# Near-Duplicate Keyframe Identification with Interest Point Matching and Pattern Learning

Wan-Lei Zhao, Chong-Wah Ngo*, Hung-Khoon Tan, Xiao Wu

***Abstract*— This paper proposes a new approach for near-duplicate keyframe (NDK) identification by matching, filtering and learning of local interest points (LIPs) with PCA-SIFT descriptors. The issues in matching reliability, filtering efficiency and learning flexibility are novelly exploited to delve into the potential of LIP-based retrieval and detection. In matching, we propose a one-to-one symmetric matching (OOS) algorithm which is found to be highly reliable for NDK identification, due to its capability in excluding false LIP matches compared with other matching strategies. For rapid filtering, we address two issues: speed efficiency and search effectiveness, to support OOS with a new index structure called LIP-IS. By exploring the properties of PCA-SIFT, the filtering capability and speed of LIP-IS are asymptotically estimated and compared to locality sensitive hashing (LSH). Owing to the robustness consideration, the matching of LIPs across keyframes forms vivid patterns that are utilized for discriminative learning and detection with support vector machines. Experimental results on TRECVID-2003 corpus show that our proposed approach outperforms other popular methods including the techniques with LSH in terms of retrieval and detection effectiveness. In addition, the proposed LIP-IS successfully speeds up OOS for more than 10 times and possesses several favorable properties compared to LSH.**

## I. Introduction

Recently, near-duplicate keyframe (NDK) identification has attracted numerous research attentions, mainly due to its unique role in news search [1], topic detection and tracking (TDT) [2] and copyright infringement detection [3]. NDKs, by definition, are keyframes near-duplicate to each other despite the slight to moderate degree of variations caused by lighting, viewpoint, acquisition time, motion and editing effects. 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 detection of NDK is critically useful for the exploitation of story redundancy and novelty in news topic threading. As experimented in [2], utilizing both NDK and text cues for modeling topic structure can achieve $10-20\%$ of improvement compared to text-only approach. The recent work in [1] also demonstrates the usefulness of NDKs in boosting the performance of interactive multimedia search.

NDK identification is a challenging task due to a variety of capturing, digitalization and editing conditions, which result in a serial of geometric and photometric transformations. Figure 1 shows four pairs of NDKs varied in terms of lighting, viewpoint, color, contrast and camera lens. Generally speaking, low-level global features cannot guarantee the effective identification of NDKs. Color features, for instance, can easily cause errors especially when lighting conditions change and the special effects are added to keyframes. Compared to global features, local features, which are distinctive and robust to changes due to different transformations and operations, appear to be prominent for NDK identification. One excellent example is the texture descriptors computed upon local interest points (LIPs) [4], [5]. LIPs (also referred to as keypoints) are salient regions detected over image scales, and their descriptors are basically features extracted from the local patches centered around LIPs at given scales. The capability of LIPs has not been fully exploited in multimedia retrieval. A success story is video google search [6] where over thousands of LIPs are detected and clustered as visual keywords for object-based retrieval and mining. Currently, there are numerous LIP detectors and descriptors in the literature. The recent studies in [5], [7] have evaluated the performance of different detectors and descriptors.

Fig. 1. Examples of NDK pairs due to lighting, viewpoint, color and lens variations

In this paper, we adopt Lowe's difference of Gaussian (DoG) detector [4] for detecting LIPs. This detector is scale invariant and can tolerate certain amount of affine transformation. For LIP representation, we use PCA-SIFT descriptor [8] which is a compact version of SIFT (scale-invariant feature transform) feature [4] undergone principal component analysis (PCA). PCA-SIFT has been shown to be distinctive and robust to color and photometric changes [3]. Although LIP-based descriptors are ideal for near-duplicate identification, the large amount of LIPs available for matching can actually overwhelm their potential. Typically a keyframe of resolution $352 \times 264$ can have more than one thousand LIPs. With this amount, comparing two keyframes can take up a million of LIP comparisons in the high dimensional feature space. To cope with the speed issue, filtering support is critically

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).

The authors are with the Department of Computer Science, City University of Hong Kong.

Chong-Wah Ngo is the contact author. Please direct all enquiries to C. W. Ngo by Email: cwngo@cs.cityu.edu.hk or Tel: 852-2784-4390 or Fax: 852-2788-8614.

demanded for suppressing the number of comparisons down to a minimum level.

This paper addresses the issues of LIP matching, PCA-SIFT filtering and pattern learning to fully exploit the potential of LIPs for NDK identification. We study two tasks: NDK retrieval and NDK detection, in these aspects. In retrieval, the goal is to pull the NDKs of a query to the higher positions of a ranked list so that users can rapidly locate them. In detection, two keyframes are given and a binary decision is made. An interesting issue we study is that the matching patterns formed by NDK pairs can actually be learned for detection. Different from non-NDK pairs, NDK pairs often show LIPs that are smoothly matched over certain spatial arrangements. These matching patterns indeed provide vivid cues for accurate detection. The major contributions of this paper include:

- *Matching*. We propose one-to-one symmetric (OOS) mapping to constrain the nearest neigbour search in LIP set matching. Under this strategy, each LIP can pair with at most one LIP, and furthermore LIPs in a pair are the nearest neighbor of each other. OOS has the merit that only the most reliable pairs are kept for learning so that the ambiguous and false matches can never overcome the true matching patterns of NDK pairs.
- *Filtering*. A multi-dimensional index structure, LIP-IS, is proposed for the rapid filtering of PCA-SIFT descriptors for OOS matching. An efficient collision function is incorporated with LIP-IS to increase the probability of locating a nearest neighbor with high speed. By exploring the data distribution of PCA-SIFT components, the lower and upper bound probabilities of colliding a similar pair of LIPs can be formulated to show the effectiveness of LIP-IS. In addition, the filtering capability of LIP-IS can be asymptotically computed and its probability of missing nearest neighbor can also be estimated.
- *Learning*. The matching of LIPs across NDKs is often highly localized and spatially smooth. We capture this cue in a histogram of matching orientations and then learn the patterns with Support Vector Machines (SVM). This approach is found to be particularly effective for NDK detection where a binary decision is required for every keyframe comparison.

The remaining sections are organized as follows. Section II describes the related works in NDK retrieval and detection. Section III briefly summarizes the properties of DoG LIP detector and PCA-SIFT descriptor. Section IV motivates the use of one-to-one symmetric (OOS) LIP matching strategy in NDK identification. Section V presents our proposed LIP-IS for accelerating OOS matching while Section VI explores the properties of PCA-SIFT descriptors to uphold the validity of LIP-IS in filtering. Section VII proposes a novel idea to learn the matching patterns of NDKs for pattern discrimination. Section VIII presents the experimental findings by comparing the performance of different approaches in NDK retrieval and detection. Finally, Section IX concludes this paper.

## II. Related Work

Duplicate detection has been studied in various media including video, image and music [9], [10], [11]. In image fingerprinting literature [9], [12], the visual content, which is normally referred to as fingerprint, is identified based on the low-level features of an image. The fingerprints are indexed and used for retrieving the suspicious pirated copies of a query image for human inspection. In video copy detection [10], [13], the researches are mostly focused on the rapid identification of similar clips, where fast algorithms are proposed by deriving signature to represent the clip contents. One successful example is the retrieval of commercial clips in large database with high speed [10]. Existing copy detection approaches, in both image and video domains, mostly employ the global or local statistic of features and concatenate them into a vector for retrieval. The use of single feature vector for rapid identification, nevertheless, is generally not tolerant to the perturbation of lighting, color and various transformations. As a consequence, these approaches, while being efficient for detecting the exact duplicate copies, tend to overlook near-duplicate copies which are perturbed by factors such as the photometric and geometric changes.

Near-duplicate detection, particularly in the context of video keyframe, has been addressed recently, partly attributed to the emergence of TRECVID benchmark [14] for multimedia search. Representative works in near-duplicate identification include [3], [15], [16]. In [15], 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 [16], 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 detected SUSAN (Smallest Univalue Segment Assimilating Nucleus) [17] corners as vertices, and the matching of ARGs is constrained by the spatial relation imposed by corner points. To reduce computational load, a distribution-based similarity model is learnt locally and globally in the vertex and graph levels respectively. The learning is feasible since the matching (or transformation) of NDK pairs often follow certain geometric arrangement, otherwise they should not be categorized as "near-duplicate". Although interesting, the approach in [16] suffers from the limitations of slow matching speed and the requirement of heuristic parameters for learning.

In contrast to [16], the approach in [3] utilizes the PCA-SIFT descriptors of LIPs for direct point set matching. To accelerate matching speed, an efficient index structure based on locality sensitive hashing (LSH) is further proposed to facilitate fast search. LSH, nevertheless, requires several user-defined parameters which impact the distortion and granularity of search. In addition, [3] only applies their approach to high-resolution art images, and its robustness remains unclear when the target database is composed of keyframes suffered from low-resolution, motion-blur and compression artifacts. Point set matching usually requires post-processing

such as RANSAC (RANdom SAmple Consensus) [18], Hough transform or homography constraint checking to explicitly eliminate false matches [3], [4], [16]. However, RANSAC is only applicable when correct matches dominate false matches. Hough transform, on the other hand, only favors keyframes with certain regular shapes like lines and circles. In videos, imposing the aforementioned post-processing techniques has certain limitations since objects can be embedded in highly cluttered and complex background, let alone the fact that objects in NDKs may have undergone illumination and viewpoint changes.

In this paper, as in [3], we adopt point set matching, and propose a new index structure called LIP-IS to support fast filtering. LIP-IS is theoretically and practically favorable than LSH [19] in the context of LIP-based NDK identification. It is more capable of locating nearest neighbors in approximate search and requires less parameters than LSH. The proposed NDK detector, as in [16], adopts supervised learning, but it learns the patterns of matching, rather than the statistical parameters of pre-defined distributions. Since the matching of LIPs are supposed to be spatially coherent for NDK pairs, indeed it may not be necessary to explicitly encode the spatial relation with ARG to constrain matching. Instead, our approach performs point set matching with one-to-one symmetric constraint. The outcomes of matching form either spatially regular or random patterns ready for NDK learning and detection. In addition, unlike [3], [16], no post-processing is required.

## III. Feature Extraction with LIP Detector and Descriptor

We employ the DoG (Difference of Gaussian) detector proposed by D. Lowe for LIP detection [4], and the PCA-SIFT descriptors proposed by Ke and Sukthankar to characterize LIPs [8]. The DoG detector and SIFT-based descriptor provide several characteristics that are ideal for NDK identification. First, the detected LIPs are invariant to scale and rotation. Secondly, the descriptors are robust to geometric and photometric transformations such as affine warp, brightness, contrast and color changes. Correct matches can be effectively located even if occlusion and cropping exist. In DoG detector, LIPs are searched over all scales and locations of keyframes. Basically a pyramid of Gaussian images is constructed, and a DoG function is used to locate potential LIPs that are invariant to scale and orientation. A detailed model is then fitted, to the sub-pixel and sub-scale accuracy, to determine the location and scale of a LIP. LIPs that are not distinctive and poorly localized along the edges are rejected at this stage. Finally, one or more orientations are assigned to individual LIP based on the directions of local gradient. The orientations, scale and location of a LIP provide an effective mean of extracting salient descriptor that is invariant to similarity transforms.

The local descriptor characterizes a LIP based on a patch of pixels in its local neighborhood. The patch is centered on a local extreme, scaled to appropriate size, and rotated based on its dominant orientation. SIFT (scale-invariant feature transform), originally proposed in [4], is a 128-dimensional feature vector that captures the spatial structure and local orientation distributions of a patch. Basically, each patch is divided into $4 \times 4$ blocks, and a 8-bin histogram of orientation is computed for each block. The 16 histograms are cascaded one after another to form the descriptor SIFT. The PCA-SIFT, proposed in [8], is a modified version of SIFT that can achieve better accuracy and efficiency. In PCA-SIFT, an eigen space is offline computed to represent the gradient images of local patches. Given a $41 \times 41$ patch extracted according to the scale, location and orientations of a LIP, its gradient image is computed, normalized and then projected to the eigen space. The top few components of the projected vector are then extracted to form a local descriptor. Based on the property of PCA, each component in PCA-SIFT is normally distributed and orthogonal to other components. In our approach, we extract the first 36 components of a projected patch as descriptor. This 36-dimensional PCA-SIFT descriptor has been shown to achieve the best matching performance in [8].

## IV. One-to-one Symmetric LIP Matching

Given LIPs extracted separately from two keyframes, we need to align them, from one keyframe to another, so as to facilitate the similarity measurement. LIP matching is considered as a bipartite graph matching problem. There are numerous algorithms for point set matching. 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 are always suffering from low-resolution, motion-blur and compression artifact. Noise becomes a crucial factor in selecting matching algorithm, particularly when the matching decision is made upon a small local patch. In NDK identification, keyframe transformation introduces noise, and noise itself affects the performance of LIP detection [4]. The localization errors caused by LIP detector deteriorate the distinctiveness of PCA-SIFT. It becomes very common that a LIP fails to find its corresponding LIP in another keyframe, and on the other extreme, a LIP can simply match to many other LIPs 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 instance, although M2M is applicable for many retrieval problems [20], there exists no effective mechanism to restrict false matches. Figure 2 contrasts the M2O and O2O matching of a NDK pair. The M2O matching is implemented with Lowe's nearest neighbor search (NN) [4], where exhaustive search is used to match every LIP to its nearest neighbor in another LIP set under Euclidean distance. The O2O matching is implemented with optimal matching (OM) algorithm [21]. Two LIP sets are optimally matched to maximize the overall score (similarity) of the bipartite graph under the O2O constraint. As shown in this figure, M2O matches (marked by red circle) are mainly caused by noise and local visual ambiguity. They indeed complicate the similarity measurement of point sets. OM, in

contrast, basically removes these matches and retains only the matches that survive the O2O restriction. However, OM is not necessarily a superior choice than NN. The ultimate goal of OM is to optimize matches and it does not guarantee a match pair to possess the nearest neighbor property.

Fig. 2. Matching comparison: many-to-one (left) vs one-to-one (right)

For NDK retrieval, the choice of NN or OM may not be critically important. The relative similarity ranking among keyframes can somewhat alleviate certain degree of faulty matches. However, for NDK detection, a single faulty match can overcome a clever decision. In our approach, to allow effective learning of matching patterns, false matches should be kept to a minimum level. To retain only the most reliable matches for learning, we introduce a new scheme – namely one-to-one symmetric matching (OOS). Similar to OM, OOS belongs to O2O, but as NN, OOS ensures all the matches are the nearest neighbors. The symmetric property is also emphasized so that if LIP $P$ 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 LIP set $\mathsf{A}$ to set $\mathsf{B}$ is exactly the same as $\mathsf{B}$ to $\mathsf{A}$, unless there are LIPs that have more than one nearest neighbor. The O2O and symmetric constraints indeed come from the intuition that LIPs should reliably match each other if they do belong to a duplicate version.

OOS meets the definition of near-duplicate keyframe nicely. A NDK pair, by nature, should be symmetric of each other in certain regions of keyframes. By imposing symmetric constraint, we indeed enforce the stability of LIP matching. The stability can lead to more efficient and reliable NDK detection if the transitivity closure of NDK pairs holds. In other words, if keyframes $k_1$ and $k_2$ are NDK pair, and similarly $k_2$ and $k_3$ are NDK pair, we can draw the conclusion that $k_1$ and $k_3$ are NDK pair as well without any effort. However, to propagate this transitivity, the matches between ($k_1$,$k_2$) and ($k_2$,$k_3$) should be symmetric and stable. Generally speaking, O2O matching cannot guarantee each matched LIP pair to be meaningful. Some false matches indeed could exist with high similarity value. But it becomes a rare case for these false matches to be symmetrically stable and paired to each other in both directions. Figure 3 shows the matching results of a NDK pair with OOS and OM. Basically OOS successfully excludes most of the false matches (red lines) found in OM.

Besides matching constraint, another concern is whether to permit full or partial matching. In full matching, all points in two sets are matched. In partial matching, only a subset of points are matched to exclude point pairs with low similarity

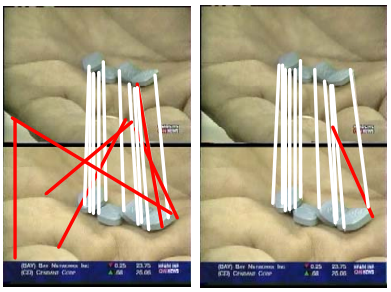


Fig. 3. Matching comparison between one-to-one (left) and one-to-one symmetric (right) strategy (white line indicates correct match, and red line shows false match).

measure. In NDKs, due to editing operations, very often only sub regions are near-duplicate. LIPs outside these regions can introduce meaningless matches. As a consequence, similar to [3], [4], we adopt partial matching with the aid of thresholding as follows

$$Sim(Q, P) \geq \alpha \tag{1}$$

The threshold $\alpha$ specifies the minimum allowable similarity between two LIPs for a potential match. In Eqn (1), the similarity of two LIPs $Q$ and $P$ described by PCA-SIFT, denoted as $Q = [q_1, q_2, \ldots, q_{36}]$ and $P = [p_1, p_2, \ldots, p_{36}]$, is defined as

$$\mathcal{S}im(Q, P) = \sum_{i=1}^{36} q_i \times p_i \tag{2}$$

where $p_i$ and $q_i$ are normalized such that they are in the range of $[-1, 1]$ and $|P| = |Q| = 1$[1]. Eqn (2) specifies the cosine angle of $Q$ and $P$ in the high dimensional feature space. Let $\mathsf{K}$ as the set of LIP pairs that satisfy Eqn (1) and being matched by OOS in two keyframes. Subsequently, the similarity between keyframes can be determined directly based on the cardinality of $\mathsf{K}$, or by taking into account each LIP pair as follows

$$\mathcal{K}Sim(\mathsf{K}) = \frac{\sum_{(Q,P)\in\mathsf{K}} \mathcal{S}im(Q, P)}{|\mathsf{K}|} \tag{3}$$

where $|\mathsf{K}|$ denotes the cardinality of $\mathsf{K}$, and $\mathcal{K}Sim$ computes the average similarity of LIP pairs being matched in two keyframes.

## V. Filtering with Approximate Nearest Neighbor Search

In OOS matching, given a query LIP $Q$ and a keyframe $\mathbf{I}$ with a collection of LIPs $\mathsf{C}$, the fundamental task is to find the nearest neighbor $\hat{P} \in \mathsf{C}$ of $Q$. This task essentially involves a series of comparisons which can be computationally expensive. The aim of filtering is to omit some comparisons by rapidly locating the set of candidates in $\mathsf{C}$ that has potential to become the nearest neighbor of $Q$. In the current literature, there are various multi-dimensional index structures (e.g., locality sensitive hashing [19]) that allow the approximation

[1]Note that it is not necessarily to set the norms of $P$ and $Q$ to 1. We set $|P| = |Q| = 1$ to relate matching and filtering, and simplify Eqn (7) and Eqn (13).

of neighbors. In this section, we begin by proposing a new index structure to accommodate PCA-SIFT (Section V-A).

For filtering purpose, we define a similarity, $\mathcal{S}im^{'}$, to approximate Eqn (2). Let $P$ and $Q$ as the PCA-SIFT descriptors of two LIPs, the similarity measurement between $P$ and $Q$ is estimated as

$$\mathcal{S}im^{'}(Q,P) = \sum_{i=1}^{36} \mathcal{C}(q_i, p_i) \tag{4}$$

where the degree of closeness between two $1D$ points is elaborated by a *collision* function $0 < \mathcal{C} \leq 1$ (Section V-B). An interesting note is that the upper and lower bound probabilities of colliding a similarity pair of $q_i$ and $p_i$ can be estimated (Section V-C). Then a decision is made to gate whether $Q$ and $P$ are potentially the nearest neighbor, defined as

$$\mathcal{N}(Q,P) = \begin{cases} \text{Yes} & \text{if } \mathcal{S}im^{'}(Q,P) \geq 36 - M \\ \text{No} & \text{Otherwise} \end{cases} \tag{5}$$

The gating parameter $M$ is a margin that can be rigorously computed (Section V-D). In brief, the problem of filtering is to rapidly find a subset of $\mathsf{C}$, called $\mathsf{A}$, where

$$\mathsf{A}(Q) = \{P | \mathcal{N}(Q,P) = \text{Yes}\} \tag{6}$$

### A. Indexing and Query Processing

To allow fast filtering, we introduce a new index structure, namely LIP-IS, specifically for LIP indexing with PCA-SIFT descriptor. LIP-IS is indeed a hash structure with a group of 36 histograms formed independently by every component of PCA-SIFT. Given a keyframe with LIP set $\mathsf{C}$, a LIP-IS is constructed by equally and independently quantizing each dimension into 8 bins, with a resolution of $\Delta = 0.25$ (the range of a component is $[-1, 1]$). Given $P = [p_1, p_2, \ldots, p_{36}]$, the index of $p_i$ is hashed to,

$$\mathcal{H}(p_i) = \lfloor \frac{p_i + 1}{\Delta} \rfloor \tag{7}$$

In total, a LIP-IS is composed of $8 \times 36$ bins, as illustrated in Figure 4. Given a point $P$, we repeatedly index it into the corresponding bins of 36 histograms, according to its quantized value in a particular dimension. Thus, each LIP is hashed for 36 times in LIP-IS. Given a query point $Q$, similarly we hash it repeatedly for 36 times, and retrieve all points that satisfy Eqn (5). Based on the collision function which will be presented in Eqn (8), when $q_i$ is hashed to bin $j$ of $i^{th}$ dimension, all points in bins $j-1$, $j$ and $j+1$ will be considered, as shown in Figure 4. Finally, those points that meet Eqn (5) form the set $\mathsf{A}(Q)$ in Eqn (6) for LIP matching. The overall algorithm is summarized in Algorithm 1.

### B. Designing Collision Function

Because the axes of PCA-SIFT descriptors are orthogonal to each other and the data is modeled as Gaussian distributed along each direction, there are several ways to design the index structure and its corresponding collision function. One possible structure is with adaptive binning in each dimension, by exploring the data distribution of PCA-SIFT in every dimension. In other words, the size of bins varies within and across the dimensions. Such structure, however, comes with the price of time which can slow down both the indexing and querying speed.

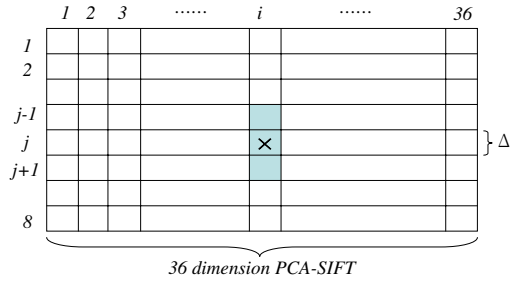



Fig. 4. LIP-IS for approximate search. Given a query point falls in location "×", all LIPs inside gray bins will be retrieved.

**Algorithm 1** Local interest point matching with LIP-IS filtering mechanism.

- INPUT: Keyframe $\mathbf{I}_1$ with LIP set $\mathsf{C}_1$, and keyframe $\mathbf{I}_2$ with $\mathsf{C}_2$
- OUTPUT: One-to-one symmetric LIP pairs
  1) Hash all points in $\mathsf{C}_2$ to LIP-IS;
  2) For each point $Q$ in $\mathsf{C}_1$,
     a) Hash $Q$ to LIP-IS;
     b) Retrieve the set $\mathsf{A}(Q)$ satisfying Eqn (5);
     c) If $\mathsf{A}(Q) \neq \emptyset$, find the nearest neighbor $P \in \mathsf{A}(Q)$ with one-to-one symmetric constraint;
  3) Return the set of matched pairs, i.e., $\{(Q,P) | Q \in \mathsf{C}_1, P \in \mathsf{C}_2\}$.

LIP-IS adopts uniform quantization. A possible way to design its collision function $\mathcal{C}$ is to measure the Gaussian distance between two LIPs when they collide. One disadvantage is that this function is slow in evaluation. Furthermore, the Gaussian parameters at each dimension need to be precomputed and updated. For speed efficiency, we measure the distance between two LIPs $Q$ and $P$ at $i$ dimension with uniform distribution:

$$\mathcal{C}(q_i, p_i) = \begin{cases} 1 & \text{if } |\mathcal{H}(q_i) - \mathcal{H}(p_i)| \leq 1 \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

The hashing function $\mathcal{H}$ is defined in Eqn (7). Combining Eqn (8) with Eqs (4)-(6), $P \in \mathsf{A}(Q)$ if they collide in $36 - M$ dimensions. This algorithm has the merit that it is efficient and easy to implement with simple bit operation. We compare Eqn (8) with a Gaussian-based collision function, and find that their performance is quite close but the former is significantly faster in filtering.

### C. Probability of Collision

Since LIP-IS is uniformly quantized and its collision function assumes uniform distribution, we estimate the probability of colliding two LIPs $P$ and $Q$, when their underlying distribution is Gaussian. There are two extreme conditions for $P$ and $Q$ to and not to collide at dimension $i$. In the first case, the

distance between them is $2\Delta$ and they collide. One example is when $P$ and $Q$ fall into $k$ and $k-1$ bins respectively. The value of $P$ is just slightly less than $(k+1)\times\Delta$ and the value of $Q$ is exactly $(k-1)\times\Delta$. Thus $P$ and $Q$ will collide at $i$ dimension. In the second case, the distance between $P$ and $Q$ is about $\Delta$ but they do not collide. One example is when $P$ and $Q$ falls into $k$ and $k-2$ bins respectively. The value of $P$ is $k\times\Delta$ and the value of $Q$ is just less than $(k-1)\times\Delta$. In other words, $P$ and $Q$ slightly miss for collision. For ease of analysis, we term the first case as "best case", and the latter as "worst case" of collision.

Because PCA-SIFT descriptors are Gaussian distributed with mean $\mu_i$ and $\sigma_i$ in $i$ dimension, its distance distribution between $q_i$ and $p_i$, i.e., $z_i = q_i - p_i$, can be characterized by a probabilistic density function as,

$$\mathcal{P}(z_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\{-\frac{z_i^2}{4\sigma_i^2}\} \quad \text{and} \quad z_i \geq 0 \tag{9}$$

In the best case, the probability that $Q$ collides with $P$ at $i$ dimension is expressed as,

$$\mathbf{P}_b = 2\int_0^{2\Delta}\frac{1}{\sqrt{2\pi}\sigma_i}\exp\{-\frac{z_i^2}{4\sigma_i^2}\}\,dz_i \tag{10}$$

In the worst case, two LIPs collide with the following probability,

$$\mathbf{P}_w = 2\int_0^{\Delta}\frac{1}{\sqrt{2\pi}\sigma_i}\exp\{-\frac{z_i^2}{4\sigma_i^2}\}\,dz_i \tag{11}$$

Basically two LIPs which are similar are likely to collide at $i$ dimension. Eqn (11) sets the lower bound probability for this type of collision. On the other hand, two LIPs with lower similarity are less likely to collide at $i$ dimension, and Eqn (10) sets the upper bound of the probability for collision.

### D. Computing Gating Parameter $M$

*Theorem 1:* The LIPs satisfying Eqn (1) will collide each other at least in $36-M$ out of 36 dimensions if

$$M \leq \frac{2(1-\alpha)}{\Delta^2} \tag{12}$$

*Proof:* Given two LIPs $Q$ and $P$ which are vectors in high dimensional space, we can relate their Euclidean distance and the cosine similarity in Eqn (2) by exploiting the law of cosines as below

$$|Q-P|^2 = |Q|^2 + |P|^2 - 2\times|Q|\times|P|\times\mathcal{S}im(Q,P) \tag{13}$$

Since $|P| = |Q| = 1$, we have

$$\mathcal{S}im(Q,P) = 1 - \frac{\mathcal{D}ist^2(Q,P)}{2} \tag{14}$$

where

$$\mathcal{D}ist^2(Q,P) = |Q-P|^2 = \sum_{i=1}^{36}(q_i-p_i)^2 \tag{15}$$

Suppose there are $M$ out of 36 dimensions with $(q_i-p_i)^2 > 2\Delta$. Then, given the index structure, the minimum distance, $\mathcal{D}ist(Q,P)$, between $Q$ and $P$ is

$$\begin{aligned}\mathcal{D}ist^2(Q,P) &= \sum_{i=1}^{36}(q_i-p_i)^2 \\ &\geq \underbrace{0^2+0^2+\ldots+0^2}_{36-M} + \underbrace{\Delta^2+\Delta^2+\ldots+\Delta^2}_{M}\end{aligned} \tag{16}$$

Comparing Eqn (16) and Eqn (14) with the inequality, $Sim(P,Q) \geq \alpha$, in Eqn (1), we can easily derive

$$1 - \frac{M\Delta^2}{2} \geq \alpha \tag{17}$$

which is exactly Eqn (12) after simple manipulation, and this ends the proof.

Theorem (1) specifies how to compute $M$ in Eqn (5), given the parameter $\alpha$. Meanwhile, it guarantees that all LIP candidates being filtered are with the similarity smaller than $\alpha$. Note that this does not mean the candidate set $\mathsf{A}$ will keep all LIPs with similarity values greater than $\alpha$. However, it does assure in all cases that the nearest neighbor $\hat{P}$ of $Q$ must be inside $\mathsf{A}$, except when $\mathcal{S}im(Q,\hat{P}) < \alpha$. Theorem (1) also specifies how to select $\Delta$, if the parameters $\alpha$ and $M$ are given.

## VI. Filtering Analysis and Comparison

LIP-IS has the property that it is specifically constructed to facilitate the similarity measurement in Eqn (4). Its relationship with the actual LIP similarity (Eqn (2)) is expressed in Theorem (1), which relates the parameters $M$, $\alpha$ and $\Delta$. In this section, we further explore the complexity of LIP-IS in supporting the query processing and in locating the nearest neigbour.

### A. Filtering Capability

The size of candidate LIPs, i.e., $|\mathsf{A}|$, determines the filtering effectiveness. The smaller this value, the better the filtering capability. For simplicity, we assume the features in all dimensions of PCA-SIFT have the same mean $\mu$ and variance $\sigma$ under Gaussian distribution. Then according to Eqn(10), the upper bound probability of colliding LIPs $P$ and $Q$ at a dimension is $\mathbf{P}_b$. Assuming[2] the query LIP is independent of the LIPs in $\mathsf{C}$. The upper bound probability of a LIP being retrieved, after looking at 36 dimensions, can be described as follows

$$\mathbf{P}_f = \mathbf{P}_b^{36} \tag{18}$$

Notice that $\mathbf{P}_f \ll 1$ in general. Let $n = |\mathsf{C}|$ as the set cardinality, the expected number of interest points in $|\mathsf{A}|$ is

$$|\mathsf{A}| = O(\mathbf{P}_f \times n) \quad \text{and} \quad \mathbf{P}_f \ll 1 \tag{19}$$

[2]This assumption is in general true when comparing non-NDK pair (NNDK), not true, however, for NDK pair (NDK). Since in real world the number of non-NDK pairs is much greater than the NDK pairs, i.e., $|\text{NNDK}| \gg |\text{NDK}|$, this assumption actually holds for most cases.

### B. Index Construction and Query Speed

LIP-IS construction is straightforward since each interest point is hashed independently for $d$ times, where $d$ is the dimension of PCA-SIFT. Denote $n = |\mathsf{C}|$, the construction takes $d \times n$ operations. Given a query interest point $Q$, similarly, we hash $Q$ for $d$ times in order to locate the set $\mathsf{A}$. The checking of whether a LIP in $\mathsf{C}$ collides with $Q$ in a dimension can be efficiently carried out with simple bit operation. Since, in theory, $n \times \mathbf{P}_f$ LIPs will be retrieved for matching under OOS constraint, the query speed is bounded by $O(d + n \times \mathbf{P}_f)$.

### C. Probability of Missing Nearest Neighbor

According to Theorem (1), the nearest neighbor of $Q$ with similarity larger than $\alpha$ will collide with $Q$ in $36 - M$ dimensions. The nearest LIP $\hat{P}$, however, may not be found inside set $\mathsf{A}$ if $\mathcal{S}im(Q, \hat{P}) < \alpha$ and $\hat{P}$ does not collide with $Q$ in the remaining $M$ dimensions. To simplify the estimation of losing $\hat{P}$, we assume the probability of collision of $\hat{P}$ and $Q$ in each dimension is equal but independent, and its lower bound probability is $\mathbf{P}_w$ as specified in Eqn (11). The upper bound probability of missing $\hat{P}$, by imposing the restriction that $\hat{P}$ should collide in the remaining $M$ dimensions, can be modeled with Bernoulli process as follows

$$\mathbf{P}_{miss} = \sum_{i=1}^{\hat{M}} \left( \begin{array}{c} \hat{M} \\ i \end{array} \right) \mathbf{P}_w^{\hat{M}-i} (1 - \mathbf{P}_w)^i = 1 - \mathbf{P}_w^{\hat{M}} \qquad (20)$$

where $\hat{M} = \frac{2(1-\alpha)}{\Delta^2}$ with reference to Theorem (1). Practically $\mathbf{P}_{miss}$ is very small for two reasons. Firstly, $\hat{P}$ normally collides with $Q$ with a much higher probability than $\mathbf{P}_w$. Secondly the value of $\hat{M}$ is usually much smaller than 36 to allow fast filtering.

### D. Comparison with LSH

Locality Sensitive Hashing (LSH) has been applied for LIP filtering in [3]. Compared with LSH, LIP-IS has the advantage that it maximizes the possibility of collision when two LIPs are similar to each other. Table I shows the comparison of LIP-IS, LSH and the naive approach (without index structure). In LSH [19], [22], four parameters ($l$, $s$, $c$, $\varepsilon$) are required. The parameter $l$ specifies the number of times a LIP set to be tessellated. Each tessellated LIP set will be further partitioned, in random, by $s$ times. Both $l$ and $s$ impact the speed of constructing LSH. Practically, the parameter $s$ should be larger than $d$ to guarantee efficient filtering. With a larger $l$, the probability of locating nearest neighbor becomes higher, however, with the expense of huge memory space consumption [3]. In addition, a higher value of $l$ means the increase in size of candidates set $\mathsf{A}$. Practically a constant $c$ is used to restrict the number of returning candidates. In general, LSH uses $c$ and $l$ to adapt filtering capability. It is difficult to optimize the parameters in LSH to seek a balance between speed and accuracy. In contrast, LIP-IS rigorously estimate its filtering capability with the probability $\mathbf{P}_f$ tailored specifically to the distance distribution of PCA-SIFT descriptor.

TABLE I
COMPARISON WITH LOCALITY SENSITIVE HASHING (LSH)

| | LIP-IS | LSH | Naive |
|---|---|---|---|
| Offline index | $d \times n$ | $l \times s \times n$ | - |
| Filtering capability | $O(\mathbf{P}_f \times n)$ | $n - c \times l$ | - |
| Online query time | $O(d + \mathbf{P}_f \times n)$ | $O(d \times n^{1/1+\varepsilon})$ | $O(n)$ |
| Prob. of missing NN | $\mathbf{P}_{miss}$ (Eqn (20)) | ? | 0 |

$d$: dimension; $n$: number of LIPs; $l$, $s$, $c$ and $\varepsilon$: parameters in LSH; ?: not directly estimated

In LSH, the query speed is affected by a user-defined parameter $\varepsilon$ which specifies the threshold of error tolerance. In theory, LIP-IS can be about the same speed as LSH in term of query time since $\mathbf{P}_f \ll 1$ while $\varepsilon$ cannot be too large. Similar to query speed, the probability of missing nearest neighbor is not directly estimated in LSH, but depends on the input of $\varepsilon$. In LIP-IS, this probability is estimated with $\mathbf{P}_{miss}$ which depends on the setting of gating parameter $M$. The structure of LIP-IS indeed maximizes the probability that LIPs similar to each other collide in their neighborhood. Furthermore, because LIP-IS capitalizes on the distribution of points in each dimension, the probability of missing nearest neighbors is generally low. This property makes LIP-IS more adaptive to different datasets, without the need of simultaneously optimizing several parameters to tradeoff different factors as in LSH.

## VII. LEARNING MATCHING PATTERN FOR NDK DETECTION

NDKs often share common objects in some sub regions of keyframes. If LIPs are correctly detected and matched, the matching lines formed by the matched LIPs should be regularly, nearly parallel, in these sub regions (see figures 11(b)-(c) for illustration). In principle, the matching of LIPs for NDK pairs should be spatially smooth and highly localized. In other words, the groups of matched LIPs should be close in space, and overall exhibit unique matching patterns that follow certain spatial arrangements. For instance, NDK pairs are often linked by a bunch of parallel or zoom-like lines formed by matched LIPs. Non-NDK pairs, in contrast, may simply show random patterns with matching lines being arbitrarily crossed in space (see figures 11(e)-(f)). In this section, we propose to capture the matching patterns with the histogram of matching orientation, and then learn the patterns with Support Vector Machines (SVM) for discriminative classification.

The histogram of matching orientations can be easily constructed by aligning two keyframes vertically, and quantizing the angles formed by the matching lines and the horizontal axis. With reference to Figure 5, two keyframes are arranged vertically, and the matched LIPs are linked with straight lines. The angles $\theta$ between the matching lines and the horizontal axis, in the range of $0^\circ$ to $180^\circ$, are computed. Denote $h$ as the height of the upper keyframe (Keyframe-1 in Figure 5), and the coordinates of LIP $A$ in Keyframe-1 and LIP $A^{'}$ in Keyframe-2 as $(x_0, y_0)$ and $(x_1, y_1)$ respectively. The angle $\theta$ is computed as follows

$$\theta = \arccos\left(\frac{x_1 - x_0}{\sqrt{(x_1 - x_0)^2 + (y_1 + h - y_0)^2}}\right) \qquad (21)$$

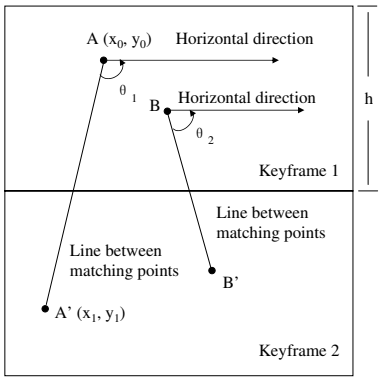



Fig. 5. Computing orientation of matching lines

TABLE II
EXPERIMENTED APPROACHES WITH DIFFERENT MATCHING STRATEGIES AND FILTERING SCHEMES.

| | LIP-IS | LSH | CH |
|---|---|---|---|
| One-to-One Symmetric (OOS) | √ | √ | √ |
| Nearest neighbor (NN) | √ | √ | √ |
| Optimal matching (OM) | | | √ |

The histogram is constructed by counting the number of LIP pairs at a particular range of $\theta$. We quantize the histogram into 36 bins with a step of $5^\circ$ from $0^\circ$ to $180^\circ$. An advantage of this histogram is that the sizes of two compared keyframes are not required to be the same.

The histograms of NDK pairs and non-NDK pairs are input to SVM as positive and negative examples respectively for learning and detection. SVM is based on the idea of structural risk minimization. It achieves higher generalization performance than other classifiers. We use radial basis function (RBF) to map the training histograms into higher dimensional feature space for pattern discrimination. With SVM, the detection of NDK pairs becomes straightforward in our approach. Given two keyframes, LIP set matching is performed with the support of LIP-IS. The histogram of matching orientation is directly computed based on the upshot of matching. Subsequently, the histogram is fed into SVM to perform binary classification to output the decision of either yes or no detection.

## VIII. EXPERIMENTS

We conduct experiments to assess the performance of NDK retrieval and detection. We use the data set provided by [16] for experiments. This data set consists of 600 keyframes with 150 NDK pairs and 300 non-NDKs extracted from TRECVID 2003 corpus [14]. All the keyframes are with the same resolution of $352 \times 264$ pixels. By Lowe's DoG LIP detector, the number of local interest points for each keyframe ranges between 10 to 2553. On average, there are 1200 LIPs per keyframe.

To assess the performance, we compare different approaches, varying in terms of features, matching and filtering strategies, as shown in Table II. We experiment three matching strategies: nearest neighbor (NN), one-to-one optimal (OM) and one-to-one symmetric (OOS) matching. NN is originally proposed by Lowe for matching LIPs [4]. This strategy allows many-to-one matching. For OM, there exist several algorithms including the approximation versions varying in term of computational efficiency and accuracy [23], [21]. We use the classical MWBG (maximum weighted bipartite graph) matching algorithm for implementing OM. OOS, originally proposed in this paper, is straightforward to implement and enforces both the one-to-one and symmetric matching of nearest neighbors.

For filtering, we test three strategies: the proposed LIP-IS, locality sensitive hashing (LSH) and color histogram (CH). For LIP-IS, we set $\alpha = 0.87$ and $M = 0$ to reduce the candidate size as many as possible. For LSH, we manually optimize the parameters and finally set $l = 2$ and $s = 108$ to tradeoff both speed and accuracy. Under these settings, both LIP-IS and LSH cannot guarantee the retrieval of nearest neighbors. However, the probability of missing a nearest neighbor is much less in LIP-IS than LSH. We will demonstrate this by showing that LIP-IS gives significantly better results in both NDK retrieval and detection when incorporating with OOS and NN. In CH, each keyframe is represented by a $3D$ color histogram of 162 bins in HSV color space. Hue, saturation and brightness are quantized to 18, 3, 3 bins respectively. We experiment both OOS and NN matching with LIP-IS, LSH and CH supports. We only test OM with CH support. LIP-IS and LSH are not considered since OM involves optimization and both filtering schemes cannot support this procedure. Note that, however, OM without any filtering support is computationally intractable. Each keyframe comparison takes averagely 44 seconds to complete. For distance measure, all the approaches use Cosine similarity measure (Eqn (2)). Three exceptions are CH which uses histogram intersection, NN and NN with CH support which use Euclidean distance as proposed in [4].

### A. NDK Retrieval: Performance Comparison

We use all NDK pairs (300 keyframes) as queries for performance evaluation. For each query, basically 599 keyframe comparisons are involved and then a ranked list of keyframes is produced according to their underlying similarity. We adopt the measure proposed in [16] to assess the retrieval performance by estimating the chance of hitting a correct NDK in the top-$k$ position of a ranked list. Formally, the probability of the successful top-$k$ retrieval is defined as

$$P(k) = \frac{Q_c}{Q} \tag{22}$$

where $Q_c$ is the number of queries that rank their NDKs within the top-$k$ position, and $Q$ is the total number of queries.

Figures 6-8 show the performance comparison of the proposed approaches (OOS, LIP-IS+OOS) with other methods. We use CH as the baseline to judge the improvement of LIP matching and filtering. CH returns the top-40 similar keyframes for further ranking by other matching strategies. Basically, the approaches using CH for filtering make use of both the color and PCA-SIFT features for retrieval. During NDK ranking, the keyframe similarity is based on the cardinality of matched pairs. Intuitively, two keyframes with larger number of matched point pairs reserve the higher

chance of claiming their NDK identity. The keyframes having the same matching cardinality are further ranked according to the average similarity of matched pairs, as indicated in Eqn (3). Note that the keyframes with more number of LIPs do not imply high matching cardinality, since we adopt partial matching where a matched pair needs to satisfy Eqn (1).

Figure 6 shows the comparison of two matching strategies, OOS and NN, with CH as the baseline. OOS shows constantly better performance than NN across all $k = [1, 30]$. This is mainly because OOS sustains only the most reliable matches for ranking through one-to-one and symmetric constraints. OOS successfully removes some faulty matches found in NN. Overall, both matching algorithms show significantly superior performance than CH, indicating the advantages of LIP-based representation over color histogram which is sensitive to lighting and viewpoint variations. Figure 7 shows the performance of various filtering schemes when incorporating with OOS. LIP-IS yields the best performance since the chance of including nearest neighbors for OOS matching is higher compared with LSH. From our observation, LSH very often fails in returning the nearest neighbors and, as a result, the OOS matching pairs sometime appear not meaningful. Comparing the OOS with and without LIP-IS, the performance of top-$k$ retrieval is close to each other. LIP+OOS is slightly better by its ability in pulling NDK to higher rank position. This is because we set the margin $M = 0$, in other words, we require all the 36 features of PCA-SIFT to collide in LIP-IS. This simple scheme actually removes few more false matches and makes OOS more stable.

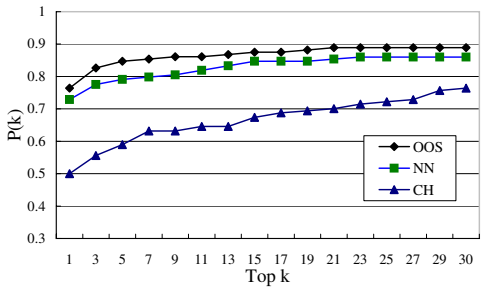



Fig. 6. Performance comparison of different matching algorithms

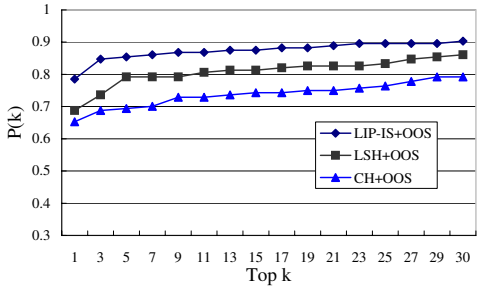



Fig. 7. Performance comparison of various filtering schemes with OOS as matching algorithm

Figure 8 compares our proposed approach (LIP-IS+OOS) with four other methods. Our approach achieves constantly better results than others across all $k$ being tested. The results of NN with different filtering schemes are indeed quite different. The combination of NN with LSH is basically poorer than with color histogram. From our observation, since LSH cannot always include the nearest neighbors, the matching results of NN become somewhat random and cause ineffectiveness in NDK ranking. When NN is integrated with LIP-IS, however, the performance is significantly better since more nearest neighbors are returned. This finding empirically proves the advantage of LIP-IS. The combination of CH+OM shows comparable performance to LIP-IS+NN for some $k$. It outperforms CH+NN in most top-$k$ retrieval. This also indicates the benefit of imposing one-to-one constraint in retrieving. Overall, our proposed approach shows the best performance and is capable of maintaining a success rate of over $85\%$ for $k \geq 3$. When $k = 1$, almost $80\%$ of NDK pairs are retrieved. Compared to the recent results in [16] where $P(k) \approx 0.6$ when $k = 1$ and $P(k) \approx 0.75$ when $k = 30$, our approach demonstrates considerably better results although no training is engaged.

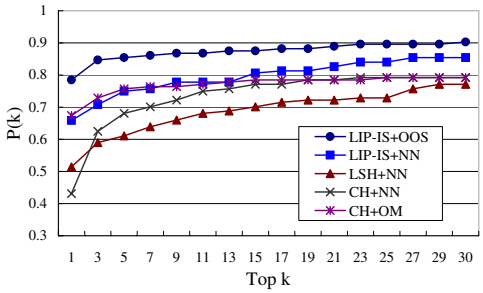



Fig. 8. Performance comparison of the proposed approach (LIP-IS+OOS) with four other approaches

### *B. NDK Detection: Precision, Recall and Scalability Testing*

NDK detection is indeed a harder problem than NDK retrieval. First, detection involves hard decision, i.e., true or false for each comparison in our case. Secondly, the number of NDK pairs increases exponentially with the number of keyframes in videos, resulting in a critical demand of speed and accuracy for each decision. Considering 600 keyframes with 150 NDK pairs, we have 179700 of candidate pairs for detection. The chance of success for a random selection of NDK pair is only 0.0008. Due to heavy computational load, we conduct experiments on 150 NDK pairs and 18000 non-NDKs pairs randomly drawn from the 600 keyframes. The chance of success for a random selection is 0.008. We use precision and recall defined as follows for performance evaluation,

$$\text{Precision} = \frac{\text{\# of correctly detected NDK pairs}}{\text{Total number of detected NDK pairs}}$$

$$\text{Recall} = \frac{\text{\# of correctly detected NDK pairs}}{\text{Total number of NDK pairs}}$$

$$\text{Accuracy} = \frac{\text{\# of correctly detected NDK and non-NDK pairs}}{\text{Total number of keyframe pairs}}$$

We randomly choose 20 NDK pairs (positive examples) and 40 non-NDK pairs (negative examples) for SVM training. We train two SVM classifiers, one with OOS and the other with NN matching strategy. The training set is selected from TREC 2003 corpus, and is independent of the testing set used in the experiments. More negative examples are used since the number of non-NDK pairs is much larger than NDK pairs in real

TABLE III
EXPERIMENTAL RESULTS OF NDK DETECTION

| Method | LIP-IS + OOS | LIP-IS + NN | OOS | NN | LSH +OOS | LSH + NN |
|---|---|---|---|---|---|---|
| Recall | 0.8133 | 0.80 | 0.82 | 0.78 | 0.8125 | 0.7847 |
| Precision | 0.6667 | 0.481 | 0.7885 | 0.4835 | 0.2532 | 0.2603 |
| Accuracy | 0.9543 | 0.9185 | 0.9692 | 0.9190 | 0.8086 | 0.8189 |

cases. From our empirical finding, allowing more non-NDK pairs for training will generally lead to better performance. This is not surprising since the matching patterns of non-NDK pairs varies a lot compared with NDK pairs, ranging from just few random matches to large amount of scrambled and false matches. In practice, more negative examples are required for effective discrimination. During training, we also attempt different kernels including polynomial, sigmoid and RBF. The following experiments are based on RBF kernel which gives overall the best performance.

Table III shows the performance of six different approaches for NDK detection, on a dataset with 150 NDK pairs and 1800 non-NDK pairs. Basically the approaches with OOS matching perform better. The matches found by OOS are robust for learning and detection. In contrast, the matches by NN are less reliable and the resulting patterns which include many-to-one matches are found to be less discriminative for learning. LIP-IS, again, is shown to be a better filtering scheme than LSH. A necessary condition for learning and detection is that the matches need to be less noisy, and the chance of satisfying this condition is higher when the filtering scheme can retain as many nearest neighbors as possible. However, LSH fails in this aspect and gives lower precision. The results of LIP-IS, although encouraging, is still not as good as with OOS alone. This is because we set the margin $M = 0$ for speed reason and, as a result, LIP-IS+OOS basically retains less matches than pure OOS. In general, SVM needs more and reliable matches for effective discrimination and thus OOS exhibits the best performance.

To further understand the performance of detection on different types of NDKs, we manually group the 150 NDK pairs into three types, based on their factors of near-duplicate. The considered types are: (i) scenes and objects 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 types (i) and (ii). In type (ii), there are 25 pairs of NDKs which are exactly identical. Table IV summarizes the performance of OOS and LIP-IS in detecting the three types of NDK pairs. OOS is able to detect approximately 75% of type (i) NDKs, 95% of type (ii) NDKs, and 70% of type (iii) NDKs. Type (ii) NDKs are easier to be detected as the matching patterns of LIPs are simpler to learn (see figures 11(b)-(c)). In contrast, type (iii) NDKs, which represents the diverse set of NDKs populated by types (i) and (ii), are relatively difficult to detect. Type (i) NDKs present certain aspects of challenge due to viewpoint and lighting variations. When manual editing, in particular at the region-level, is further added resulting in type (iii) NDKs, the number of matched LIP pairs could become few for effective NDK identification.

TABLE IV
TYPES OF CORRECTLY DETECTED NDK PAIRS.

| Type | Pairs | OOS | LIP-IS+OOS |
|---|---|---|---|
| (i) scene | 43 | 32 | 33 |
| (ii) edit | 65 | 61 | 62 |
| (iii) scene + edit | 42 | 30 | 27 |

To test the scalability of detection, we conduct experiments by incrementally and randomly adding 1800 non-NDK pairs to the 150 NDK pairs. Figures 9 and 10 show the precision and accuracy of detection respectively when the number of non-NDK pairs is increased, but NDK pairs remain the same. In general, the precision drops as non-NDK pairs increase since the chance of generating false alarms is higher. As indicated in Figure 9, although the precision drops, OOS and LIP-IS+OOS still perform satisfactory. Their precision at 18000 non-NDK pairs is higher or about the same compared to other approaches at 9000 non-NDK pairs. The accuracy of detecting NDK and non-NDK pairs, as shown in Figure 10, is indeed excellent for OOS and LIP-IS+OOS. OOS maintains a detection rate at the level of 0.975, while LIP-IS+OOS achieves a rate at 0.95 level.

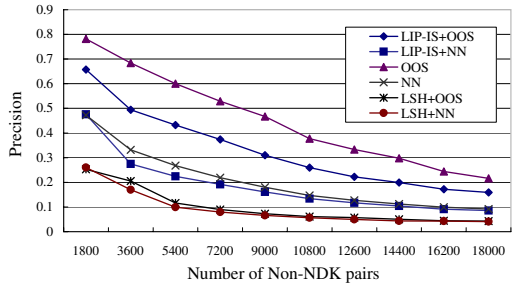



Fig. 9. Precision of NDK detection by randomly adding 1800 non-NDK pairs each time

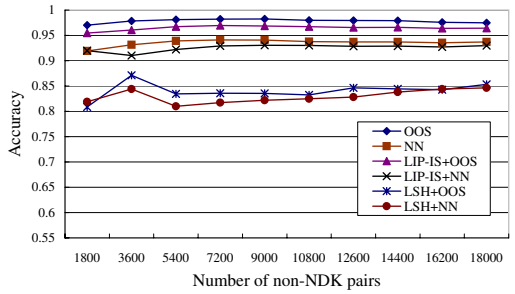
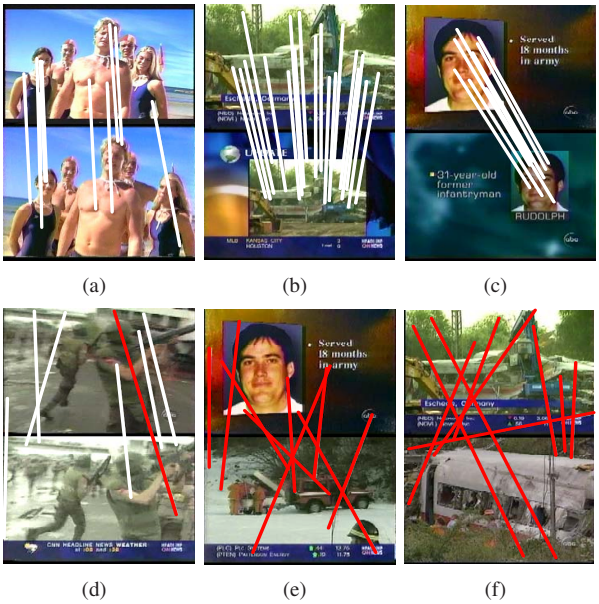



Fig. 10. Accuracy of detection by randomly adding 1800 non-NDK pairs each time

Figure 11 shows NDK and non-NDK pairs detected by OOS. The figures 11(a)-(c) depict three typical matching patterns of NDK pairs which accommodate changes due to viewpoint, scale, translation, color, illumination and editing operations. Figure 11(d) shows two NDKs which are acquired respectively at slightly different time slices and suffered from motion-blur and illumination change. Overall, our approach can successfully deal with these four common cases. Figures 11(e)-(f) show two non-NDK pairs and their matching patterns. The majority of matches are randomly paired across




(a) (b) (c)

(d) (e) (f)

Fig. 11. Examples of NDK and non-NDK pairs. (a)-(d): correct positives, (e)-(f): correct negatives


(a) (b) (c)

Fig. 12. Examples of NDK pairs not being detected

space without following any logical arrangement. Figure 12 also shows several difficult NDK pairs yet to be detected. These pairs are undergone significant color and lighting changes. Only few matches are found and our SVM classifier normally rejects these types of NDK pairs.

### C. Speed Efficiency

To verify the efficiency of various approaches, we sample twenty pairs of NDKs and non-NDKs each for measuring the average speed of comparing two keyframes. A total of 40 queries is involved in this experiment. Same sets of parameter setting are used, i.e., $\alpha = 0.87$ for the matching strategies, $M = 0$ for LIP-IS, and $l = 2$ and $s = 108$ for LSH. Table V shows the average speed of matching one keyframe pair by different approaches. All methods are implemented with VC++ 6.0 coding, and tested on a 3GHz Pentium 4 machine with 512M memory. Color histogram is the fastest since it involves only two feature vectors, and optimal matching is the slowest due to the optimization procedure. LIP-IS and LSH significantly improve the matching speed of OOS by 12.5 and 29 times respectively, while LIP-IS can still maintain quite comparable matching effectiveness compared to pure OOS. LSH is about 2.3 times faster than LIP-IS, however its

TABLE V
SPEED EFFICIENCY FOR COMPARING TWO KEYFRAMES

| Method | CH | LIP-IS + OOS | LIP-IS + NN | OOS | NN | OM | LSH +OOS | LSH +NN |
|---|---|---|---|---|---|---|---|---|
| Time (s) | $10^{-5}$ | 0.028 | 0.027 | 0.35 | 0.34 | 44 | 0.012 | 0.010 |

matching performance is relatively poor in both NDK retrieval and detection. Our experiment indeed shows that LIP-IS can be more efficient than LSH when comparing some non-NDK pairs, simply because LIP-IS is effective in rejecting false matches, and the candidate size for OOS matching could be as small as $0 - 5$ LIPs in some cases.

In addition to the comparison of approximate search, we also compare OOS and LIP-IS with two exact search techniques under the support of R-tree [24] and kd-tree [25], [26] respectively. Due to the fact that our PCA-SIFT descriptors are in 36 dimensions, R-tree is indeed about two times (0.74 sec) slower than OOS. A large set of minimum bounding rectangles (MBRs), overlapping each other, are generated during R-tree indexing, resulting in a significant search overhead compared with simple linear search. The kd-tree indexing, which performs hyper-space partitioning without overlap, achieves 2.5 times speed-up (0.14 sec) than OOS. The improvement, however, is less significant compared to LIP-IS which speeds up efficiency by 12.5 times.

## IX. SUMMARY AND CONCLUSIONS

We have presented the proposed LIP-based approach for NDK retrieval and detection, with new techniques in matching, filtering and learning. Several critical issues are discussed, including (i) how to retain the reliable matches with one-to-one symmetric constraint, (ii) how to speed up matching while still ensuring the search quality with LIP-IS as filtering support, and (iii) how to learn matching patterns for detection. Empirical findings on TRECVID-2003 video corpus also demonstrate several interesting facts. First, the LIP based features are significantly better than the baseline color features for NDK retrieval. Second, reliable matches are the key factor to the success of LIP-based NDK identification and OOS matching indeed fits the requirement nicely. Third, LIP-IS has preference over LSH in terms of the number of required input parameters and the capability of approximating nearest neighbors. In the experiments, LSH basically performs poorer due to the fact than it often fails in locating the nearest neighbors. As a result, the matching strategy cannot rely on the candidates to produce quality matches.

Although encouraging, the current speed of LIP-IS (or even LSH) still cannot efficiently handle millions of keyframe pairs in large video corpus such as TRECVID benchmark which consists of several months of broadcast videos across sources. Nevertheless, one can still use the cues such as time constraint and NDK transitivity to restrict and to propagate the search of NDK pairs, as we have attempted recently in [27], for efficient linking of news stories. The first cue prunes unnecessary keyframe comparisons since the chances of finding NDK pairs become less likely when the recording time between them is far apart. The second cue can stop the searching of NDK pairs

as earlier as possible by keeping track of the NDK groups found so far, with the condition that the transitive closure of NDK pairs holds.

The applications of NDK detection come in naturally in broadcast domain, where NDKs can be exploited for threading multi-lingual news stories. Intuitively speaking, the visual-based near-duplicate detection has no language barrier and is more straightforward than text-based detection, though both modalities can complement each other as demonstrated in [2]. Since NDKs are typically the materials for reminding the evolution of news events over time and across languages, NDKs can form constraints to guide various multimedia tasks such as news video clustering and summarization.

## REFERENCES


[1] 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.

[2] X. Wu, C.-W. Ngo, and Q. Li, "Threading and autodocumenting news videos," *Signal Processing Magazine*, Mar 2006.

[3] Y. Ke, R. Suthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," in *ACM Multimedia Conference*, 2004, pp. 869–876.

[4] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[5] 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*, vol. 65, no. 1/2, pp. 43–72, 2005.

[6] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *IEEE Intl. Conf. on Computer Vision*, 2003.

[7] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descritors," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[8] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. 506–513.

[9] E. Chang, J. Wang, C. Li, and G. Wiederhold, "Rime: A replicated image detector for the world wide web," in *Proceeding of SPIE Multimedia Storage and Archiving Systems III*, 1998.

[10] K. Kashino, T. Kurozumi, and H. Murase, "A quick search method for audio and video signals based on histogram pruning," *IEEE Trans. on Multimedia*, vol. 5, no. 3, 2003.

[11] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Int. Conf. Music Information Retrieval*, 2002.

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

[13] S. C. C. . A. Zakhor, "Fast similarity search and clustering of video sequences on the world-wide-web," *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 524–537, 2004.

[14] TREC Video Retrieval Evaluation (TRECVID), in *http://www-nlpir.nist.gov/projects/trecvid/*.

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

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

[17] S. Smith, "A new class of corner finder," in *British Machine Vision Conf.*, 1992, pp. 139–148.

[18] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–393, 1981.

[19] P. I. . R. M. A. Gionis, "Similarity search in high dimensions via hashing," in *Int. Conf. on Very Large Data Bases*, 1999, pp. 518–529.

[20] Y. Rubner, C. Tomasi, and L. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[21] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin: Springer, vol. A, pp. 267-290, 2003.

[22] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," in *Int. Conf. on Computer Vision*, 2003.

[23] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.

[24] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of ACM SIGMOD*, 1984, pp. 47–57.

[25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[26] S. Arya and D. Mount, in *ftp://ftp.cs.umd.edu/pub/faculty/mount/ANN/*.

[27] C.-W. Ngo, W.-L. Zhao, and Y.-G. Jiang, "Fast tracking of near-duplicate keyframes in broadcast domain with transitivity propagation," in *ACM Multimedia Conference*, 2006.



PLACE PHOTO HERE

**Wan-Lei Zhao** received his M.S. and B.C. degrees in Department of Computer Science and Engineering from Yunnan University in 2006 and 2002 respectively. He was with Software Institute, Chinese Academy of Science from Oct.2003 to Oct.2004 as an exchange student. He is currently a research assistant in Department of Computer Science, City University of Hong Kong. His research interests include video computing and manifold learning.

PLACE PHOTO HERE

**Chong-Wah Ngo (M'02)** received his Ph.D in Computer Science from the Hong Kong University of Science & Technology in 2000. He received his MSc and BSc, both in Computer Engineering, from Nanyang Technological University of Singapore. Before joining City University of Hong Kong in 2002, he was with Beckman Institute of University of Illinois in Urbana-Champaign. He was also a visiting researcher of Microsoft Research Asia in 2002. His research interests include video computing and multimedia information retrieval.

PLACE PHOTO HERE

**Hung-Khoon Tan** received his MPhil in Computer Science from City University of Hong Kong, M.Eng in Microelectronics from Multimedia University (MMU) and B.Eng in Computer Engineering from University Technology of Malaysia (UTM). He was a test development and senior design engineer in Altera's R&D Center in Penang, Malaysia from 1999 to 2004. His research interests include multimedia content analysis, data mining and pattern recognition.

PLACE PHOTO HERE

**Xiao Wu** received the B.Eng. and M.S. degrees in computer science from Yunnan University in 1999 and 2002 respectively. He is a Ph.D. candidate in the Department of Computer Science at the City University of Hong Kong. Currently, he is at the School of Computer Science, Carnegie Mellon University, as a visiting scholar. From 2003 to 2004, he was with the Department of Computer Science of the City University of Hong Kong as a research assistant. His research interests include multimedia information retrieval and video processing.