it becomes a rare case for these false matches to be symmetrically stable and paired to each other in both directions.

4.2 Fast Keypoint Filtering

Point-by-point matching between two sets is generally a time consuming task especially when the set cardinality is high. To allow fast retrieval of OOS, we perform an approximate nearest neighbor search by indexing PCA-SIFT descriptors in a multi-dimensional structure called LIP-IS. The structure is a group of 36 histograms formed independently

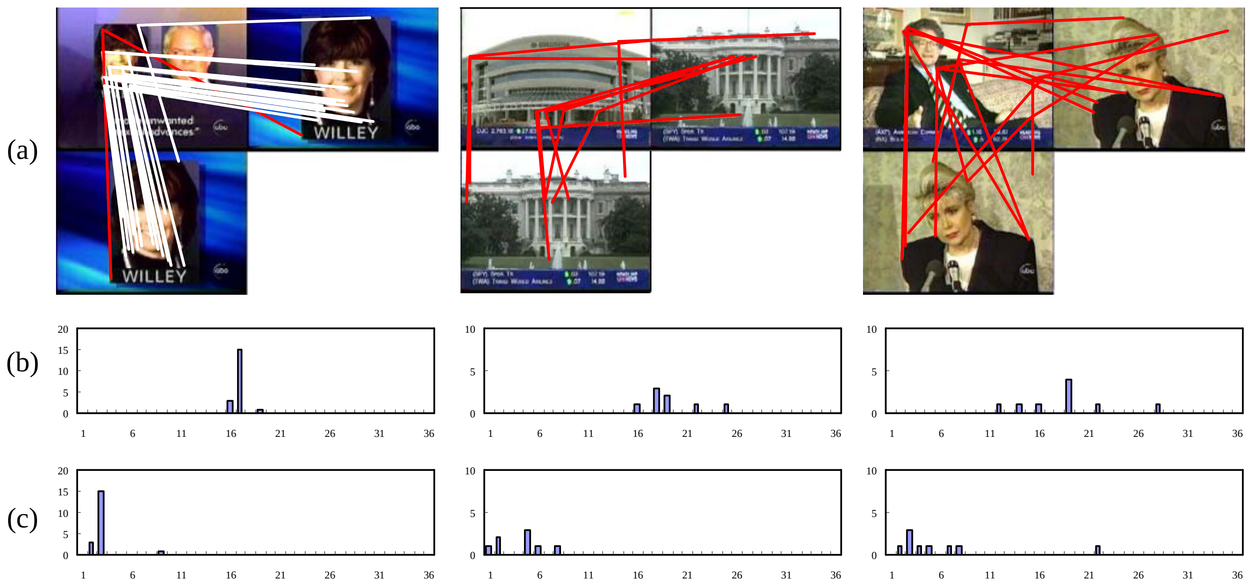


Figure 6: Histograms of matching patterns for three keyframe pairs: (a) matching lines between keypoints, (b) vertical histogram $\mathcal{G}_v$, (c) horizontal histogram $\mathcal{G}_h$. (White and red lines indicate the correct and false keypoint matches, respectively.)

by every components of PCA-SIFT. LIP-IS is constructed by equally and independently quantizing each histogram into 8 bins, with a resolution of $\Delta = 0.25$ (the range of a PCA-SIFT components is [-1,1]). Given $P = [p_1, p_2, ..., p_i, ..., p_{36}]$, the index of P in dimension i is defined as

$$\mathcal{H}(p_i) = \lfloor \frac{p_i + 1}{\Delta} \rfloor \tag{1}$$

In total, this index structure is composed of 8×36 bins. During indexing, a keypoint P is repeatedly indexed into the corresponding bins of 36 histograms, according to its quantized value in a particular dimension. Thus, each keypoint is hashed and then distributed into 36 bins in this structure. In principle, the structure encodes the keypoints of a keyframe by decomposing the PCA-SIFT components and modeling them as 36 independent distributions. This structure is intuitive and reasonable since the PCA-SIFT components are orthogonal to each other. Based on this structure, we define the function that any two keypoints P and Q collide in dimension i if

$$\mathcal{C}(q_i, p_i) = \begin{cases} 1 & \text{if } |\mathcal{H}(q_i) - \mathcal{H}(p_i)| \le 1 \\ 0 & \text{Otherwise} \end{cases} \tag{2}$$

When searching for the nearest neighbor of a query keypoint Q, the structure will return a candidate set $\mathsf{A}(Q)$, which includes the points that collide with Q across all the 36 dimensions. Then we search for Q's nearest neighbor from the set $\mathsf{A}(Q)$ by the OOS matching algorithm. This structure has the merit that it is efficient and easy to implement with simple bit operation.

With LIP-IS, basically two keypoints, which are similar to each other, are more likely to collide in every dimension. In contrast, the dissimilar keypoints have a relatively lower chance of collision. Since each component of the PCA-SIFT descriptors is theoretically Gaussian distributed, the probability that any two keypoints collide in a dimension can be estimated. The probability that a keypoint Q will collide with P in dimension i, in its best ($\mathbf{P}_b$) and worst ($\mathbf{P}_w$) cases, can be estimated as follows

$$\mathbf{P}_b = 2\int_0^{2\Delta} \frac{1}{\sqrt{2\pi\sigma_i}} \exp\{-\frac{q_i^2}{4\sigma_i^2}\}\, dq_i \tag{3}$$

$$\mathbf{P}_w = 2\int_0^{\Delta} \frac{1}{\sqrt{2\pi\sigma_i}} \exp\{-\frac{q_i^2}{4\sigma_i^2}\}\, dq_i \tag{4}$$

Then, the probability that a point will collide with Q in 36 dimensions can be expressed as

$$\mathbf{P}_f = \mathbf{P}_b^{36} \tag{5}$$

Notice that $\mathbf{P}_f \ll 1$ in general. This also implies that the cardinality of $A(Q)$ can be very small, i.e., $\mathbf{P}_f \times n$, where n is the total number of keypoints to be searched.

The probability of missing the nearest neighbor can also be estimated. Suppose M is the maximum number of dimensions that the nearest neighbor $\hat{P}$ and Q will not collide, the probability in worst case is

$$\mathbf{P}_{miss} = \sum_{i=1}^{M} \begin{pmatrix} M \\ i \end{pmatrix} \mathbf{P}_w^{M-i}(1 - \mathbf{P}_w)^i = 1 - \mathbf{P}_w^M \tag{6}$$

In theory, M can be estimated (and this value is much smaller than 36), if we set a threshold to exclude keypoints with low similarity from consideration. In our simulation, when searching for a nearest neighbor from a 1000 keypoint set, LIP-IS is often capable of filtering 99.5% of the points without missing the real candidate for OOS matching.

5. ENTROPY OF MATCHING PATTERN

Intuitively, two matched keypoints indicate a pair of duplicate sub-regions between two frames. Since NDK pairs share the duplication of scenes, the upshot of keypoint matching should form certain spatially coherent patterns that can distinguish themselves from random matching. These patterns could be one or several bunches of parallel or zoom-like

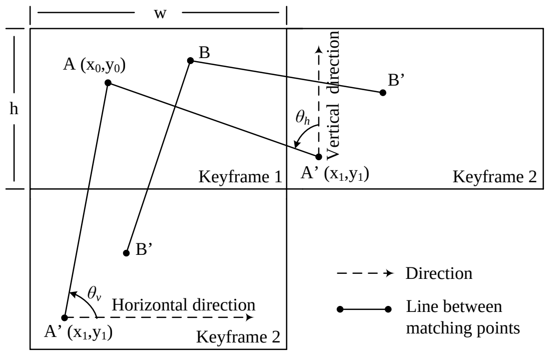


Figure 7: Computing orientation of matching lines.

matching lines across the sub-regions of the keypoints. Figure 6(a) shows the examples of matching patterns for NDK (left) and non-NDK (middle and right) pairs. Basically, the matching of non-NDK pairs often show random patterns with matching lines being arbitrarily crossed across space. The matching formats, in principle, provides vivid pattern cues for the discrimination of NDK and non-NDK pairs. In this section, we propose a novel measure, called pattern entropy (PE), to measure the information for being an NDK pair.

We capture the matching patterns of keyframes with two histograms of matching orientations. Histograms $\mathcal{G}_h$ and $\mathcal{G}_v$ are constructed, respectively, by aligning two frames horizontally and vertically, as shown in Figure 6(a) and Figure 7. Depending on the alignment, a histogram is composed of the quantized angles formed by the matching lines and horizontal or vertical axis. Denote h as the height of the upper keyframe (keyframe-1 in Figure 7), and the coordinates of keypoint A in Keyframe-1 and keypoint A' in Keyframe-2 as (x_0, y_0) and (x_1, y_1), respectively. The angle θ_v of a line formed by two matched keypoints is computed as follows

$$\theta_v = \arccos\left(\frac{x_1 - x_0}{\sqrt{(x_1 - x_0)^2 + (y_1 + h - y_0)^2}}\right) \tag{7}$$

Histogram $\mathcal{G}_v$ is formed by computing θ_v of lines and then accumulating the count to the corresponding bins. Histogram $\mathcal{G}_h$ is computed in a similar manner by the angle θ_h. Denote w as the width of the left keyframe (keyframe-1 in Figure 7), the angle θ_h is computed as

$$\theta_h = \arccos\left(\frac{x_1 - x_0}{\sqrt{(x_1 + w - x_0)^2 + (y_1 - y_0)^2}}\right) \tag{8}$$

We quantize the histograms into 36 bins with a step of 5^o from 0^o to 180^o. Ideally, the parallel or nearly parallel lines should fall in the same bin of a histogram. Histograms $\mathcal{G}_h$ and $\mathcal{G}_v$ intuitively suggest the spatial coherency of matching in the horizontal and vertical directions. Figures 6(b) and 6(c) shows the histograms corresponding to an NDK (left) and two non-NDK pairs (middle and right). The distributions of $\mathcal{G}_h$ and $\mathcal{G}_v$ depict the different partitions of orientation for the same set of matched keypoints. For an NDK pair, both histograms should be correlated. Specifically, whenever a peak in $\mathcal{G}_h$ is found, there should exist a corresponding peak of the same keypoints in $\mathcal{G}_v$. To reveal the mutual information between $\mathcal{G}_h$ and $\mathcal{G}_v$, we use entropy to measure the homogeneity of histogram patterns. For abbreviation, we call our measure Pattern Entropy (PE).

Denote N as the number of bins in a histogram, and define $\mathbf{P} = [p_1, p_2, \ldots, p_m]$ and $\mathbf{Q} = [q_1, q_2, \ldots, q_n]$, where $m \le n \le N$. The notation p_i (similarly for q_i) is a non-empty set of keypoint pairs that falls in a bin of histogram. Physically, $\mathbf{P}$ corresponds to one of the histograms ($\mathcal{G}_h$ or $\mathcal{G}_v$) with less non-empty bins, while $\mathbf{Q}$ corresponds to the other histogram. In principle, $\mathbf{P}$ is more compact than $\mathbf{Q}$ since less bins are used to accommodate the matched keypoints. In pattern entropy, we measure the degree of points in p_i being distributed in $\mathbf{Q}$, defined as

$$\mathrm{PE}(\mathbf{Q}, \mathbf{P}) = \frac{1}{S} \sum_{q_i \in \mathbf{Q}} \mathrm{Entropy}(q_i, \mathbf{P}) \tag{9}$$

where

$$\mathrm{Entropy}(q_i, \mathbf{P}) = -\frac{1}{\log m} \sum_{p_j \in \mathbf{P}} \frac{|q_i \cap p_j|}{|q_i|} \times \log \frac{|q_i \cap p_j|}{|q_i|}$$

$$\mathrm{S} = \sum_{q_i \in \mathbf{Q}} |q_i| = \sum_{p_i \in \mathbf{P}} |p_i|$$

and $|p_i \cap q_i|$ is the cardinality of intersection between two sets p_i and q_i. Basically, Entropy$= [0, 1]$ measures the extent of dispersing a set p_i across the bins of another histogram of the orthogonal direction. An entropy value of 0 indicates the keypoints in p_i are found exactly in another set q_i of $\mathbf{Q}$. A value of 1 indicates that the keypoints in p_i are evenly distributed in some sets of $\mathbf{Q}$. Overall, PE$= [0, 1]$. The extreme value of PE$= 0$ indicates a perfect coherent match in both the horizontal and vertical directions. Conversely, the value of PE$= 1$ basically hints a random match across space. PE indeed bears two interesting facts. First, the NDK and non-NDK pairs can be distinguished according to the degree of matching coherency (and randomness) in space formed by keypoints. Second, the measure fits the symmetric property of NDK pairs due to the selection of $\mathbf{Q}$ for testing the dispersion of q_i with respect to $\mathbf{P}$.

In Eqn (9), to restrict the random matches from being considered, practically we should only consider p_i with high enough cardinality. For this purpose, a parameter γ is set to decide whether a set p_i should participate in PE evaluation (i.e., Eqn (9) is computed based upon all $p_i \ge \gamma$). We will demonstrate in our experiments that γ is not sensitive as long as its value is not small so as to eliminate noisy matches. Finally, to determine whether two compared keyframes are NDK, a gating threshold is set with PE$= 0.5$. Thus, in the evaluation, keyframes with PE≤ 0.5 are declared as NDK pairs.

6. TRANSITIVITY PROPAGATION

Based on the transitivity property, we propose to detect NDK pairs within d days. The pairs with time span greater than d are tracked by propagating the transitivity relationship along the time dimension. Figure 8 illustrates an example with $d = 3$. The black arrows mark the detected NDKs in an exhaustive manner, while the dotted red arrows indicate the propagated pairs. Both types of NDK pairs are grouped to form a thread.

To calculate the speed-up due to transitivity propagation, we denote T as the total number of days under investigation, and μ as the average number of keyframes per day. For an

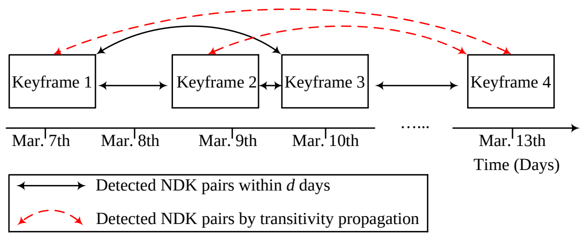


Figure 8: Transitivity propagation.

exhaustive evaluation, the number of keyframe comparisons is

$$N_e = \frac{T \times (T+1) \times \mu^2}{4} \tag{10}$$

When propagation is used, the amount of comparisons is

$$\begin{aligned} N_t &= ((T-d) \times (d+1) + \frac{d \times (d+1)}{2}) \times \frac{\mu^2}{2} \\ &= (d+1) \times (2T-d) \times \frac{\mu^2}{4} \end{aligned} \tag{11}$$

Note that the number of comparisons within the same day is indeed $\frac{\mu \times (\mu-1)}{2}$. We approximate this value with $\frac{\mu^2}{2}$ in both equations. The amount of speed-up by comparing Eqn (10) and Eqn (11) is

$$\text{Speed-up} = \frac{T \times (T+1)}{(d+1) \times (2T-d)} \tag{12}$$

Let $T = 30$, the speed-up is approximately 5.34 times for $d = 2$, and 2.82 times for $d = 5$.

7. RETRIEVAL EXPERIMENT

The aim of this section is to justify the choice of keypoint detector and the performance of OOS matching and LIP-IS. We use the data set given by [15] for evaluation. The data set consists of 600 keyframes with 150 NDK pairs. Notice that in this dataset each keyframe is near-duplicate with at most one keyframe, and thus transitivity propagation is not applicable for the experiments in this section.

We use all NDKs (300 keyframes) as queries for NDK retrieval in the experiments. The retrieval performance is evaluated with the probability of the successful top-k retrieval, defined as

$$\mathbf{R}(k) = \frac{F_c}{F_a} \tag{13}$$

where F_c is the number of queries that find its duplicate in the top-k list, and F_a is the total number of queries. In the retrieval experiments, the ranking is based on the cardinality of keypoints being matched. In case the cardinality is the same, the average similarity of matched keypoints is further used.

Figure 9 compares the performance of the one-to-one symmetric (OOS) and many-to-one (M2O) matching strategies, against the baseline using color moment (CM). For OOS, we further contrast the performance of two keypoint detectors: Hessian-Affine and Difference of Gaussian (DoG). For M2O mapping, we adopt the nearest neighbor search proposed in [6]. The DoG+M2O strategy has also been used by [5] for near-duplicate detection. For CM, three color moments (i.e., mean, standard deviation, skewness) are computed. Basically each keyframe is divided in 5×5 grids, and

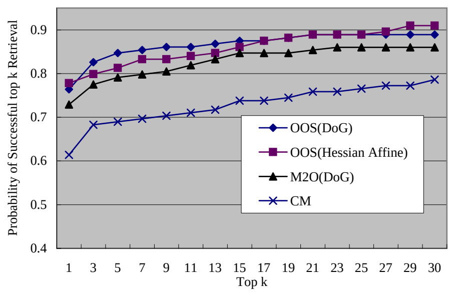


Figure 9: Comparison of keypoint detectors and mapping strategies.

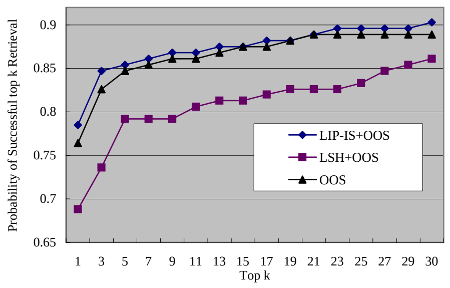


Figure 10: Comparison of LIP-IS and LSH.

the color moments are computed for each grid in *Lab* color space. As shown in this figure, OOS outperforms M2O due to its ability in pruning false alarms. The performance of Hessian-Affine is slightly lower, but close to, the DoG detector. Considering the average number of keypoints detected in a keyframe, however, Hessian-Affine (average of 600 keypoints) has advantage over DoG (average of 1,200 keypoints) with a relatively smaller data size.

Figure 10 assesses the performance of two index structures: the proposed LIP-IS and the locality sensitive hashing (LSH) used in [5]. There are two critical parameters for LSH: K (number of random partition) and L (number of times to tessellate a set). We optimize these parameters with respect to the data set and set $K = 108$ and $L = 2$. The experimental results justify the performance of LIP-IS where it constantly outperforms LSH across all the tested top-k retrieval. LIP-IS is about twelve times faster than pure OOS, however, is still slower than LSH for about two times. Comparing our results with [15] on the same dataset, where $\mathbf{R}(k) = 0.6$ when $k = 1$ and $\mathbf{R}(k) = 0.76$ when $k = 30$, the performance of our proposed LIP-IS+OOS shows great improvement.

8. DISCOVERY EXPERIMENT

This section justifies the performance of discovering NDK pairs and threads, as well as the speed improvement due to transitivity propagation. We present our empirical findings separately in four parts. First, a baseline comparison is given to justify the performance of pattern entropy with brute-force keyframe comparison. Second, the effect of transitivity propagation is investigated by presenting the performance gap between the results obtained by propagation and brute-force comparison. Third, we assess the ability of pattern entropy and transitivity propagation in discovering the

Table 1: Top five frequently reported news topics in CNN and ABC channels during the whole month of March 1998.

Theme	# of Stories	# of NDK Pairs
Basketball	78	73
Clinton Sexual Scandal	58	250
Finance	53	159
El nino	38	12
Arkansas school shooting	37	149

NDK threads of news videos. Finally, we investigate the degree of speed improvement by varying the setting of time span in transitivity propagation.

We use one month of TRECVID 2004 video corpus for performance evaluation. The data covers 52 broadcasts of CNN and ABC in March of 1998. Basically two pieces of news, one from CNN and the other from ABC, were reported each day. In total, there are 29 themes covering 805 news stories in the videos. One representative keyframe (given by TRECVID) was extracted for each shot, resulting in 7,006 keyframes. These keyframes form $24,538,515$ candidate pairs. Among them, $3,388$ NDK pairs were manually identified by three assessors and finally became the ground-truth dataset for experiments. The NDK pairs form 693 NDK threads, and involve a total of $1,953$ keyframes. The proportion of NDKs to all keyframes is approximately 28%. Considering the number candidate pairs, the chance of successfully finding a correct NDK pair in random, however, is only $1.38e^{-4}$.

Table 1 lists the top five most frequently reported news themes, along with their NDK pairs, in the ground-truth. During the assessment, to avoid the potential ambiguity in NDK pairs, two assessors were asked to mark the NDK pairs separately. Another assessor was then invited to compare the two sets of NDK pairs, compile and then group these NDK pairs into the final ground-truth sets. Careful, although subjective, judgment was required for the third assessor whenever there was a conflict of an NDK pair.

For performance evaluation of NDK pair discovery, we use Precision, Recall and F-measure, defined as

$$\text{Recall} = \frac{\text{Number of NDK pairs correctly detected}}{\text{Total Number of NDK pairs}} \quad (14)$$

$$\text{Precision} = \frac{\text{Number of NDK pairs correctly detected}}{\text{Number of detected NDK pairs}} \quad (15)$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (16)$$

Recall measures the accuracy of returning ground-truth NDK pairs, while precision assesses the ability of excluding false positives. F-measure calculates the fitness of ground-truth and detected NDK pairs by jointly considering recall and precision.

8.1 Baseline Comparison

The baseline performance is grounded on brute-force (exhaustive) comparison. In other words, a total of $24,538,515$ candidates pairs are exhaustively compared. A decision is then made to gate whether a candidate belongs to an NDK pair. We compare three approaches: pattern entropy (PE), cardinality threshold (CT), and color moment (CM). PE and CT are based on keypoint matching using LIP-IS+OOS mapping, while CM is based on color feature. In CT, a gating threshold (η), which is the cardinality of matched keypoints, is set for determining NDK pairs. This strategy is also used in [5]. In principle, a candidate should have a large enough number of keypoints being matched in order to claim the NDK identity. In CM, the similarity of candidate pairs are compared and then ranked in descending order. The top-k pairs are then declared as NDK pairs.

Table 2: Baseline performance comparison.

	Pattern Entropy	Cardinality Threshold	Color Moment
Precision	0.892	0.829	0.151
Recall	0.779	0.696	0.151
F-measure	0.832	0.757	0.151

Table 3: Performance of NDK detection with pattern entropy (γ: minimum required cardinality in each bin).

	$\gamma = 3$	$\gamma = 4$	$\gamma = 5$	$\gamma = 6$	$\gamma = 7$
Precision	0.626	0.824	0.892	0.914	0.931
Recall	0.829	0.808	0.779	0.745	0.714
F-measure	0.713	0.816	**0.832**	0.821	0.81

Table 2 shows the performance comparison of the three approaches. For the proposed measure PE, the minimum cardinality γ is set to 5. For CT, $\eta = 12$, and for CM, we show the results for top-3388 retrieved pairs (k is equal to the number of ground-truth NDK pairs). Experimental results indicate that the proposed PE outperforms CT across all performance measures. By taking into account the spatial coherency of matching patterns between keypoints, PE shows approximately a 10% improvement in term of F-measure over CT. Note that by using keypoints, there are approximately 300 NDK pairs with matching cardinality less than 3. In other words, PE shows excellent recall performance, and the majority of false negatives indeed belong to pairs with very few matched keypoints. Both PE and CT show significantly better performance than CM, which demonstrates the advantage of keypoints over color features for NDK detection.

To show the sensitivity of parameter setting, the performance of the three approaches, by adjusting their parameters (γ, η, k), are experimented. As shown in Table 3, PE performs consistently satisfactory across all $\gamma \geq 4$, and achieves the best result at $\gamma = 5$. It is not hard to understand that precision improves while recall degrades, when

Table 4: Performance of NDK detection with cardinality threshold η.

	$\eta = 8$	$\eta = 10$	$\eta = 12$	$\eta = 16$	$\eta = 18$
Precision	0.150	0.528	0.829	0.959	0.97
Recall	0.811	0.754	0.696	0.598	0.558
F-measure	0.253	0.621	**0.757**	0.736	0.708

Table 5: Performance comparison with transitivity propagation.

	Pattern Entropy ($\gamma = 5$)			Cardinality Threshold ($\eta = 12$)		
Span (day)	Precision	Recall	F-measure	Precision	Recall	F-measure
1	0.910	0.369	0.525	0.893	0.341	0.493
2	0.884	0.482	0.624	0.681	0.412	0.510
3	0.818	0.700	0.754	0.547	0.554	0.551
4	0.832	0.717	0.770	0.443	0.644	0.524
5	0.824	0.739	0.779	0.428	0.661	0.520
6	0.831	0.765	0.797	0.399	0.723	0.540
7	0.813	0.781	0.797	0.348	0.733	0.472
Brute-force	0.892	0.779	0.832	0.829	0.696	0.757

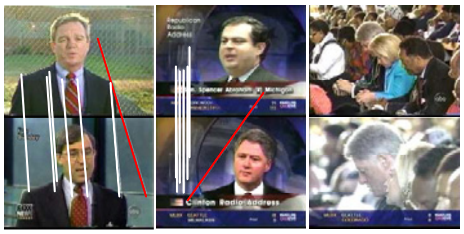

(a) False-alarm (b) False-alarm (c) Missed pair

Figure 11: False positives and true negative. (White and red lines indicate the correct and false keypoint matches, respectively.)

the value of γ increases. The aim of setting γ is to restrict the random matches of keypoints from consideration. The higher value γ is, the better chance of finding an NDK pair, although certain true positives may be accidentally missed due to the restriction. Figure 11 shows a few examples of false positives and true negatives by PE. In our experiments, most false alarms are indeed due to the similar background scene (see Figure 11(b)) and partly because of the editing effects. The missed pair in Figure 11(c) is due to sharp lighting change and camera zoom, resulting in no matched keypoint being found. Compared with PE, CT demonstrates a relatively unstable performance as shown in Table 4. Similar to PE, when the value of η becomes larger, precision improves and approaches to 100% when $\eta = 18$. The improvement, nevertheless, comes with the price of low recall. The result of CT indeed indicates that when the cardinality of matched keypoints of a candidate exceeds a certain value, the NDK identity is almost certain. The difficulty of NDK detection, however, should not be overlooked since the cardinality can range from as few as three to as large as several hundreds for true positives. As a consequence, the recall and precision of CT can fluctuate significantly even with a small change of threshold η. CM, in contrast to PE and CT, performs poorly with low recall values for all the tested top-k retrieval.

8.2 Tracking with Transitivity

Table 5 shows the performance of NDK detection with transitivity propagation. We test the time span ranging from one day to one week. In our data set, the brute-force approach is basically equivalent to the time span of one month. With transitivity propagation, NDK pairs within a time span are exhaustively searched with LIP-IS+OOS. The detected NDK pairs are then transitively grown to form more pairs. We compare two approaches, PE and CT, with the best parameter setting found in the Section 8.1. As indicated in Table 5, as the time span becomes larger, the performance approaches to the brute-force one. The slope of improvement, based on F-measure, is gradually less significant when time span is equal or greater than four days. This probably implies that the time span of four days is a good choice when transitivity propagation is employed. With the time span, the NDK pairs being correctly tracked with transitivity is 937. Meanwhile, 357 pairs are falsely generated. For PE, compared to brute-force detection, the degradation in F-measure is about 0.07 when day span is equal to four.

Comparing the performance of PE and CT, apparently PE is significantly better than CT across all the tested time spans. When day span is equal to three, PE already outperforms all the F-measure values of CT with reasonably good recall and precision. The performance is indeed close to the results of CT with brute-force detection. This again demonstrates the performance stability of PE over CT when the spatial coherency of matched keypoints are utilized.

Table 6: Performance of NDK thread discovery.

	PE ($\gamma = 5$)	CT ($\eta = 12$)
Cluster entropy	0.0228	0.116
F-measure	0.688	0.551

8.3 Thread Discovery

To assess the performance of discovering NDK threads, we compare the detected threads and ground-truth threads with two measures: average cluster entropy and F-measure [13]. Both measures are widely adopted in evaluating the performance of clustering. Cluster entropy measures the homogeneity of threads. The higher the homogeneity of a thread, the lower the entropy, and vice versa. On the other hand, F-measure assesses the quality of thread by intersecting a detected thread with a ground-truth thread, which offers the highest F-measure value. The higher the F-measure, the better the performance. Table 6 shows the performance of PE and CT. The proposed PE shows a significantly better performance than CT in the brute-force approach.

8.4 Speed Efficiency

Currently, the average time for the comparison of one keyframe pair with LIPS-IS+OOS is approximately 0.08 sec on a Pentium-4 3G machine, including the time to upload keyframes and save results. For one month of TRECVID

Table 7: Speed efficiency of NDK pair detection with transitivity propagation.

Span (day)	1	2	3	4	5	6	7	Brute-force
Keyframe comparison	2,463,203	4,001,612	5,453,222	6,894,733	8,225,922	9,484,087	10,711,597	24,538,515
Actual Speed (hour)	55	89	121	153	183	211	238	545
Speed-Up	9.97	6.13	4.50	3.56	2.98	2.59	2.29	1

videos, the brute-force approach requires nearly one month (about 23 days) to compute all pair combination of keyframes. Transitivity propagation is regarded as an important factor to accelerate the computation. Table 7 lists the speed-up due to transitivity propagation in terms of the number of keyframe comparisons, and the actual running time spent in obtaining the final results. When the time span is equal to one day, the speed-up is about ten times of that of the brute-force approach, which is considered as a significant boost in speed efficiency. In other words, instead of spending 23 days for an exhaustive comparison, we only need about 2.3 days to complete the task. Nevertheless, to seek a balance between efficiency and accuracy, a reasonable choice is the day span of 3 or 4, which can end up with about 4.5 or 3.5 times of speed-up. As a result, basically one needs approximately 5 days ($d = 3$) or 1 week ($d = 4$) to obtain a performance that is comparable to the brute-force evaluation.

9. CONCLUSION AND DISCUSSION

We have presented our approach for fast detection and tracking of NDK pairs and threads by utilizing the nature of symmetric and transitivity. In NDK detection, we have proposed OOS and LIP-IS, which are shown to be reliable and fast. The measure of pattern entropy is further proposed to confirm the NDK identity by evaluating the spatial coherency of matching patterns due to keypoint mapping. In NDK tracking, the feature of transitivity propagation is successfully demonstrated, with its capability in linking NDKs rapidly without causing significant degradation in recall and precision.

With propagation, nevertheless, false positives do appear whenever the property of transitivity is violated as indicated in Figure 3(c). In our view, the false NDK pairs due to violation can be very helpful for tasks like news story threading. The explicit detection of transitivity violation is possible, although not considered in this paper. Considering that the number of NDK pairs (to be further inspected for detecting violation) is far less than the number of candidate pairs, the computational saving due to transitivity propagation can be significant. In addition, with the employment of local keypoint matching, some false positives are indeed caused by the captions and logos in the keyframes. Captions and logos generate quite a number of keypoints, resulting in false and noisy matches. We believe the performance of keypoint-based detection and tracking approaches can be further improved with certain pre-processing steps such as the removal of captions and logos prior to NDK detection.

Acknowledgements

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 118905).

10. REFERENCES

[1] C. Chang, J. Wang, C. Li, and G. Wiederhold. RIME: A replicated image detector for the world-wide web. In *Multimedia Storage and Archiving Systems*, 1998.

[2] S. F. Chang, W. Hsu, L. Kennedy, L. Xie, A. Yanagawa, E. Zavesky, and D.-Q. Zhang. Columbia university trecvid-2005 video search and high-level feature extraction. In *TRECVID*, 2005.

[3] P. Duygulu, J.-Y. Pan, and D. A. Forsyth. Towards auto-documentary: Tracking the evolution of news stories. In *ACM Multimedia Conference*, pages 820–827, 2004.

[4] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *CVPR*, volume 2, pages 506–513, 2004.

[5] Y. Ke, R. Suthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia Conference*, pages 869–876, 2004.

[6] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal on Computer Vision*, 60(2):91–110, 2004.

[7] Y. Meng, E. Chang, and B. Li. Enhancing dpf for near-replica image recognition. In *CVPR*, 2003.

[8] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *ECCV*, 2002.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on PAMI*, 27(10), 2005.

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. Journal on Computer Vision*, 65(1/2):43–72, 2005.

[11] M. Schneider and S. F. Chang. A robust content based digital signature for image authentication. In *Int. Conf. on Image Processing*, 1996.

[12] J. S. Seo, J. Haitsma, T. Kalker, and C. D. Yoo. A robust image fingerprinting system using Randon transform. *Signal Processing: Image Communication*, 19:325–339, 2004.

[13] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[14] X. Wu, C.-W. Ngo, and Q. Li. Threading and autodocumenting news videos. *Signal Processing Magazine*, 23(2):59–68, Mar 2006.

[15] D.-Q. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *ACM Multimedia Conference*, pages 877–884, 2004.

[16] W. Zhao, Y. G. Jiang, and C. W. Ngo. Keypoint retrieval by keypoints: Can point-to-point matching help? In *Conf. on Image and Video Retrieval*, 2006.