Scalable Detection of Partial Near-Duplicate Videos by Visual-Temporal Consistency

Hung-Khoon Tan[†§] Chong-Wah Ngo[§]
{hktan, cwngo}@cs.cityu.edu.hk

Department of Computer Science[§]
City University of Hong Kong
Kowloon, Hong Kong

Richang Hong[†] Tat-Seng Chua[†]
{hongrc, chuats}@comp.nus.edu.sg

School Of Computing[†]
National University of Singapore
Singapore

ABSTRACT

Following the exponential growth of social media, there now exist huge repositories of videos online. Among the huge volumes of videos, there exist large numbers of near-duplicate videos. Most existing techniques either focus on the fast retrieval of *full* copies or near-duplicates, or consider localization in a heuristic manner. This paper considers the scalable detection and localization of *partial* near-duplicate videos by jointly considering visual similarity and temporal consistency. Temporal constraints are embedded into a network structure as directed edges. Through the structure, partial alignment is novelly converted into a network flow problem where highly efficient solutions exist. To precisely decide the boundaries of the overlapping segments, pair-wise constraints generated from keypoint matching can be added to the network to iteratively refine the localization result. We demonstrate the effectiveness of partial alignment for three different tasks. The first task links partial segments in full-length movies to videos crawled from YouTube. The second task performs fast web video search, while the third performs near-duplicate shot and copy detection. The experimental result demonstrates the effectiveness and efficiency of the proposed method compared to state-of-the-art techniques.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms Experimentation Performance

Keywords

Partial near-duplicate, temporal graph, network flow

MM'09, October 19–24, 2009, Beijing, China.

1. INTRODUCTION

With the popularity of social media, the volume of professional and user generated videos is growing exponentially. Among these massive amount of data, significant portion belongs to copies or near-duplicates. As a consequence, visual redundancy analysis becomes a topic of intensive studies recently [27, 16], being applied to various emerging applications including copy enforcement [13], news video threading [26], and novelty ranking of web videos [7]. Most existing techniques focus on the discovery of *full* copies or near-duplicates, where clips or shots are regarded as identical if there are sufficient amount of keyframes or features being duplicate or similar. This has largely fueled research into fast indexing techniques such as the locality sensitive hashing [13], hamming embedding [9], and random histogram [4]. While these techniques facilitate fast retrieval of duplicate videos, the localization of duplicate video segments are often being carried out in a heuristic manner. Typical examples include voting scheme [2] which counts the number of duplicates within different time stamps of a video to locate duplicate segments. Such heuristics become difficult to cope with the ever increasing amount of *partial* near-duplicate videos – a common practice in social media where interesting parts of a video are cut, edited, and then pasted in random positions at another video of similar or arbitrary theme.

The rapid populating of partial near-duplicates among videos indeed forms a media network that inter-relates different portions of videos. Understanding the topology of network offers advantages such as tracing the manipulation history of media [14] and video re-ranking by Page Rank like algorithm [7]. Nonetheless, in contrast to HTML web pages, "hyperlinks" of videos do not exist in reality and apparently automatic creation of such links to bridge the partial near-duplicate content of videos is not a trivial issue. Figure 1 depicts an example of several partial near-duplicate videos. In broadcast videos for example, videos with partial relationship narrate different aspects of an event or story which are not present in the other video. The degree of overlap between video pairs provides a more complete picture of their relationships by further outlining the hierarchy and dependency among them.

This paper addresses the issue of partial near-duplicate detection and localization. The links among videos are established through partially aligning video content. The partial alignment problem is novelly converted to a network flow problem. Using two videos A and B as example (the anchor and reference videos respectively in Figure 2), the network is

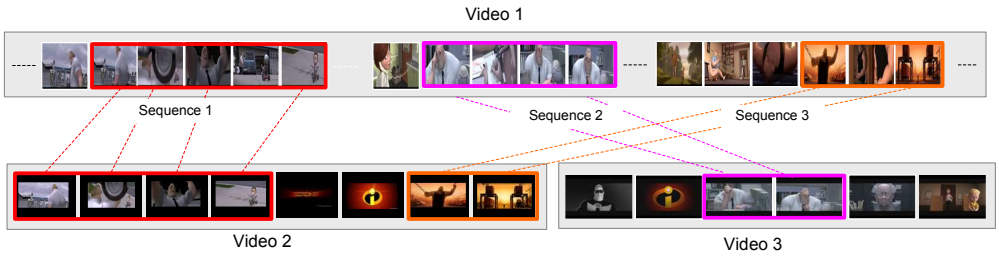

Figure 1: Partially near-duplicate videos. Given two videos, partial near-duplicate detection aims to detect and localize the near-duplicate segments in the videos.

formed by a set of frames in B retrieved by video A. The retrieved frames in B are chronologically traced following the original time stamps in B. This results in a variety of possible paths which can transfer flows of different capacities (or similarities in our definition) in the network. Finding partial alignment between videos A and B is then equivalent to searching for a maximal path which carries the maximum capacity. To precisely locate the boundaries of the overlapping segments, we also introduce two kinds of constraints: *must-link* and *cannot-link* to iteratively refine the partial alignment. The refinement could also aid in the discovery of multiple partial alignments, each of which follows certain temporal coherency. The must-links and cannot-links, which specify whether two frames from videos A and B should and should not be aligned, are continuously mined from the network along the process of searching a path with maximum capacity flow.

Compared to existing techniques in near-duplicate retrieval, the proposed approach offers several advantages as follows:

- *Partial localization* is considered. This is in contrast to fingerprint-based approaches [6] which summarize videos as compact signatures and sacrifice frame-level granularity in performing localization.
- *Joint visual-temporal detection.* In addition to visual features, the temporal coherency of frame sequence, an inherent feature embedded in the network, is also utilized for finding partial alignment. This is different from existing keyframe-based methods [20, 22, 26, 27, 28] which ignore temporal information and rely heavily on the retrieval or matching of point set features (e.g., SIFT from keypoints [17]) for robust detection, at the expense of intensive computation. With temporal coherency being considered, relatively "simple" features can be used to guarantee robust detection and localization, while enjoying good efficiency. In addition, the alignment is also less dependent on the "quality" of keyframe as in [27], where point set features cannot be accurately located if the selected keyframes have undergone motion blur or with fast moving objects. In fact, the detection rate of near-duplicates in [27] is dependent on the result of keyframe selection to certain extent. In our approach, keyframes are matched collectively where their temporal relationship are taken into account. This has resulted in a more robust approach that is less sensitive to the choice of keyframes.
- *Scalable alignment.* Expensive cross-frame and all-pair matching between two videos are not required by our approach. This is mainly due to the novel conversion of the problem to network flow optimization, where searching only needs to be performed among the frames of target video but not the query video. In addition, visual consistency of videos (represented via capacity) and temporal coherency are considered as a whole during search. These properties make our approach different from other frame alignment algorithms commonly used in the literature such as dynamic programming [8], Hungarian graph matching [15] and Earth Mover's Distance [25]. In these algorithms, visual and temporal information are considered separately, where sliding window techniques, which involve sequential scan of videos and exhaustive cross-frame matching, are often required to locate partial alignments. Using our approach, matching a full-length movie of more than 2 hours to 100 videos from YouTube requires only an average of 3 minutes.

In brief, the main contribution of this paper is the proposal of partial near-duplicate video alignment with network flow optimization. The proposed system performs *scalable* and *partial* localization of near-duplicates by joint consideration of visual and temporal consistency. In addition, the boundaries of partial alignments can be precisely located by incorporating the must-link and cannot-link constraints. With this work, we also tackle a variety of tasks that require the establishment of near-duplicate links among videos for various purposes. The first task links partial segments of full-length movies to videos crawled from YouTube. These links provide clues to detect the highlight of videos, while enriching the annotation of movie scenes with social tags. The second task performs fast web video search by rapid detection of near-duplicates in a database of more than 500 hours. The last two tasks perform near-duplicate shot detection and copy detection respectively on TRECVID [19] and Muscle-VCD-2007 datasets [11].

The remaining paper is organized as follows. Section 2 reviews related work on near-duplicate retrieval and detection. Section 3 presents temporal network modeling to detect near-duplicate segments from two videos. Section 4 explore the use of must-links and cannot-links derived from keypoint matching to refine the boundaries of the detected segments. The proposed approach is evaluated under three task settings, i.e., movie tagging (Section 5), web video retrieval (Section 6) and near-duplicate and copy detection (Section 7). We then summarize our findings in Section 8.

2. RELATED WORK

Near-duplicate retrieval and detection, as a timely research problem to several emerging applications, has been intensively studied recently. Broadly, we can categorize the existing works into three main groups: signature-based [6, 4], keyframe-based [20, 22, 26, 27] and trajectory-based [24,

16]. The definition of near-duplicate indeed varies depending on the target application. In general, near-duplicates are composed of 1) videos of the same scene but captured under different viewpoints, lighting and times, and 2) videos modified from the same source of material, resulting in copies of different versions.

Signature-based approaches summarize video content into fingerprints for fast retrieval. Typical example includes the use of global color histogram to "average" frames in video as a tiny fingerprint [22]. More advanced techniques include the recently proposed random histogram [4] which projects low-level features and embed them into a high dimensional space using locality sensitive hashing. The resulting fingerprint is not compact but sparse enough so that indexing technique such as vector space model can be used for rapid retrieval. While being efficient, temporal information is missing in fingerprints and thus retrieval of partial near-duplicates is not supported. Context-based approaches, similar to fingerprint generation, derive signatures from the context surrounding video. For instance, the recent work in [23] utilizes web context such as thumbnail, view count and time duration of a video for real-time web video re-ranking.

Keyframe-based approaches perform sparse analysis of video content by matching representative frames sampled from videos [20, 22, 26, 27]. Two classical techniques are sliding window [22] and dynamic programming [8]. Sliding window is sensitive to temporal resolution and is thus not suitable for retrieving near-duplicates with changes in frame rate, especially the fast and slow-motion videos. Dynamic programming finds the longest common subsequence and computes the edit distances to determine near-duplicate identity. However, the role of dynamic programming is typically limited to extracting temporal entities on the correspondence set generated from content matching. In the end, the robustness of the approaches relies heavily on the stability of visual content. The recent work in [20] optimizes frame alignment by considering visual similarity and alignment distortion. The two frame sequences are aligned horizontally as points in an image space and alignment distortion is based on the agreement among the angles in the corresondence set. However, the overhead to perform alignment could be quite significant for long sequences, considering that a quadratic integer optimization is employed. Heuristic voting scheme is also proposed in [2] where the aggregation of votings from near-duplicate frames makes the near-duplicate shot detection robust to individual near-duplicate image false positives. However, voting can only generate coarse alignment results which are not fully optimized and might not be adequate for precise localization of partial near-duplicates.

Trajectory-based approaches track points of interest along the video sequence to enrich keypoint features with spatio-temporal information. For instance, trajectories have been utilized to highlight different motion behaviors [16] and then assign behavioral labels to each local descriptors. In another recent work [24], the whole shot is represented using a bag of trajectories where each trajectory in turn is described as temporal patterns of discontinuities. In general, the extraction of trajectories is an extremely expensive operation. Moreover trajectory features are sensitive to camera motion and therefore their robustness is limited to copy detection, but not necessarily robust for general near-duplicate detection, especially those involving viewpoint changes.

3. TEMPORAL NETWORK

Given two videos, a video is designated as the *anchor* video Q and the other as the *reference* video R. Temporal network is initially formed by querying the top-k similar frames from R using Q. Figure 2 illustrates an example where an anchor video consisting of six frames retrieves six list of top-k frames from the reference video. Directed edges are established across the frames in top-k lists by chronologically linking frames according to their time stamp values. For example, with reference to Figure 2, the frame in the first list with time stamp value 5 can link to one frame in another list with time stamp larger than 5. In other words, when tracing the list of connected edges from left to right, the time stamp values are monotonically increasing. This form a *temporal network* encompassing all possible frame alignments between videos Q and R which follow strict temporal coherency. Two artificial nodes, *source* and *sink* nodes, are included for modeling so that all paths in the network are originated from the source node and end at the sink node.

In brief, temporal network G is basically composed of nodes from frames in reference video R, and directed edges which traverse frames from source to sink nodes. The weight of an edge is proportional to the similarity of the destination node to its query frame in Q. In this network, the weight signifies the capacity that an edge can carry. The network flow which a path can transport is equal to the accumulated weights of its edges from the source to sink nodes. Finding a maximal path with the maximum flow is thus equivalent to searching for a sequence alignment which maximizes the similarity between Q and R in monotonically increasing temporal order. The optimization is indeed an equivalent of the classical network maximum flow problem in operations research [1].

While the network appears to be simple, it actually signifies several interesting facts. First, visual and temporal information are unified in a holistic way under the network. This allows joint consideration of visual-temporal consistency between Q and R – an important property which existing techniques for near-duplicate detection does not take into account. Second, while the number of possible paths may be exponential, the temporal constraint imposed to the network can actually prune a significant number of paths. In particular, falsely retrieved frames which are often randomly positioned in the network can now be easily filtered out with the use of temporal constraint. A by-product of this joint visual-temporal consistency is that simple fingerprint features can be employed for top-k retrieval, rather than the point set features such as the SIFT of keypoints [27] which are expensive to compute.

3.1 Notation

The proposed temporal graph models the set of all viable routes that obeys temporal consistency. The viable routes in turn represent the set of possible alignment solutions that we intend to optimize. Sequence matching is thus posed as a transportation problem where the objective is to find the optimal path to transport one unit load from the source node to the sink node.

For ease of understanding, we shall use the following notations in the remainder of this paper.

- Denote $Q = \{q_1, \ldots, q_{|Q|}\}$ as the query video and $R = \{r_1, \ldots, r_{|R|}\}$ as the reference video where $|\cdot|$ denotes the cardinality of a set.

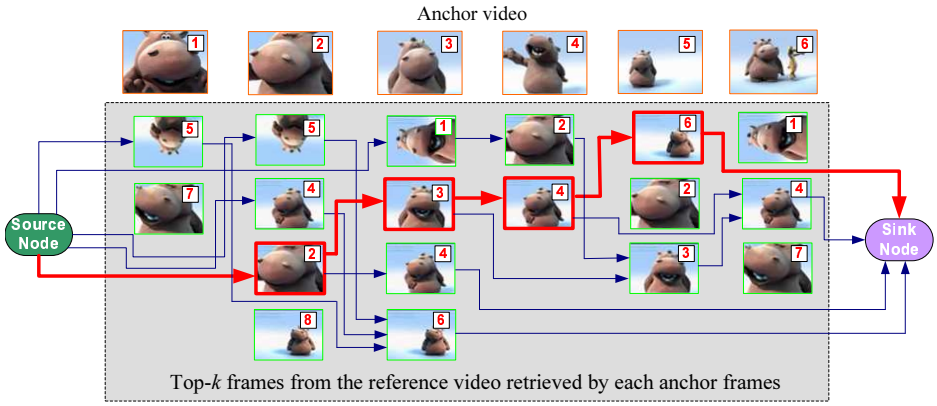

Figure 2: A Temporal Graph. The columns of the lattice are frames from the reference videos, ordered according to the k-NN of the query frame sequence. The labels on each frame shows its time stamp in the reference video. The optimal path is highlighted. For ease of illustration, not all paths are shown.

- Temporal Network is denoted as $G = (\mathbf{N}, \mathbf{E})$ where $\mathbf{N}$ and $\mathbf{E}$ are the set of nodes and edges respectively. The source and sink nodes are denoted as n_{src} and n_{sink} respectively.
- $\mathbf{N} = \{N_1, \ldots, N_{|Q|}\}$ where $N_i = [n_1, ..., n_k]$ contains the top-k frames from R retrieved by q_i, and ordered according to their similarity values. Thus, n_j denotes the j^{th} nearest frame to q_i. One query function is defined: $TS(n)$ returns the time stamp, or frame number, of a node n.
- $\mathbf{E} = \{e_{ij}\}$ is the set of all edges. e_{ij} represents a weighted directed edge linking any two nodes from top-k lists N_i to N_j, respectively.
- Given an arbitrary node n in G, $SRC(n)$ is the direct predecessors of n, or the set of nodes with one of its out-going edge directed towards n. Similarly, $DEST(n)$ is the direct successors of n, or the set of nodes with an in-coming edge emitted from n. $E_{in}(n)$ is the set of in-coming edges while $E_{out}(n)$ is the out-going edges of n.
- $A_Q = \{a_1, \ldots, a_L\}$ and $A_R = \{b_1, \ldots, \ldots, b_L\}$ represent the best maximal frame alignment between Q and R, or equivalently A_R is the optimal path in G. The alignment is one-to-one (e.g., frame a_1 matches to b_1) and L denotes the sequence length. With reference to Figure 2, $A_Q = [2, 3, 4, 5]$, $A_R = [2, 3, 4, 6]$ and $L = 4$.

3.2 Frame Alignment versus Flow Optimization

The solution space provided by the network encompasses various uncertainties of alignments arisen from differences in temporal resolution and feature similarity. Under our problem definition, we seek an optimal alignment between Q and R:

$$A_Q, A_R, L = \max_{A_Q, A_R, L} \sum_{i,\, a_i \in A_Q, b_i \in A_R}^{L} Sim(q_{a_i}, r_{b_i}) \qquad (1)$$

subject to

$$a_{i+1} > a_i, \quad b_{i+1} > b_i \qquad (2)$$

$$1 \leq a_L \leq |Q|, \quad 1 \leq b_L \leq |R| \qquad (3)$$

where Sim represents the similarity of frame q_{ai} and r_{bi}. Equations 2 and 3 imposes temporal order and one-to-one (O2O) mapping relationship between A_Q and A_R. Many-to-many (M2M) or one-to-many (O2M) relationship is indeed possible by altering the inequality operators in Equation 2. Generally M2M or O2M mapping is more robust, for example when matching a video to a slow-motion version. However, the randomness caused by such relaxed mappings can lead to noisy matches in practice. For practical and efficiency reasons, O2O mapping is adopted in our formulation.

Equation 1 can indeed be solved in its original form as a constrained binary programming problem. Possible solution includes branch-and-bound algorithm by treating Equation 2 as constraint and evaluating all possible permutation of L. The evaluation can easily become intractable with the increase in the video length – a case where we consider when matching a full-length movie to web videos. With temporal network G, in contrast, the temporal constraint in Equation 2 is *structurally* embedded into G as directed edges. This structural embedding novelly converts the alignment problem into a transportation problem, or more specifically a network flow problem [1], where efficient algorithms are readily available.

In the network flow problem, each directed edge is characterized by two terms: weight and flow. Given an edge e_{ij} directed from any arbitrary node from N_i to another node in N_j, the weight of e_{ij} is defined as:

$$w(e_{ij}) = Sim(q_j, n_j) \qquad (4)$$

where q_j is the j^{th} query frame from Q, and n_j is the node in N_j. For any edge terminating at the sink node, the weight is assigned to zero. The flow of e_{ij}, under our problem definition, is a binary indicator with value equal to 1 or 0. Given a particular solution, the flows at the edges traversed by the path is 1 while for all other edges, the flow value is 0. One important criterion to ensure a valid solution is that the path given by the solution must not be broken at any point between the source and the sink node. To meet this requirement, the temporal network must obey the flow conservation constraint where the net inflows and the outflows at a particular node must be equal to zero.

Denote $f(e_{ij})$ as the flow at edge e_{ij}. The objective is find the optimal values of $f(e_{ij})$ that maximizes the total accu-

mulated weight and at the same time obeys the equilibrium requirement. The frame alignment, based on network flow optimization, is thus formulated as:

$$\text{maximize} \sum_{e_{ij} \in \mathbf{E}} f(e_{ij}) w(e_{ij}) \quad (5)$$

subject to

$$\sum_{e_{in} \in E_{in}(n)} f(e_{in}) - \sum_{e_{out} \in E_{out}(n)} f(e_{out}) = 0, \quad \forall n \in \mathbf{N} \quad (6)$$

$$\sum_{e_{out} \in E_{out}(n_{src})} f(e_{out}) = 1 \quad (7)$$

$$\sum_{e_{in} \in E_{in}(n_{sink})} f(e_{in}) = 1 \quad (8)$$

$$0 \leq f(e_{ij}) \leq 1, \quad \forall e_{ij} \in \mathbf{E} \quad (9)$$

where Equations 6, 7 and 8 impose the *flow conservation* constraints to control a well-behaved weight transfer from the source to the sink node. The set of nodes traversed by the optimal path indicated by $f(\cdot) = 1$ constitutes the solution. Comparing equations 5 and 1, the optimal path corresponds to A_R, and A_Q is easily obtained with no effort once A_R is known.

The network flow formulation is a special constrained linear program with two interesting properties [1]. First, since the right hand sides of the constraint equations are binary, the *unimodalarity* property guarantees that the solution must also be binary and are therefore useful as an assignment problem such as ours. Second, since the coefficients of the left hand side in the constraint equations are either 0 or +1, there exist very efficient algorithms for the optimization problem. We adopt the *network simplex* algorithm in [5] for our approach. The time complexity of the algorithm is $O(B \times C)$ where B and C are the number of nodes and edges in the structure respectively. The upper bound for B is $O(k \times |Q|)$ while the upper bound for C is $O(U \times W)$ where $U = \frac{k(k-1)}{2}$ is the maximum number of directed edges between the nodes in any two top-k lists, while $W = \frac{|Q|(|Q|-1)}{2}$ is the number of permutations of two top-k lists out of $\mathbf{N}$.

3.3 Heuristic Temporal Pruning

The efficiency of flow optimization algorithm can be further improved by imposing heuristics to simplify the construction of the temporal graph G. A simple example is to reduce the length of top-k list by excluding frames whose similarities are insignificant. We introduce a total of seven constraints with three additional heuristic parameters (wnd, $\mathfrak{T}$, L_{min}) which will be explained later. These constraints jointly specify the conditions for establishing an edge e_{ij} between node b from N_i and node c from N_j, where $j > i$:

$$\begin{aligned}
C1 &: \ 0 < j - i \leq wnd \\
C2 &: \ 0 < TS(c) - TS(b) \leq wnd \\
C3 &: \ \neg \exists x \in N_k : TS(b) \leq TS(x) \leq TS(c), i < k < j \\
C4 &: \ w(e_{ij}) \geq \mathfrak{T} \\
C5 &: \ L \geq L_{min} \quad \text{or else} \quad L = 0 \\
C6 &: \ E_{in}(r) \neq \emptyset, \ j \geq |Q| - L_{min} \\
C7 &: \ E_{out}(s)) \neq \emptyset, \ j < L_{min}
\end{aligned}$$

Algorithm 1 M = GetMaximalPath (G)

```
1: M_f ← GetMaxDur (G, Forward)
2: M_b ← GetMaxDur (G, Backward)
3: M(·) ← M_f(·) + M_b(·)
```

Algorithm 2 DUR = GetMaxDur (G, $DIRECTION$)

```
1: if DIRECTION = Forward then
2:    start = 1, end = |Q|, neighbors ≡ DEST
3: else
4:    start = |Q|, end = 1, neighbors ≡ SRC
5: end if
6: DUR(·) = 0
7: for i = start to end do
8:    for n_i ∈ N_i do
9:       for m_j in neighbors(n_i) do
10:         DUR(m_j) = max(DUR(m_j), DUR(n_i) + |TS(m_j) −
            TS(n_i)|)
11:      end for
12:   end for
13: end for
```

C1 and C2 are the temporal constraints (originally discussed in the first paragraph of Section 3) to guarantee that any path in the network is linked chronologically according to the time stamp. The parameter wnd is used to specify the tolerable level of temporal distortion, where the difference between the time stamp values of two successively aligned frames specified by A_Q and A_R must not exceed wnd.

C3 trims all sub-optimal edges that will clearly be ignored in the optimal solution. For example, given an edge e_{ij}, if there exists a node x in the middle list N_k through which an alternate path can be established from b to c, then e_{ij} is essentially redundant since $w(e_{ij}) < w(e_{ik}) + w(e_{kj})$ and $w(e_{ij}) = w(e_{kj}) = Sim(c, q_j)$. In addition, although the signatures of two frames are not accurate enough to identify near-duplicate frames, it is sufficiently reliable to remove non near-duplicate frames if their similarity value is too low. The threshold $\mathfrak{T}$ in C4 specifies the minimum similarity in order for a node to be considered in our model.

C5 states that the near-duplicate segment is expected to be at least L_{min} in length. For instance, when matching online videos to full-length movie video, the degree of overlap should be proportional to the length of the online video in order to be considered a positive match. To eliminate the set of all sequences shorter than L_{min} from the structure, Algorithms 1 and 2 (given above) are proposed to find the *maximal length* for all nodes. Given a node r, its maximal length $M(r)$ is the length of the longest path from the source node n_{src} to the sink node n_{sink} that passes through r. Nodes whose maximal lengths are shorter than L_{min}, are removed from the structure together with their edges. This is accomplished by breaking the path into the two parts. The network is first scanned in the forward direction to retrieve $M_f(r)$, the maximal length from the n_{src} to r, and then in the backward direction to retrieve $M_b(r)$, the maximal length from the r to n_{sink}. The process is an efficient one since the calculation of maximal paths can be conducted simultaneously for all nodes in a single run.

The constraints C6 and C7 are the direct consequences of C5. Any sequence, whose first frame are from N_i where i is larger than $|Q| - L_{min}$, must be shorter than L_{min} and is discarded. Similarly, any sequence, whose last frame is from N_j where j is smaller than L_{min}, should not be considered.

With the constraints, the complexity of the network simplex algorithm introduced in Section 3.2 can be significantly reduced. C1 and C2 reduces the upper bound for both the edge and list permutation parameters U and W to $\frac{wnd(wnd-1)}{2}$ where $wnd \ll k$ and $wnd \ll |Q|$. C3 reduces the number of edges C by removing sub-optimal edges. C4 removes all reference frames whose similarities to its corresponding anchor frames are too low. This reduces the number of nodes B especially when matching unrelated videos. C5-C7 results in further reduction of both B and C by trimming unrealistic solutions that violate the user's expectation.

4. DETECTION FRAMEWORK

The proposed algorithm essentially extracts the chronological sequences with the set of most visually similar frame-pairs. The method, however, still lacks the mechanism to isolate random noisy sequence from genuine ones. Towards this end, we integrate keypoint matching into the detection framework, partly to verify the near-duplicate identity of the sequences, and also to refine the boundaries for precise localization. Keypoint matching has been shown to provide very reliable detection result between two near-duplicate frames. As an example, the publicly available keypoint-based NDK algorithm [27] that we use in this paper reported a precision of 94%. Given the alignment result from sequence matching, the corresponding frame-pairs are subjected to keypoint matching to confirm their near-duplicate identities.

Interestingly, the result from keypoint matching can be re-used as the *a priori* information to perform refinement of the alignment result. Possible solution includes reconfiguring the constraints where the lower bounds for the edges routing to a near-duplicate node (a reference frame that has been detected as a near-duplicate to its corresponding anchor frame by keypoint matching) are set to 1 and conversely, the upper bounds for the edges routing to a non near-duplicate node are set to 0. However, direct application of the constraints would inevitably have a noticeably adverse impact on runtime depending on the number of iterations. Motivated by recent researches on constraint-based clustering [21], the pair-wise constraints are integrated into sequence matching in a manner that takes into account the topology of the temporal network. One major difference is that, in our approach, the constraints are dynamically generated and increase in significance as more constraints are accumulated over the iterations.

Figure 3 illustrates the outline of the proposed framework. Duplicate frame-pairs detected from keypoint matching are marked as *must-links* and non-duplicate pairs as *cannot-links*. The results from keypoint matching are used to revise the network structure with two objectives in mind, to refine matching while maintaining a competitive speed. Here, the must-link nodes are novelly used as stubs to cut the network into a series of non-overlapping smaller networks. Partitioning a full optimization problem into multiple smaller ones results in considerable time saving. Assuming that the network is partitioned uniformly into L number of sub-networks, the number of nodes B and edges C in each sub network is L times lesser than the original structure, resulting in a significant reduction in time complexity from $O(B \times C)$ to $O(\frac{B \times C}{L})$.

Moreover, network construction can be completed effortlessly in an incremental manner. The must-links, which define the boundaries between fragments, will play the new role

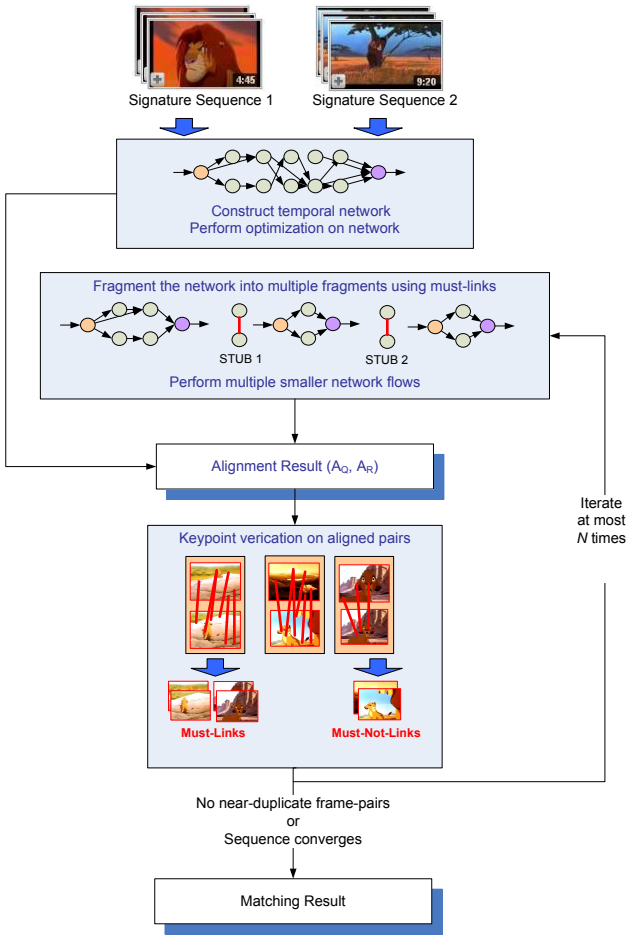

Figure 3: Verification process.

as the source and sink nodes for the respective fragments. Nodes that are not connected to the partition's source and sink node are removed from the structure together with their edges. Since partitioning the network also splits the near-duplicate sequence into multiple shorter portions, the parameter L_{min} should be either relaxed or disabled in subsequent iterations.

4.1 Multiple Partial Alignments

In practice, two partially similar videos may contain multiple randomly positioned near-duplicate segments. The relationships between potential near-duplicate segments in two different segments may come in three possible configurations, as depicted in Figure 4, namely, sequential alignment, cross alignment and self-duplication. These configurations can indeed be handled by our approach.

The handling of both sequential and cross alignments is essentially the same. Referring to Figure 2, note that the reference frames are repeated in the various top-k lists based on their similarity to the anchor frames. As such, the partitions separated by the detected segments, are self-sufficient such that network flow optimization can be carried out separately without much interference from the rest. Network reconstruction is fast since it can be done incrementally. The structure can be derived from the original structure (before the iterations) with two simple modifications. First, the reference frames which have been identified as near-duplicates

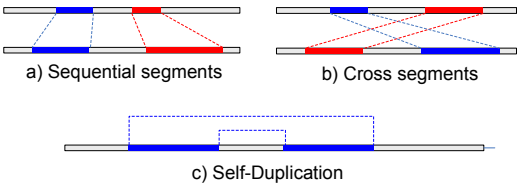

Figure 4: Different configurations of multiple partial near-duplicates.

from previous runs are removed together with their edges. Second, the edges routing to or from the top-k lists of the previously detected segments are re-routed either to the sink or source nodes. The process is highly efficient as each partition is significantly smaller compared to the original structure. The step is repeated multiple times until no segments that fulfill L_{min} can be found.

The other configuration, self-duplication occurs when there are repetitive segments within the same video. The scenario is common especially in news broadcast videos where continuous updates are given over short intervals on the development of a particular story. Self-duplication can be handled similarly by treating the video itself as both the query and reference videos. It only requires one extra heuristic, i.e., frames with the same time stamp are not allowed to be aligned.

5. MOVIE TAGGING

In this section, we showcase the effectiveness of sequence matching for the task of automatic movie tagging. With the emergence of media sharing websites such as YouTube, there has been a growing trend for fans to extract and publish online interesting scenes from their favorite movies, together with tags, comments and descriptions. To make use of these readily available resources, one fundamental task is to link partial segments from a full-length movie to online videos crawled from these websites to propagate the social tags to the correct segments. The main challenge in matching with a movie which can last for several hours is efficiency and therefore a fast matching algorithm is required.

Experiment Setup. We test on 5 full-length movies listed in Table 1 where 100 videos are crawled from YouTube for each movie using the movie's title. For labeling, only online videos containing segments that preserve the flow of the original movie content are labeled as the partial near-duplicates of the movie. Trailers, music-videos, movie reviews and commentaries, though containing random scenes from the movie, do not carry any useful context for movie tagging and therefore are labeled as non near-duplicates.

For signature representation, simple edge histogramming with cosine distance is used. By default, the alignment parameters are set as follows: nearest neighbor $k = 20$, temporal distortion $wnd = 15$, minimum expected length $L_{min} = 0.5 \times |Q|$, similarity threshold $\mathfrak{T} = 0.3$ and verification iterations $N = 3$. For the iterative keypoint verification component (refer to Figure 3), local interest points are extracted by using the Difference of Gaussian (DoG) detector and PSIFT descriptor [27] while keypoint matching is conducted using [28] and then verified using SR-PE [27]. SR-PE considers pattern coherency in order to handle arbitrary scale and rotation transformations of near-duplicate regions. The codes are written in C# and the experiments are conducted on an Intel Core 2 Duo 3.0GHz CPU with 3GB of RAM [1]. In the remaining of the paper, we shall refer to our proposed method as TNP.

Results. Table 1 shows the matching result. In general, the proposed method delivers an impressive performance, achieving a precision of 0.96 and and recall of 0.83. In terms of localization accuracy, the degree of overlap between the query video segments and the located segments in the main movies is as high as 81% in average. More importantly, our approach is highly efficient, requiring only 3.3 minutes in average to complete processing one movie. This is because the alignment speed is determined mainly by the length of the shorter sequence, in our case, the online videos.

Movie videos are often available in two versions, wide screen or full screen. One important observation is that when matching different versions of movie videos, the difference in the aspect ratio may cause keypoint matching to fail due to severe stretching. Although the near-duplicate segment can be accurately determined by the alignment algorithm, SR-PE cannot recognize near-duplicate frames and as a result the recall performance is affected. To alleviate the problem, the online videos are transformed into the same aspect ratio setting as the movie video in our experiments.

6. WEB VIDEO RETRIEVAL

In this section, we evaluate the performance of our system on a full duplicate scenario, where the task is to retrieve near-duplicate videos from a web video corpus. The web video dataset in [22] is used for evaluation. The dataset consists of 12,790 web videos collected from YouTube, Google and Yahoo, using 24 search queries and the total length of all the web videos is over 540 hours. We use 24 seed videos, one per query, as video examples for near-duplicate retrieval. The seed videos are the set of popular videos which are most viewed by users in the video sharing websites. Readers are referred to [22] for more details on the dataset.

Experiment Setup. Following the setup in [22] for comparison purposes, we use visual keywords of 1000 clusters with hard assignment as signature. Keypoints are extracted using Hessian-Affine detector [18] and described by PCA-SIFT features [12]. For parameter setting, the following setup is used: $N = 1$, $k = 3$, $wnd = 15$, $L_{min} = 0.1$ and $\mathfrak{T} = 0.75$. Compared to edge histogram, a higher value for $\mathfrak{T}$ is needed for visual keyword with hard assignment since the feature is rather sparse. To evaluate the performance of TNP, we compare it with two state-of-the-art methods for near-duplicate video retrieval. The two methods are QIP [20] which considers both visual similarity and alignment distortion in a quadratic integer programming formulation and HIRACH [22], which uses a hierarchical framework where color histograms are used for initial filtering and keypoint matchings are used to refine the result. The global color signature (CHSIG), possibly the simplest approach to measure video similarity, is used as the baseline to judge the improvement that the alignment algorithms can achieve.

Result. Table 2 shows the retrieval performance of the four approaches in terms of mean average precision (MAP) over the 24 web video queries. TNP outperforms the baseline CHSIG by a wide margin with only a slightly lower performance than QIP and HIRACH. HIRACH, although

[1]We use the same platform and settings (DoG, PSIFT and SR-PE) for the experiments in Section 5 to 7 unless specified otherwise.

Table 1: Experiment Result on Movie Dataset

	#Keyframes in the movie video	Average #Keyframes in YouTube videos	#Positive samples	Precision	Recall	Degree of overlap	Runtime (min)
3:10 to yuma	2686	68	10	1.00	0.90	0.89	1.6
BraveHeart	2816	80	4	1.00	0.75	0.58	0.9
Syriana	2323	48	21	0.95	0.90	0.88	3.8
The Incredibles	2395	119	29	1.00	0.83	0.81	6.0
The Last Emperor	3538	78	14	0.85	0.79	0.90	4.1
Average	2752	79	10	0.96	0.83	0.81	3.3

Table 2: Web Video Retrieval Result

	TNP	QIP	HIRACH	CHSIG
MAP	0.935	0.951	0.952	0.891
#Frame-pair comparisons[2]	114,154	238,769	6,471,693	-
Total time for keypoint matching & verification	2.9 hours	5.9 hours	∼7 days	-
Align speed per frame-pair	28 ms	11 s	-	-
Total align time	9.8 min	1.4 days	-	-
Total runtime	3.1 hours	1.7 days	∼7 days	1 sec

generating impressive results, is relatively slow because of a semi-brute force approach which requires a significant number of keyframe-pair comparisons. Furthermore, keypoint matching is a computationally expensive operation. To improve scalability, both TNP and QIP consider only likely candidate keyframe-pairs. In general, QIP generates a comprehensive correspondence list where most frames in the shorter sequence would be matched. On the other hand, TNP generates a shorter alignment sequence, requiring 40% less frame-pair computations, but becomes more susceptible to misalignment in the process. In addition, TNP is 95 times faster than QIP, requiring only 28 miliseconds compared to 11 seconds for QIP to align two videos. CHSIG, on the other hand, is fast but is unable to handle complex near-duplicate videos with major editing and transformations, and thus has the lowest retrieval performance.

Parameters Sensitivity. In this section, we evaluate the sensitivity of the system towards the various parameters. The experiments are obtained by varying the respective parameters while fixing the others to $L_{min} = 0$, $wnd = 5$, $k = 20$ and $N = 0$ (SR-PE verification, no iteration). The experiment results are shown in the Tables 3, 4, 5 and 6. In general, the precision is not sensitive to the parameter settings. Only when the parameters are set too rigid ($wnd = 1$ or $L_{min} = 0.5$), the performance will be affected. In terms of speed, the parameters k and wnd have a more significant impact on the runtime. To minimize the impact to runtime, L is found to be particularly useful, where the runtime decreases steadily when varied from δ is varied 0 to 0.5 where $L_{min} = \delta \times |Q|$. In addition, without verification ($N = -1$), the ranking given by the alignment result still outperforms CHSIG, with MAP=0.922. Moreover, when keypoint matching is enabled ($N \geq 0$), the performance shows considerable improvements which confirms the iterative verification framework. In addition, the algorithm shows only a subtle increase of 30% in runtime when the number of iterations N increases to 3. This justifies the strategy to break up the network into multiple fragments, which otherwise will result in a linear increase in runtime.

[2][27] reported an average speed of 90ms to process one frame-pair.

Table 3: Sensitivity towards top-k

k	1	3	5	10	20
MAP	0.945	0.946	0.940	0.934	0.934
Align Speed (ms)	14	15	22	46	94

Table 4: Sensitivity towards wnd

wnd	1	5	10	15
MAP	0.852	0.934	0.936	0.937
Align Speed (ms)	43	94	180	259

Table 5: Sensitivity towards $L_{min} = \delta \times |Q|$

δ	0	0.05	0.1	0.25	0.50
MAP	0.934	0.935	0.936	0.936	0.925
Align Speed (ms)	94	65	50	36	26

Table 6: Sensitivity towards N.

N	-1	0	1	2	3
MAP	0.922	0.934	0.935	0.934	0.934
Align speed (ms)	93	93	108	110	119
#SR-PE	0	106,483	115448	120626	125225

$N = -1$: verification mode disabled.
$N = 0$: SR-PE verification (no iterations).
$N > 0$: SR-PE verification (with iterations).

7. NEAR-DUPLICATE AND COPY DETECTION

7.1 Copy Detection and Localization

In this section, we showcase the effectiveness of the proposed system to detect video copies and localize copied segments from videos. We perform our experiments on the Muscle-VCD-2007 video dataset [11], which is the evaluation set used in the video copy evaluation in CIVR 2007. Two tasks were competed. Given a query video, task 1 retrieves copies of whole long videos while task 2, a much harder task, detects and locates the partial-duplicate segments from all the videos. For task 1, a total of 15 videos spanning over 2 hours 30 minutes with various transformations are used as queries. For task 2, a total of 21 transformed extracts are inserted into 3 query videos with a total duration of 45 minutes. In the test set, there are 102 videos with a duration of 100 hours covering a diverse variety of viewing materials and programs. We use visual keywords (500 codewords with soft-weighting [10]) for signature representation and the cosine distance for distance measure. The parameters settings are $N = 3$, $K = 5$, $wnd = 5$, $\mathfrak{T} = 0.3$ and L_{min} is set to 0.5 for task 1 and -1 for task 2.

Detection Performance. For task 1 (copy detection), a *reranking* framework is employed. An initial list is retrieved with the keypoint verification feature disabled ($N = 0$). Given a query video, the test videos are ranked based on

Table 7: Muscle-VCD-2007 Detection Result

	TNP	Team			
		ADV	IBM	CITYU	CAS
Precision	1	0.86	0.86	0.66	0.53
Runtime (min)	8.5	64	44	45	15

Table 8: MUSCLE-VCD-2007 Localization Result

	TNP (for different N)				Team	
	0	1	2	3	CITYU	ADV
QS	0.90	0.90	0.90	0.90	0.86	0.33
QF	0.79	0.81	0.82	0.82	0.76	0.17
Runtime (min)	2.44	2.60	2.85	2.97	35	33

the accumulated similarity values of the aligned frame pairs. The algorithm is then repeated on the top rank (top 1) videos with iterative SR-PE verification enabled ($N = 3$).

The results are shown in Table 7. The performance is calculated based on the detection precision of the top positioned video in the retrieved list. The best official results for all teams who participated in the evaluation are reproduced here for comparison. Our algorithm manages to retrieve the perfect score in this experiment. In fact, all video copies are correctly pulled to the top-most position in the list even in the initial run. More importantly, our algorithm is very efficient. Eventually, the proposed reranking framework invoked only 551 frame-pair computations. The fastest runtime by the other methods is 15 minutes (ADV) but at the expense of accuracy where the precision is only 0.53. In contrast, our method delivers the best possible result with a runtime of only 8.5 minutes.

Localization Performance. For task 2 (copy localization), two criteria are used to evaluate the performance, where $QF = 1 - \frac{|MissedFrames|}{|Frames|}$ to evaluate the overlap accuracy for the 21 segments and $QS = \frac{|Correct|-|FalseAlarm|}{|Segments|}$ is used to evaluate the detection accuracy of the segments. Since no prior knowledge of the partial copy sequence is available, L_{min} is set to -1. Table 8 shows the result for this experiment. In general, the boundaries specified by TNP is very accurate with the highest score of $QF = 0.82$ and $QS = 0.90$. Our results are superior to the reported results from CITYU and ADV systems. As the number of iterations N increases, the localization performance increases steadily, indicating the boundaries of partial duplicates get increasingly accurate. Again, the increase in runtime is observed to be not significant as N increases.

7.2 Near-duplicate Shot Detection

In this section, we evaluate our algorithm for general near-duplicate detection. In particular, we show that NDK detection is sensitive to the choice of keyframes to certain extent since it is difficult to select a single frame that can consistently represent the set of all near-duplicate shots in the dataset. To overcome this problem, we experiment the use of multiple frames to represent each shot. When comparing two shots, TNP is employed to align the two frame sequences and near-duplicate detection is carried out on the alignment result. If one of the frame pair is detected as near-duplicate, the two shots will be regarded as near-duplicate.

Experiment Setup. For evaluation, we use the NDK-7006 dataset [27] which is collected from TRECVID 2003 [19]. The experiment is conducted in two runs. First, keyframe-based near-duplicate detection is tested on the 3384 positive keyframe pairs in the ground truth, as well as on a set of 6000 negative keyframe pairs randomly selected from the dataset. In the second run, the multiple frame representation is evaluated where TNP is used to align the frame sequences. Each shot is segmented into multiple sub-shots through simple visual thresholding (set to 0.85) using color histograms and the frame with the highest accumulated similarity to all other frames is selected to represent each sub-shot. For the alignment parameters and frame-level representation, the same settings as the copy detection task are used.

Table 9: Near-duplicate shot detection result.

	SR-PE	TNP
True Positive	2588 (76%)	2839 (84%)
False Negative	796 (24%)	545 (16%)
True Negative	6000 (100%)	5999 ($\approx$ 100%)
False Positive	0 (0%)	1 ($\approx$ 0%)

Result and Discussion. Table 9 shows the result of the near-duplicate shot detection. Using only one keyframe per-shot, SR-PE correctly detects 2588 positive frame pairs but misses another 796 pairs. By TNP, 288 of the keyframe pairs missed by SR-PE are correctly recovered but in the process 37 of the keyframe pairs originally detected by SR-PE are lost. In the end, a total of 2839 numbers of near-duplicate shots is detected through TNP, a 9.7% improvement in recall performance compared to the keyframe-based approach. For the negative samples, SR-PE successfully identifies all the non near-duplicate keyframe pairs. When applied to the set of aligned frames, SR-PE maintains a decent performance on the negative set, with only 1 false positive. In average, 3 additional frame-pair computations are required to process a shot but there exists indexing techniques, e.g. the inverted file to speed up the process.

In addition, TNP detects two erroneously labeled frame pairs (see Figure 7 for one of the mislabeled pairs). The labeling error is transparent to a human annotator since a single keyframe is insufficient to handle the diverse contents in these shots. In general, we can identify three types of shots where TNP is useful. The first category is the strong-motion shot (Figure 5) where two near-duplicate shots are affected by mis-alignments among the keyframes. After alignment by TNP, they could be correctly identified as near-duplicate shots. The second category is the subtle-motion shot where subtle shift in scale (Figure 6(a)) and viewpoint (Figure 6(b) and 6(c)) results in a significant distortion at the low-level feature space. The problem is alleviated after performing alignment where the aligned frame pairs are correctly detected by SR-PE. The third category is the multi-segment shot (Figure 7) which happens when multiple unrelated segments are erroneously grouped into a single shot as a result of inaccurate shot boundary detection.

8. CONCLUSIONS

We have presented a novel approach to detect and localize near-duplicate segments from two videos through the joint consideration of visual features and temporal coherency of frame sequence. Temporal constraint is embedded into a network structure and partial alignment is novelly posed as a network flow problem where efficient solutions exist. To increase precision, the set of must-links and cannot-links generated from keypoint matching is used to refine the boundaries of the overlapping segments. In our experiments, we demonstrate the scalability of our system by matching full-length movies to YouTube videos which takes only 3 min-

(a) (b) (c)

Figure 5: Motion-based shots. The top row shows the keyframes for two shots. The aligned frames generated by TNP are shown in the second and third rows.

(a) (b) (c)

Figure 6: Subtle motion shots. The top row shows the keyframes for two shots. The second row shows the aligned frames generated by TNP.

utes to complete per movie. In addition, the experiments display satisfactory detection and localization performance in the various datasets where the proposed algorithm outperforms other state-of-the-art algorithms. Lastly, we improve the performance of near-duplicate shot detection by tackling the misalignment problem in the keyframe-based approach. For future work, we plan to extend our work to broadcast videos where recurring video segments [3] are commonly found. The hyper-links among the partial-duplicate segments will be explored to improve the performance of topic tracking and story summarization.

ACKNOWLEDGEMENT

The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (CityU 119508).

9. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Reading, Massachusetts, 1993.

[2] O. Chum, J. Philbin, M. Isard, and A. Zisserman. Scalable near identical image and shot detection. *CIVR*, 2007.

[3] I. Döhring and R. Lienhart. Mining tv broadcasts for recurring video sequences. *CIVR*, 2009.

[4] W. Dong, Z. Wang, M. Charikar, and K. Li. Efficiently matching sets of features with random histograms. *ACM Multimedia*, 2008.

[5] D. Goldfarb and Z. Jin. An o(nm)-time network simplex algorithm for the shortest path problem. *Operations Research*, 47(3):445–448, 1999.

[6] Hampapur, A. Bolle, M. Rudolf, and K.-H. Hyun. Comparison of sequence matching techniques for video copy detection. *SPIE: Storage and Retrieval for Media Databases*, 2001.

[7] W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Video search reranking through random walk over document-level context graph. *ACM Multimedia*, 2007.

Figure 7: A multi-segment shot (top row). The keyframes are highlighted in bold frame and the alignment results are shown.

[8] X.-S. Hua and H.-J. Z. X. Chen. Robust video signature based on ordinal measure. *ICIP*, 2004.

[9] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *ECCV*, 2008.

[10] Y.-G. Jiang and C.-W. Ngo. Visual word proximity and linguistics for semantic video indexing and near-duplicate retrieval. *CVIU*, 113(3):405–414, March 2009.

[11] J.Law-To, A. Joly, and N. Boujemaa. Muscle-vcd-2007: a live benchmark for video copy detection. *http://www.-rocq.inria.fr/media/civr/bench*, 2007.

[12] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *CVPR*, 2004.

[13] Y. Ke, R. Sukthankar, and L. Huston. Efficient near duplicate detection and sub-image retrieval. *ACM Multimedia*, 2004.

[14] L. Kennedy and S.-F. Chang. Internet image archaeology: Automatically tracing the manipulation history of photographs on the web. *ACM Multimedia*, 2008.

[15] H. Kim, J. Lee, H. Liu, and D. Lee. Video linkage: Group based copied video detection. *CIVR*, 2008.

[16] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa. Robust voting algorithm based on labels of behavior for video copy detection. *ACM Multimedia*, 2006.

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[18] K. Milolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[19] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. *ACM MIR*, 2006.

[20] H.-K. Tan, X. Wu, C.-W. Ngo, and W.-L. Zhao. Accelerating near-duplicate video matching by combining visual similarity and alignment distortion. *ACM Multimedia*, 2008.

[21] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. *ICML*, 2001.

[22] X. Wu, A. G. Hauptman, and C.-W. Ngo. Practical elimination of near-duplicates from web search video. *ACM Multimedia*, 2007.

[23] X. Wu, C.-W. Ngo, A. G. Hauptmann, and H.-K. Tan. Real-time near-duplicate elimination for web video search with content and context. *TMM*, 11(2):196–207, Feb. 2009.

[24] X. Wu, M. Takimoto, S. Sato, and J. Adachi. Scene duplicate detection based on the pattern of discontinuities in feature point trajectories. *ACM Multimedia*, 2008.

[25] D. Xu and S.-F. Chang. Video event recognition using kernel methods with multilevel temporal alignment. *TPAMI*, 30(11):1985–1997, 2008.

[26] D. Zhang and S.-F. Chang. Detecting image near duplicate by stochastic attribute relational graph matching with learning. *ACM Multimedia*, 2004.

[27] W.-L. Zhao and C.-W. Ngo. Scale-rotation invariant pattern entropy for keypoint-based near-duplicate detection. *TIP*, 18(2):412–423, 2009.

[28] W.-L. Zhao, C.-W. Ngo, H.-K. Tan, and X. Wu. Near duplicate keyframe identificiation with interest point matching and pattern learning. *TMM*, 9(5):213–238, 2007.